Universidad ORT Uruguay

Facultad de Ingeniería

Aplicación de técnicas de ingeniería de software en emprendimientos de software mediante mentoría y guías de apoyo

Entregado como requisito para la obtención del título de Ingeniero en Sistemas

Federico Riaño - 207356

Santiago Alpuy - 247034

Tutor: Gerardo Matturro

2023

Declaración de autoría

Nosotros, Federico Riaño y Santiago Alpuy, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizamos el proyecto de grado.
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

	
Federico Riaño	Santiago Alpuy
Montevideo, 28 de septiembre de 2023	

Agradecimientos

En primer lugar, deseamos expresar nuestro más profundo agradecimiento al Dr. Gerardo Matturro por su invaluable orientación, paciencia y experticia. Su compromiso con esta investigación ha contribuido significativamente al desarrollo de este proyecto.

Además, queremos agradecer sinceramente a todos los encuestados y entrevistados que participaron en este estudio. Su disposición para compartir sus experiencias, conocimientos y tiempo ha sido crucial para nuestro trabajo.

Abstract

El adecuado uso de técnicas de ingeniería de software se presenta como un desafío fundamental. A pesar de que el 90% de los emprendimientos de software fracasa, sigue existiendo una notable brecha en el soporte específico para la construcción y diseño de soluciones basadas en software. Este trabajo se centró en analizar las técnicas de ingeniería de software más empleadas en emprendimientos a través de la revisión de 315 artículos académicos relacionados a este tipo de emprendimientos. Como resultado, se identificaron 12 técnicas predominantes para las cuales se desarrollaron guías de aplicación práctica, las cuales se validaron con seis expertos, conformados por docentes del laboratorio de software, egresados que realizaron proyectos de investigación sobre emprendimientos de software y egresados, los cuales su proyecto final fue un emprendimiento de software.

Adicionalmente también se realizaron validaciones a las guías por una modalidad en retrospectiva, comparando las disposiciones de dichas guías con la experiencia en la aplicación en sus proyectos, con dos egresados que realizaron proyecto de emprendedurismo y dos estudiantes que al momento de realizar esta validación estaban cursando el proyecto de fin de carrera del tipo emprendimiento.

Todas las validaciones anteriormente mencionadas se han llevado a cabo mediante diferentes métodos, como entrevistas y encuestas. Cabe destacar que todas las personas contactadas tienen o han tenido relación con Universidad ORT Uruguay.

Además, se diseñó un plan de mentoría para facilitar la integración de estas guías en los emprendimientos de software. Aunque se encontraron limitaciones para la aplicación directa de este plan en un emprendimiento universitario en Universidad ORT Uruguay, tanto las guías de apoyo como el diseño del plan de mentoría propuesto acabaron siendo recursos muy bien valorados para la utilización por parte de emprendedores en búsqueda de consolidar y mejorar sus prácticas de ingeniería de software.

Palabras Clave

Software startup, Emprendimiento de software, Ingeniería de software, Emprendedurismo, Mentoría en emprendimientos, Técnicas de Ingeniería de Software.

Índice

1. Introducción 12

2. Marco Teórico 14

2.1. Emprendedurismo 14

2.2. Emprendimiento de Software (Startup) 14

2.4. Ingeniería de Software 15

2.5. SWEBOK, Software Engineering Body of Knowledge 16

2.5.1. Requisitos de Software 17

2.5.2. Diseño de Software 17

2.5.3. Construcción de software 18

2.5.4. Pruebas de Software 18

2.5.5. Mantenimiento de Software 19

2.5.6. Gestión de la configuración de software 20

2.5.7. Gestión de la Ingeniería de Software 20

2.5.8. Proceso de Ingeniería de Software 21

2.5.9. Modelos y métodos de la Ingeniería de Software 21

2.5.10. Calidad del Software 22

2.5.11. Práctica Profesional de la Ingeniería de Software 22

2.5.12. Economía de la Ingeniería de Software 23

2.5.13. Fundamentos de Computación 23

2.5.14. Fundamentos Matemáticos 23

2.5.15. Fundamentos de Ingeniería 24

2.6. Modelo Lean Startup 24

2.7. BML (Build-Measure-Learn) 26

2.8. Mentoría 26

2.9. Mentoría para emprendimientos 27
2.10. Revisiones rápidas en ingeniería de software. 28
2.11. Mentoría en software para emprendimientos: Trabajos vinculados 29
3. Problema de investigación 30
4. Diseño metodológico 31
4.1. Objetivos 31
4.1.1. Objetivos Generales 31
4.1.2. Objetivos Específicos 31
4.2. Preguntas de investigación 32
4.3. Fase 1: Identificación de técnicas 32
4.3.1. Preparación y ordenamiento de los artículos 32
4.3.2. Obtención de técnicas de ingeniería de software 32
4.3.3. Validación de completitud y exactitud 33
4.3.4. Revisión cruzada de artículos académicos 33
4.3.5. Agrupamiento de técnicas de ingeniería de software 34
4.3.6. Obtención de cantidad de ocurrencias de técnicas de ingeniería de software 34
4.3.7. Definición de parámetros para la selección de las técnicas 34
4.3.8. Selección de las técnicas de ingeniería de software para las guías de apoyo 35
4.4. Fase 2: Diseño de guías de apoyo y del proceso de mentoría 35
4.4.1. Diseño de guías de apoyo 35
4.4.2. Diseño de plan de mentoría 36
4.5. Fase 3: Validación e implementación de guías y plan de mentoría 36
4.5.1. Búsqueda y selección de personas para validación de guías 36
4.5.2. Agrupamiento de expertos en grupos heterogéneos 36
4.5.3. Diseño y distribución de encuestas 36

4.5.4. Análisis de las encuestas a expertos 37
4.5.5. Diseño de entrevistas a expertos 38
4.5.6. Análisis de las entrevistas a expertos 38
4.5.7. Búsqueda de equipos de proyectos 38
4.5.8. Inicio y ejecución del plan de mentoría 39
5. Resultados obtenidos 40
5.1. Reporte de resultados de fase 1: Identificación y clasificación de prácticas 40
5.1.1. Preparación de los artículos académicos 40
5.1.2. Análisis de técnicas de software en artículos académicos 41
5.1.3. Validación de información recolectada en exactitud y completitud 41
5.1.4. Revisión cruzada de artículos académicos 42
5.1.5. Agrupamiento de técnicas de ingeniería de software 43
5.1.6. Contabilidad de técnicas de ingeniería de software obtenidas 43
5.1.7. Definición de parámetros para la selección final de las técnicas 44
5.1.8. Selección de las técnicas de ingeniería de software para las guías 45
5.2. Reporte de resultados del diseño de guías de apoyo y del plan de mentoría 52
5.2.1. Diseño de guías de apoyo 52
5.2.2. Diseño del plan de mentoría 54
5.2.2.1. Diseño de la estructura del plan de mentoría 54
5.2.2.2. Plan de mentoría 56
5.3. Reporte de validación e implementación de guías y plan de mentoría 61
5.3.1 Búsqueda y selección de expertos para validación de guías 61
5.3.2. Formación de grupos, diseño de encuestas y respuestas de expertos 62
5.3.2.1. Creación de grupos heterogéneos 63
5.3.2.2. Diseño y distribución de encuestas 64
5.3.2.3. Respuestas de expertos 64

5.3.3. Diseño de entrevistas a expertos 68
5.3.4. Realización de entrevistas a expertos 71
5.3.5. Búsqueda de equipos emprendedores y presentación de guías y mentoría 72
5.3.6. Inicio y ejecución del plan de mentoría 73
5.3.7. Validación de guías por retrospectiva 74
5.3.8. Distribución de encuesta de validación en retrospectiva 75
5.3.9. Encuestas de validación en retrospectiva 76
5.3.10. Validación de guías a través del uso práctico 86
5.3.10.1. Preparación y ejecución de entrevista con estudiantes de proyecto 86
6. Análisis y discusión de resultados 89
6.1. Análisis de encuesta de validación de guías de apoyo por expertos 89
6.1.1. Análisis de preguntas sobre consideraciones generales de las guías 89
6.1.2. Análisis de preguntas sobre consideraciones específicas de las guías 90
6.2. Análisis de entrevistas de validación de guías de apoyo por expertos 92
6.3. Análisis de encuestas de validación de guías de apoyo en retrospectiva 97
6.4. Análisis de guías de apoyo a través del uso práctico 102
6.5. Respuesta a preguntas de investigación 103
7. Conclusiones 110
8. Lecciones aprendidas 112
9. Amenazas a la validez del estudio 114
9.1. Dependencia 114
9.2. Transferencia 114
9.3. Transferencia y aplicabilidad de resultados 115
10. Propuestas de trabajos futuros de investigación 116
11. Referencias bibliográficas 117

ANEXO 1 - Guía de encuestas 121
ANEXO 2 - Guía de entrevistas 126
ANEXO 3 - Guía de estudio de casos 130
ANEXO 4 - Guía de integración y despliegue continuo 134
ANEXO 5 - Guía de manejo de deuda técnica 137
ANEXO 6 - Guía de mockups 142
ANEXO 7 - Guía de observación 146
ANEXO 8 - Guía de pruebas A/B 150
ANEXO 9 - Guía de prototipos 154
ANEXO 10 - Guía de pruebas de usabilidad 158
ANEXO 11 - Guía de pruebas unitarias 163
ANEXO 12 - Guía de Wireframes 168
ANEXO 13 - Script contador de ocurrencias de técnicas de ingeniería de software 171
ANEXO 14 - Formulario para validación de guías de apoyo por expertos del Grupo 1 172
ANEXO 15 - Formulario para validación de guías de apoyo por expertos del Grupo 2 174
ANEXO 16 - Formulario para validación de guías de apoyo por expertos del Grupo 3 176
ANEXO 17 - Transcripciones de entrevistas a expertos 178
ANEXO 18 - Transcripción de entrevista a equipo de proyecto en conjunto con industria 190
ANEXO 19 - Agrupamiento de técnicas de ingeniería de software 193
ANEXO 20 – Subconjunto de Artículos Relevantes 199

Índice de tablas

Tabla 1. Descripción de expertos contactados durante la investigación 63
Tabla 2. Asignación de grupos de expertos y técnicas 64
Tabla 3. Resultados de preguntas generales realizadas a expertos 65
Tabla 4. Resultados de preguntas específicas realizadas al grupo 1 66
Tabla 5. Resultados de preguntas específicas realizadas al grupo 2 67
Tabla 6. Resultados de preguntas específicas realizadas al grupo 3 68
Tabla 7. Preguntas preparadas para entrevista con Mario Camerota 69
Tabla 8. Preguntas preparadas para entrevista con Andrés Bentos 70
Tabla 9. Preguntas preparadas para entrevista con Darío Macchi 70
Tabla 10. Preguntas preparadas para entrevista con Rodrigo Stratta 71
Tabla 11. Preguntas preparadas para entrevista con Pablo Benitez 71
Tabla 12. Ejemplo de sección de encuesta de validación por retrospectiva 75
Tabla 13. Técnicas aplicadas por participantes de acuerdo con su experiencia previa 78
Tabla 14. Respuestas a preguntas sobre la guía de apoyo de estudio de casos 79
Tabla 15. Respuestas a preguntas sobre la guía de apoyo de pruebas unitarias 80
Tabla 16. Respuestas a preguntas sobre la guía de apoyo de prototipo 81
Tabla 17. Respuestas a preguntas sobre la guía de apoyo de mockups 81
Tabla 18. Respuestas a preguntas sobre la guía de apoyo de entrevistas 82
Tabla 19. Respuestas a preguntas sobre la guía de apoyo de observación 83
Tabla 20. Respuestas a preguntas sobre la guía de apoyo de encuestas 84
Tabla 21. Respuestas a preguntas sobre la guía de apoyo de manejo de DT 85
Tabla 22. Respuestas a preguntas sobre la guía de apoyo de CI/CD 86
Tabla 23. Preparación de entrevista sobre aplicación práctica de guías 88

1. Introducción

En la actualidad, el mundo de los emprendimientos de software se presenta como un panorama en constante evolución, donde las innovaciones y cambios son bastante frecuentes. Sin embargo, a pesar de las múltiples oportunidades que estos emprendimientos emergentes ofrecen, enfrentan retos significativos, especialmente en el ámbito de la ingeniería de software. El presente trabajo lleva por título "Aplicación de técnicas de ingeniería de software en emprendimientos de software mediante mentoría y guías de apoyo", y surge como una respuesta a la necesidad de comprender y mejorar la forma en que estos emprendimientos abordan los aspectos de la ingeniería de software.

En este complejo ecosistema, la definición de funcionalidades del producto, la toma de decisiones técnicas fundamentales, la rápida iteración y prototipado, y la comunicación eficaz dentro y fuera del equipo, son desafíos presentes en todo momento. Estos retos se quedan en evidencia aún más cuando se considera que hasta el 90% [1] de estas empresas fracasan o que dos tercios de ellas no generan rendimientos positivos para sus inversores [2]. Aunque el fracaso de un emprendimiento de software puede atribuirse a múltiples factores, las prácticas deficientes de ingeniería de software son frecuentemente señaladas como una razón principal [3] y [4].

El panorama actual revela que, aunque estos emprendimientos reconocen la importancia de aplicar técnicas de ingeniería de software adecuadas, frecuentemente enfrentan dificultades en seleccionar y aplicar las más apropiadas para sus necesidades [5] y [6]. Es más, aunque las incubadoras ofrecen mentoría y apoyo en diversas áreas, hay una notable carencia de orientación específica en ingeniería de software, dejando un vacío crítico en uno de los pilares esenciales para el éxito de un emprendimiento de este tipo.

Ante esta realidad, nuestro estudio tiene el propósito de ofrecer a los emprendimientos de software las herramientas y orientación necesarias en ingeniería de software para

potenciar sus probabilidades de éxito. A través de una investigación exhaustiva y la formulación de guías prácticas, buscamos ser el puente entre la teoría y la aplicación efectiva de las mismas. Estamos convencidos de que, con el adecuado enfoque y mentoría en ingeniería de software, estas empresas emergentes pueden incrementar dichas probabilidades de éxito.

2. Marco Teórico

2.1. Emprendedurismo

El emprendedurismo es un tema que ha despertado mucho interés en el mundo académico dentro del área de negocios. Sin embargo, estudiarlo puede ser complicado porque no hay una única definición clara sobre qué es. Aunque este concepto ha estado presente por mucho tiempo y ha sido clave para el progreso económico y social, aún no hay un consenso total sobre su definición. Sin embargo, todos coinciden en que incluye aspectos como: características especiales, tomar riesgos, ver oportunidades, tener motivación e innovar [7].

Desde una perspectiva económica, el emprendedurismo generalmente se conceptualiza como la creación de un nuevo negocio y la asunción del riesgo asociado, a cambio de ganancias que derivan de la explotación de oportunidades de mercado. Definido de esta manera, el emprendedurismo puede tomar una variedad de formas. Una de las más famosas, es la "creación destructiva" de Schumpeter. Según Schumpeter, el emprendedor está impulsado por la innovación, que puede tomar la forma de un nuevo producto, proceso, o cambio innovador en los productos o procesos existentes, que en última instancia destruyen o vuelven obsoletos los productos y procesos que se han utilizado en el pasado [7]. Por otro lado, para Byers y Nelson [8], emprender va más allá de solo abrir un negocio y ganar dinero. Ven el emprendedurismo como el camino que siguen personas o equipos para juntar lo que necesitan, aprovechar oportunidades y, al final, generar riqueza y bienestar para todos.

2.2. Emprendimiento de Software (Startup)

Un emprendimiento de software puede definirse como una empresa de reciente creación sin historial operativo que se centra en el desarrollo de tecnologías de vanguardia, en particular productos o servicios de software [9]. Entre los principales desafíos de los emprendimientos de software es frecuente que se mencionen su escasez de recursos, ser altamente reactivas, estar formadas por pequeños equipos con poca experiencia, confiar en un solo producto, y comenzar a operar en condiciones de incertidumbre, de rápida evolución, con presión del tiempo, altos riesgos y dependencia [3].

En cuanto a las etapas de un emprendimiento de software, según Lee Von Kraus [10], durante la etapa temprana, el emprendimiento se dedica principalmente a confirmar la viabilidad de su propuesta, trabajando en la creación de una versión básica del producto conocida como mínimo producto viable (MVP). Es crucial en este periodo comprobar la validez de la propuesta de negocio con posibles consumidores, desarrollar un MVP que atienda una necesidad concreta y generar interés entre los clientes.

Al avanzar a la etapa semilla, el emprendimiento se adentra en la búsqueda activa de financiamiento, ya sea a través de inversores ángeles o de capital de riesgo. Este capital tiene como objetivo principal avanzar en el desarrollo de un producto o servicio integral y su posterior introducción al mercado.

Una vez que el emprendimiento entra en la fase de crecimiento y establecimiento, la prioridad se centra en expandir su presencia en el mercado. Es esencial en esta etapa incrementar el volumen de clientes, elevar las cifras de ventas e ingresar en nuevos mercados.

Al alcanzar la madurez, el emprendimiento ya ha logrado un ritmo estable y sostenido de crecimiento. Aquí, el mismo se concentra en mantener ese crecimiento, maximizar sus utilidades y, cuando es posible, diversificar su oferta introduciendo nuevos productos o servicios.

En la etapa de adquisición, el emprendimiento se posiciona para fusionarse o ser comprado por una organización más grande. El objetivo en este punto es asegurar un acuerdo de compra que sea beneficioso y facilitar una integración eficiente con la empresa compradora.

2.4. Ingeniería de Software

La ingeniería de software es una disciplina de ingeniería que se interesa por todos los aspectos de la producción de software, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del mismo después de que se pone en operación. En esta definición se presentan dos frases clave:

- Disciplina de ingeniería. Los ingenieros hacen que las cosas funcionen. Aplican teorías, métodos y herramientas donde es adecuado. Sin embargo, los usan de manera selectiva y siempre tratan de encontrar soluciones a problemas, incluso cuando no hay teorías ni métodos aplicables. Los ingenieros también reconocen que deben trabajar ante restricciones organizacionales y financieras, de modo que buscan soluciones dentro de tales limitaciones.

- Todos los aspectos de producción de software. La ingeniería de software no sólo se interesa por los procesos técnicos del desarrollo de software, sino también incluye actividades como la administración del proyecto de software y el desarrollo de herramientas, así como métodos y teorías para apoyar la producción de software.

La ingeniería busca obtener resultados de la calidad requerida dentro de la fecha y del presupuesto. A menudo esto requiere contraer compromisos: los ingenieros no deben ser perfeccionistas. Sin embargo, las personas que diseñan programas para sí mismas podrían pasar tanto tiempo como deseen en el desarrollo del programa.

En general, los ingenieros de software adoptan en su trabajo un enfoque sistemático y organizado, pues usualmente ésta es la forma más efectiva de producir software de alta calidad. No obstante, la ingeniería busca seleccionar el método más adecuado para un conjunto de circunstancias y, de esta manera, un acercamiento al desarrollo más creativo y menos formal sería efectivo en ciertas situaciones. El desarrollo menos formal es particularmente adecuado para la creación de sistemas basados en la Web, que requieren una mezcla de habilidades de software y diseño gráfico.[11]

2.5. SWEBOK, Software Engineering Body of Knowledge

Esta guía [12] se trata de un documento creado por la *Software Engineering Coordinating Committee*, este fue promovido por el IEEE *Computer Society*, es una guía de las principales temáticas que se encuentran en el área de la Ingeniería de Software. La versión 2005 se publicó como estándar [13].

A continuación, detallaremos los principales aportes de esta guía, estos se descomponen a áreas de conocimiento y las mismas son:

2.5.1. Requisitos de Software

El área de conocimiento de los requisitos de software se ocupa de la obtención, el análisis, la especificación y la validación de los requisitos de software, así como la gestión de los requisitos durante todo el ciclo de vida del producto de software.

Se reconoce ampliamente entre investigadores y profesionales de la industria que los proyectos de software son críticamente vulnerables cuando las actividades relacionadas con los requisitos se realizan de manera deficiente. Los requisitos de software expresan las necesidades y limitaciones impuestas a un producto de software que contribuyen a la solución de algún problema del mundo real.

Una propiedad esencial de todos los requisitos de software es que sean verificables como característica individual, como requisito funcional o a nivel del sistema como requisito no funcional.

2.5.2. Diseño de Software

El diseño puede entenderse como el método meticuloso que establece la arquitectura, los componentes, las interfaces y otras características relevantes de un sistema o componente. Además, el diseño no solo se refiere al proceso en sí, sino también al resultado final obtenido a partir de este procedimiento

El diseño de software representa una etapa crucial dentro del ciclo de vida de la ingeniería de software. Durante esta fase, se lleva a cabo un análisis exhaustivo de los requisitos del software con el objetivo de generar una descripción detallada de su estructura interna. Esta descripción, a su vez, proporcionará un fundamento sólido para la construcción subsiguiente del software.

El diseño de software juega un papel importante en el desarrollo del mismo, durante el diseño, los ingenieros de software crean múltiples modelos que configuran una especie de plano detallado para la solución a ser implementada. Estos modelos pueden ser analizados y evaluados con el fin de verificar su capacidad para satisfacer los diversos requisitos planteados

2.5.3. Construcción de software

La construcción de software se refiere a la creación detallada de software funcional a través de una combinación de codificación, verificación, pruebas unitarias, pruebas de integración y depuración. El área de conocimiento de la Construcción de Software está vinculada a todas las demás áreas de conocimiento, pero tiene fuertes vínculos con el Diseño de Software y las Pruebas de Software, ya que el proceso de construcción de software implica un diseño y pruebas de software significativos. Este proceso emplea la salida del diseño y proporciona una entrada para las pruebas.

El área de conocimiento de la Construcción de Software también se relaciona estrechamente con la Gestión de Configuración de Software, dado que la construcción de software generalmente produce la mayor cantidad de elementos de configuración que necesitan ser gestionados en un proyecto de software, tales como archivos fuente, documentación, casos de prueba, entre otros.

2.5.4. Pruebas de Software

Las pruebas de software consisten en la verificación dinámica de que un programa proporciona los comportamientos esperados en un conjunto finito de casos de prueba, seleccionados adecuadamente del dominio de ejecución, el cual generalmente es infinito.

Algunas características que se deben satisfacer en las pruebas de software son:

Dinámico: este término significa que la prueba siempre implica ejecutar el programa en entradas seleccionadas. Para ser precisos, el valor de entrada por sí solo no siempre es suficiente para especificar una prueba, ya que un sistema complejo y no determinista podría

reaccionar a la misma entrada con diferentes comportamientos, dependiendo del estado del sistema.

Finito: incluso en programas simples, teóricamente son posibles tantos casos de prueba que la prueba exhaustiva podría requerir meses o años para ejecutarse. Es por eso que, en la práctica, un conjunto completo de pruebas generalmente puede considerarse infinito y las pruebas se realizan en un subconjunto de todas las pruebas posibles, determinado por criterios de riesgo y priorización.

Seleccionado: las muchas técnicas de prueba propuestas difieren esencialmente en cómo se selecciona el conjunto de pruebas y los ingenieros de software deben ser conscientes de que diferentes criterios de selección pueden producir grados muy diferentes de efectividad.

Esperado: debe ser posible, aunque no siempre sea fácil, decidir si los resultados observados de las pruebas del programa son aceptables o no; de lo contrario, el esfuerzo de prueba es inútil.

En los últimos años, la visión de la prueba de software se ha desarrollado de manera constructiva. La prueba ya no se ve como una actividad que comienza sólo después de que se completa la fase de codificación con el propósito limitado de detectar fallas. La prueba de software es o debería ser omnipresente en todo el ciclo de vida de desarrollo y mantenimiento. De hecho, la planificación de la prueba de software debería comenzar en las primeras etapas del proceso de requisitos de software, los planes y procedimientos

2.5.5. Mantenimiento de Software

Los esfuerzos de desarrollo de software dan como resultado la entrega de un producto de software que satisface los requisitos del usuario. En consecuencia, este producto debe cambiar o evolucionar. Una vez en funcionamiento, se descubren los defectos, cambian los entornos operativos y surgen nuevos requisitos del usuario. La fase de mantenimiento del ciclo de vida comienza después de un período de garantía o de una entrega de soporte posterior a la implementación, pero las actividades de mantenimiento ocurren mucho antes.

Este es una parte integral del ciclo de vida del software. Sin embargo, no ha recibido el mismo grado de atención que las demás fases. Históricamente, el desarrollo de software ha tenido un perfil mucho más alto que el mantenimiento de software en la mayoría de las organizaciones.

Según esta guía, el mantenimiento de software se define como la totalidad de actividades necesarias para proporcionar un soporte rentable al software. Estas actividades se realizan tanto en la etapa previa a la entrega como en la etapa posterior a la misma. Las actividades previas a la entrega incluyen la planificación de las operaciones posteriores a la entrega, la mantenibilidad y la determinación logística para las actividades de transición. Las actividades posteriores a la entrega incluyen la modificación del software, la formación y la operación o interconexión con un servicio de ayuda.

2.5.6. Gestión de la configuración de software

Un sistema puede definirse como la combinación de elementos que interactúan organizados para lograr uno o más propósitos establecidos. La configuración de un sistema son las características físicas y funcionales del hardware o software tal como se establece en la documentación técnica o se logra en un producto; también se puede considerar como una colección de versiones específicas de elementos de hardware, firmware o software combinados de acuerdo con procedimientos de compilación específicos.

La gestión de configuración del software (SCM) es un proceso de soporte en el ciclo de vida del software que beneficia la gestión del proyecto, las actividades de desarrollo y mantenimiento, las actividades de aseguramiento de la calidad, así como a los clientes y usuarios del producto final. Los conceptos de gestión de configuración se aplican a todos los elementos a controlar.

2.5.7. Gestión de la Ingeniería de Software

La gestión de ingeniería de software se puede definir como la aplicación de actividades de gestión (planificación, coordinación, medición, seguimiento, control y generación de informes) para garantizar que los productos de software y los servicios de ingeniería de software se entreguen de manera eficiente, eficaz y en beneficio de las partes interesadas.

Las actividades de gestión de ingeniería de software ocurren en tres niveles: gestión organizativa y de infraestructura, gestión de proyectos y gestión del programa de medición.

2.5.8. Proceso de Ingeniería de Software

Un proceso de ingeniería consiste en un conjunto de actividades interrelacionadas que transforman una o más entradas en salidas mientras consumen recursos para lograr la transformación. Muchos de los procesos de las disciplinas tradicionales de la ingeniería (p. ej., eléctrica, mecánica, civil, química) se ocupan de transformar la energía y las entidades físicas de una forma a otra, como en una represa hidroeléctrica que transforma la energía potencial en energía eléctrica o una refinería de petróleo que utiliza procesos químicos para transformar el petróleo crudo en gasolina.

Los procesos de software se especifican por varias razones: para facilitar la comprensión, la comunicación y la coordinación humanas; para ayudar en la gestión de proyectos de software; medir y mejorar la calidad de los productos de software de manera eficiente; para apoyar la mejora de procesos; y proporcionar una base para el soporte automatizado de la ejecución del proceso.

2.5.9. Modelos y métodos de la Ingeniería de Software

Los modelos y métodos de ingeniería de software imponen una estructura con el objetivo de hacer que esa actividad sea sistemática, repetible y, en última instancia, más orientada al éxito.

La utilización de modelos ofrece una metodología eficaz para la resolución de problemas, proporcionando una notación clara y procedimientos detallados para la construcción y el análisis de estos modelos. Los métodos ofrecen una estrategia sistemática para la especificación, el diseño, la construcción, el ensayo y la verificación del software del producto final, así como para los productos de trabajo relacionados.

Los modelos y métodos de ingeniería de software varían ampliamente en su alcance, desde abordar una sola fase del ciclo de vida del software hasta cubrir el ciclo de vida completo del mismo.

2.5.10. Calidad del Software

La calidad de software es un término amplio y sobrecargado que se refiere a las características deseables de los productos de software, el grado en que un producto de software en particular posee esas características y los procesos, herramientas y técnicas utilizados para lograr. Las definiciones de calidad de software varían según el autor y la organización. El objetivo principal de cualquier producto fabricado es entregar el máximo valor para las partes interesadas, mientras se equilibran las restricciones del costo y el cronograma de desarrollo. El valor para los interesados se expresa en los requisitos. Para los productos de software, los interesados podrían valorar el precio (lo que pagan por el producto), el tiempo de entrega (la rapidez con la que reciben el producto) y la calidad del software.

Los requisitos de calidad del software son en realidad atributos o restricciones de los requisitos funcionales. La calidad del software se logra mediante la conformidad con todos los requisitos, independientemente de las características especificadas o de cómo se agrupan o nombran los mismos.

2.5.11. Práctica Profesional de la Ingeniería de Software

El área de conocimiento de Práctica Profesional de Ingeniería de Software se ocupa del conocimiento, las habilidades y las actitudes que los ingenieros de software deben poseer para practicar la ingeniería de software de manera profesional, responsable y ética. Debido a las aplicaciones generalizadas de los productos de software en la vida social y personal, la calidad de los productos de software puede tener un profundo impacto en nuestro bienestar personal y armonía social. Los ingenieros de software deben manejar problemas de ingeniería únicos, produciendo software con características y confiabilidad conocidas.

El término “práctica profesional” se refiere a una manera de realizar servicios para lograr ciertos estándares o criterios tanto en el proceso de prestación de un servicio como en el

producto final resultante del mismo. Estos estándares y criterios pueden incluir aspectos tanto técnicos como no técnicos.

2.5.12. Economía de la Ingeniería de Software

La economía de la ingeniería de software trata de tomar decisiones relacionadas con la ingeniería de software en un contexto empresarial. El éxito de un producto, servicio o solución de software depende de una buena gestión empresarial. Sin embargo, en muchas empresas y organizaciones, las relaciones comerciales de software con el desarrollo y la ingeniería de software siguen siendo vagas.

La economía es el estudio del valor, los costos, los recursos y su relación en un contexto o situación determinada. En la disciplina de la ingeniería de software, las actividades tienen costos, pero el software resultante también tiene atributos económicos. La economía de la ingeniería de software proporciona una forma de estudiar los atributos del software y los procesos de una manera sistemática que los relaciona con medidas económicas.

2.5.13. Fundamentos de Computación

El alcance del área de conocimiento de Fundamentos de computación abarca el entorno operativo y de desarrollo en el que el software evoluciona y se ejecuta. Porque ningún software puede existir o funcionar sin una computadora, el núcleo de dicho entorno es la computadora y sus diversos componentes. El conocimiento sobre la computadora y sus principios subyacentes de hardware y software sirve como marco en el que se ancla la ingeniería de software.

2.5.14. Fundamentos Matemáticos

El área de conocimiento de Fundamentos matemáticos ayuda a los ingenieros de software a comprender esta lógica, que a su vez se traduce en código de lenguaje de programación. Las matemáticas que son el enfoque principal en esta área de conocimiento son bastante

diferentes de la aritmética típica, donde se tratan y discuten los números. La lógica y el razonamiento son la esencia de las matemáticas que un ingeniero de software debe abordar.

2.5.15. Fundamentos de Ingeniería

IEEE define la ingeniería como "*la aplicación de un enfoque sistemático, disciplinado y cuantificable a estructuras, máquinas, productos, sistemas o procesos*"

A medida que la teoría y la práctica de la ingeniería de software madura, es cada vez más evidente que la ingeniería de software es una disciplina de ingeniería que se basa en conocimientos y habilidades comunes a todas las disciplinas de ingeniería. Esta área de conocimiento de Fundamentos de Ingeniería se ocupa de los fundamentos de ingeniería que se aplican a la ingeniería de software y a otras disciplinas de ingeniería. Los temas de esta área de conocimiento incluyen métodos empíricos y técnicas experimentales, análisis estadístico, medición, diseño de ingeniería, modelado, prototipado y simulación, estándares y análisis de causa raíz. La aplicación de este conocimiento, según corresponda, permitirá a los ingenieros de software desarrollar y mantener software de manera más eficiente y efectiva. Completar su trabajo de ingeniería de manera eficiente y efectiva es un objetivo de todos los ingenieros en todas las disciplinas de ingeniería.

2.6. Modelo Lean Startup

Metodología que favorece la experimentación sobre la planificación elaborada, la retroalimentación del cliente sobre la intuición y el diseño iterativo sobre el desarrollo tradicional de realizar un gran diseño por adelantado.

Conceptos, como producto mínimo viable y pivote, han llevado rápidamente al mundo de los emprendimientos de software a incursionar en ella, y consecuentemente, desafiando las nociones convencionales sobre el emprendimiento.

Este movimiento ha llevado a que las escuelas de negocios comenzaron a adaptar sus currículas para incluir estos conceptos.

Los nuevos emprendimientos de todo tipo intentan mejorar sus posibilidades de éxito siguiendo sus principios de fallar rápido y aprender continuamente.

El método Lean tiene tres principios clave: En primer lugar, en vez de dedicarse a meses de planificación e investigación, los empresarios aceptan que todo lo que tienen desde el primer día es una serie de hipótesis no probadas, básicamente, buenas conjeturas. Entonces, en lugar de escribir un intrincado plan de negocios, los fundadores resumen sus hipótesis en un marco llamado “esquema de modelo de negocios” (*business model canvas)*. Esencialmente, este es un diagrama de cómo una empresa crea valor para sí misma y para sus clientes.

En segundo lugar, las empresas lean utilizan un enfoque de "salir del edificio" llamado desarrollo de clientes (*customer development)*, para probar sus hipótesis. Salen y piden comentarios a los usuarios, compradores y socios potenciales sobre todos los elementos del modelo de negocio, incluidas las características del producto, los precios, los canales de distribución y las estrategias asequibles de adquisición de clientes. El énfasis está en la agilidad y la velocidad: las nuevas empresas ensamblan rápidamente productos mínimos viables e inmediatamente obtienen comentarios de los clientes. Luego, utilizando los aportes de los clientes para revisar sus suposiciones, comienzan el ciclo nuevamente, prueban las ofertas rediseñadas y realizan pequeños ajustes adicionales (iteraciones) o ajustes más sustantivos (pivotes) a las ideas que no funcionan.

Tercero, las startups lean practican algo llamado desarrollo ágil, que se originó en la industria del software. El desarrollo ágil funciona de la mano con el *customer development*. A diferencia de los ciclos típicos de desarrollo de productos que presuponen el conocimiento de los problemas de los clientes y las necesidades del producto, el desarrollo ágil elimina el tiempo y los recursos desperdiciados al desarrollar el producto de manera iterativa e incremental. Es el proceso mediante el cual las empresas emergentes crean los productos mínimos viables que prueban [14].

2.7. BML (Build-Measure-Learn)

Según Eric Ries [15], la actividad fundamental de un emprendimiento de software es transformar ideas en productos (*Build*), medir cómo los clientes responden (*Measure*), y luego aprender si pivotar o perseverar en el camino tomado (*Learn*). Todos los procesos de este tipo de emprendimiento exitoso deberían apuntar a acelerar este ciclo de retroalimentación.

2.8. Mentoría

El término mentoría deriva de mentor, que tiene un origen mitológico y fue utilizado por Homero en su obra La Odisea. Hace referencia a Mentor, personaje de esta novela, que quedó a cargo de educar (en el sentido de formar, guiar y aconsejar) a Telémaco, hijo de Ulises, para ser rey y sustituir a su padre llegado el momento.

Adey, Early y Foster [16] entienden que la relación de mentoría ocurre cuando una persona ayuda a otra en su desarrollo personal, a adquirir nuevas habilidades, interiorizar puntos de vista y a desarrollar su potencial. Para Conway [17], la mentoría es una relación privada entre dos personas, basada en un deseo mutuo para el desarrollo personal y profesional hacia un objetivo organizacional. Hezlett y Gibson [18] definen mentoría como una relación intensa y diádica en la que una persona con más experiencia, llamada mentor, brinda apoyo y asistencia a un colega más joven y menos experimentado, denominado protegido (*protégé*) o aprendiz (*mentee*). Por su parte, Jeong y Park [19] consideran que, si bien las definiciones operativas de mentoría varían a lo largo de la literatura, muchos académicos comparten consistentemente el concepto tradicional de que la tutoría describe una relación entre un mentor y un protegido o aprendiz. Un mentor generalmente se considera una persona mayor con más experiencia que brinda apoyo y asistencia personal y profesional a una persona menor con menos experiencia (es decir, aprendiz).

Starr [20] caracteriza al mentor como alguien que asume el papel de un asesor, partidario o maestro de confianza para aconsejar sabiamente a otra persona, y apoyar el aprendizaje, el desarrollo y el éxito final de esta. Para este autor, en particular, las razones por las cuales una

persona puede buscar un mentor son tanto personales como profesionales y, en este sentido, destaca las siguientes:

- Siente que le falta experiencia, contactos o conocimiento en un área o situación.
- Siente que falta algo en lo que saben, lo que hacen o cómo piensan, y quiere aprender de alguien que considera que puede ayudarlo a cerrar la brecha.
- Ha llegado a algún tipo de barrera o bloqueo, y siente que necesita una relación más personal con alguien que tenga experiencia directa en su tipo de situación.

2.9. Mentoría para emprendimientos

Las actividades de mentoría para socios fundadores de emprendimientos suelen darse en el marco de las incubadoras y aceleradoras de empresas.

La incubación de empresas es una combinación única y flexible de procesos de desarrollo empresarial organizados que brindan asistencia a empresas nuevas y empresas pequeñas incipientes al proporcionarles un apoyo crítico para sobrevivir y crecer en sus primeras etapas de desarrollo [21].

En cuanto a la aceleración de empresas, Yusubova y Clarysse [22] caracterizar las aceleradoras como programas de 3 a 6 meses de duración que ayudan a las nuevas empresas en las primeras etapas de desarrollo, al proporcionar servicios de apoyo como espacios de oficina (*co-work*), entrenamiento y mentoría, una pequeña cantidad de apoyo financiero, y un conjunto de programas educativos. Según estos autores, uno de los elementos más cruciales de las aceleradoras de negocios y la razón principal por la que el equipo de inicio participa en ellas, es la mentoría proporcionada por mentores de alta calidad.

Respecto a las actividades de mentoría en estos ámbitos, Tripathi y Oivo [23] agregan que el rol del mentor es guiar y capacitar a los fundadores de empresas emergentes y a los miembros del equipo para lograr las habilidades necesarias para el desarrollo de productos y del negocio, y que los mentores de empresas emergentes deban dedicar bastante tiempo durante

la etapa inicial para brindar el apoyo y la experiencia necesarios a los emprendedores sin experiencia. Según su estudio, la mentoría es uno de los factores de apoyo que juegan un papel crucial en la etapa inicial de los emprendimientos, donde los fundadores tienen como objetivo identificar el problema-solución y crear su equipo para establecer el producto mínimo viable o prototipo, para enfocarse en el estudio e identificación de lo que habitualmente se denomina ajuste del producto al mercado (*product-market fit*).

Por su parte, Busulwa, Birdthistle y Dunn [24] explican que en un programa de aceleración (*accelerator program*) las empresas emergentes suelen tener lo que se denominan sesiones de mentoría con los mentores del programa acelerador. Estas sesiones de mentoría implican pasar tiempo uno-a-uno o uno-a-equipo con fundadores experimentados, inversionistas, o especialistas en servicios comerciales profesionales (por ejemplo, abogados, contadores, expertos en modelos de negocio) y agregan que, por lo general, estas sesiones pueden variar de 15 minutos a unas pocas horas cada vez.

En relación con los programas o actividades de mentoría que incubadoras y aceleradoras suelen poner a disposición de los emprendimientos, diversos estudios indican que el aprendizaje a partir de la mentoría contribuye a resultados de aprendizaje importantes como habilidades de negocio y resiliencia [25] que, en última instancia, contribuyen a prevenir fallas y mejorar la supervivencia del emprendimiento [26]. La literatura más reciente también argumenta que la mentoría reduce la incertidumbre en el proceso empresarial y aumenta la seguridad y la legitimidad a través de la expansión del capital social y el modelado de roles [27].

2.10. Revisiones rápidas en ingeniería de software.

Las revisiones rápidas [28] son estudios orientados a la práctica. El objetivo principal de las revisiones rápidas es proporcionar evidencia para respaldar la toma de decisiones hacia la solución, o al menos la atenuación de los problemas que aparecen en la práctica.

Para respaldar este objetivo y cumplir con las limitaciones de tiempo, las revisiones rápidas deben entregar evidencia en plazos más cortos, en comparación con las revisiones sistemáticas, que a menudo demoran meses o años.

Para que estas cumplan con tales características, algunos pasos de las revisiones sistemáticas se omiten o simplifican.

Los estudios que evaluaron el impacto de las adaptaciones metodológicas en las revisiones rápidas en comparación con las revisiones sistemáticas encontraron que los resultados de ambos tipos de revisiones eran notablemente similares.

2.11. Mentoría en software para emprendimientos: Trabajos vinculados

La aplicación de mentoría en ingeniería de software tiene diversos antecedentes, tanto a nivel de enseñanza como en la práctica industrial.

A nivel de la práctica industrial de la ingeniería de software, Winters y colegas señalan el uso de mentorías en la compañía Google [29]. Una de las áreas en las que se aplica mentoría es la de revisiones y legibilidad de código. Según mencionan estos autores, en Google, la legibilidad se refiere a algo más que la legibilidad del código en sí. Es un proceso de mentoría estandarizado en toda la organización para difundir las mejores prácticas del lenguaje de programación. La legibilidad cubre una amplia gama de conocimientos, incluidos, entre otros, expresiones idiomáticas, estructura de código, diseño de API, uso adecuado de bibliotecas comunes, documentación y cobertura de pruebas.

Otra área en la que se ha estudiado el uso de mentorías es la relativa al diseño de software. Así, por ejemplo, Hatley, Al-Freih y Bannan [30] reportan los resultados de un estudio en una empresa de ingeniería de software sobre las interacciones y comportamientos que ocurren durante las reuniones de diseño donde la mentoría y el diseño ocurren simultáneamente durante una parte de la fase de diseño de un proyecto de software. Para estos autores, el uso de mentoría en estos casos se muestra especialmente importante para apoyar transiciones más fluidas desde la capacitación hasta convertirse en ingeniero de software (diseño, código, prueba) y ser contratado con el objetivo de aplicar las habilidades de ingeniería de software aprendidas.

3. Problema de investigación

En el entorno de los emprendimientos de software, son múltiples los desafíos que enfrentan sus fundadores en relación con el desarrollo de su producto o servicio. Estos desafíos incluyen definir y validar las funcionalidades adecuadas del producto, para satisfacer a su segmento de mercado objetivo, que quizás aún no se ha identificado claramente, tomar y justificar decisiones técnicas fundamentales, a pesar de la extrema incertidumbre, crear prototipos y evaluar rápidamente nuevas ideas y características, al tiempo que construyen productos mínimos viables, priorizar el esfuerzo de ingeniería en un ambiente de escasez de recursos humanos y materiales, además de comunicarse de manera efectiva tanto dentro del equipo como con partes interesadas externas no técnicas.

Por otra parte, existen reportes que muestran una tasa de fracaso de los emprendimientos de software en general en torno al 90% [1] o que dos tercios de los mismos nunca generan dividendos positivos para sus inversores [2]. Aunque hay muchos factores que pueden llevar al fracaso de este tipo de emprendimiento, deficiencias en la aplicación de ingeniería de software se señalan como una razón clave, tal como se menciona en [3] y [4]. Estudios anteriores han mostrado que los socios fundadores de las mismas utilizan en la práctica un subconjunto muy acotado de técnicas de ingeniería de software, y que suelen tener dificultades en seleccionar las que son más adecuadas para su entorno de emprendimiento y que tienen dificultades en aplicarlas correctamente [5], [6].

Por otra parte, Richter y Schildhauer [31] sostienen que las empresas emergentes necesitan apoyo durante su etapa inicial en al menos las siguientes áreas: dar forma a la comprensión de sus propias fortalezas y debilidades, apoyo con respecto a los cursos de acción, e interacción con mentores y expertos de diferentes campos. Sin embargo, es importante destacar que si bien las incubadoras y aceleradoras ofrecen apoyo, asesoramiento y mentoría en múltiples áreas, su enfoque está predominantemente orientado hacia aspectos de negocio. Consecuentemente, existe una notable brecha en cuanto al soporte específico en diseño y construcción de soluciones basadas en software. Esta carencia representa un problema significativo, ya que deja a los emprendimientos de software sin una guía adecuada en una de las áreas cruciales para su éxito: la ingeniería de software.

4. Diseño metodológico

4.1. Objetivos

4.1.1. Objetivos Generales

El propósito general de este proyecto es diseñar, implementar y evaluar un conjunto de guías al estilo de "cómo hacer para…" (*how-to*) destinadas a orientar y mejorar prácticas en los emprendimientos de software, complementando e incorporando lo anterior con un modelo de mentoría en ingeniería de software donde las guías sean utilizadas como insumo.

4.1.2. Objetivos Específicos

Los objetivos específicos para este trabajo son:

1. Identificar cuáles son las técnicas de ingeniería de software usualmente utilizadas en los emprendimientos de software para abordar los diferentes aspectos de la ingeniería de su producto o servicio basado en software.

2. Diseñar y elaborar un conjunto de guías que orienten la selección e implementación de técnicas de ingeniería de software adecuadas para emprendimientos de software. Adicionalmente, es vital establecer y aplicar un mecanismo de retroalimentación para medir la satisfacción de los usuarios al interactuar con estas guías, lo que incluye encuestas y entrevistas.

3. Complementar estas guías con un plan de mentoría a demanda y validar empíricamente la eficacia y relevancia de las guías propuestas, así como del plan en sí. Para esta validación, se considerará la opinión de personas idóneas en la materia, pudiendo ser estos docentes de laboratorio de ingeniería de software, egresados que han llevado a cabo proyectos de investigación en emprendimientos de software, y emprendedores, todos ellos vinculados a la Universidad ORT Uruguay.

4.2. Preguntas de investigación

Considerando los objetivos definidos anteriormente, las preguntas planteadas para esta investigación son las siguientes:

1. ¿Cuáles son las técnicas de ingeniería de software más utilizadas en emprendimientos de software?
2. ¿Qué estructura deberían tener las guías de apoyo sobre las técnicas de ingeniería de software más utilizadas?
3. ¿Qué nivel de satisfacción reportan quienes interactuaron con las guías, respecto al contenido generado?
4. ¿Cómo se puede definir y especificar un proceso de mentoría sobre técnicas de ingeniería de software orientado a emprendimientos de software?

4.3. Fase 1: Identificación de técnicas

4.3.1. Preparación y ordenamiento de los artículos

Partimos de la base de que nuestro tutor Gerardo Maturro a lo largo de los años ha recopilado papers sobre emprendimientos de software que han sido publicados en conferencias y revistas arbitradas. Aprovechamos este compendio ya existente principalmente debido a las limitaciones de tiempo asociadas a nuestro proyecto. El tiempo destinado para la realización de este trabajo no hubiera permitido una recopilación desde cero de tantos documentos especializados. La utilización de estos documentos nos permitió centrarnos en aspectos más críticos de nuestra investigación, sin la necesidad de destinar recursos significativos a la fase inicial de búsqueda y acumulación de artículos.

4.3.2. Obtención de técnicas de ingeniería de software

El objetivo en esta sección es analizar cada uno de los papers, iniciando del más antiguo al más nuevo, utilizando la técnica de revisiones rápidas (*rapid reviews*) tal como fue presentada por B. Cartaxo, G. Pinto, y S. Soares. A través de este enfoque, nos centraremos

en identificar y recopilar técnicas de ingeniería de software que hayan sido aplicadas por los emprendimientos de software.

Hemos optado por que cada paper sea leído inicialmente por uno de nosotros debido a razones prácticas y de eficiencia. Esta decisión nos permite cubrir una mayor cantidad de literatura en menor tiempo y con ello maximizar la amplitud de nuestra investigación. Sin embargo, reconocemos que algunos papers pueden ser particularmente ricos en contenido y, en estos casos, es beneficioso obtener la perspectiva de ambos investigadores. Es por esto que al detectar un artículo de especial relevancia o que pueda beneficiarse de una interpretación adicional, se marcará para una revisión cruzada. Este enfoque equilibra la eficiencia con la profundidad y asegura una comprensión completa de los temas tratados.

4.3.3. Validación de completitud y exactitud

En este paso cada uno de los integrantes debe iterar a través de la información recolectada de cada uno de los papers que analizó el otro compañero, y evaluará si las formas en que se hace mención a las técnicas de ingeniería de software son suficientemente explícitas. En caso de que encontrar posibilidades de mejora, se deberá dejar la correspondiente constancia para que el revisor original la tenga en consideración, y luego el equipo en conjunto evalúe la correctitud de la sugerencia.

4.3.4. Revisión cruzada de artículos académicos

Para los artículos académicos marcados durante los procesos anteriores para revisión cruzada, el integrante que no leyó en primera instancia al mismo deberá realizar el análisis adecuado para poder obtener más y mejor información.

Es importante recalcar que la nueva lectura y análisis debe de realizarse sin tener en cuenta lo recabado por el otro integrante inicialmente, es decir, el paper debe ser tratado como si no hubiese sido analizado previamente. En este caso no se aplicó la técnica de revisiones rápidas debido a que dicho artículo fue marcado como relevante, por esto decidimos analizarlo íntegramente de manera completa.

Para cada paper sujeto a revisión cruzada, deberá registrar el resultado de su análisis. Se pretende realizar revisión cruzada a todos los artículos que durante la revisión inicial sean considerados de alta relevancia de acuerdo con la temática que nos concierne.

4.3.5. Agrupamiento de técnicas de ingeniería de software

En este paso se llevará a cabo un proceso meticuloso para categorizar las diversas técnicas de ingeniería de software identificadas en los papers. Esto implica tomar decisiones estratégicas sobre cómo agrupar técnicas con diferentes denominaciones o que puedan encajar dentro de una categoría más general. Para asegurar una categorización precisa y comprehensiva, se realizarán múltiples iteraciones a través de todas las técnicas recolectadas en las etapas anteriores, refinando en cada iteración el agrupamiento hasta llegar a un punto donde estemos satisfechos con el resultado.

4.3.6. Obtención de cantidad de ocurrencias de técnicas de ingeniería de software

La cantidad de ocurrencias de estas técnicas de ingeniería de software obtenidas en los pasos anteriores será un punto fundamental (aunque no único) a la hora de seleccionar dichas técnicas para la creación de las guías, por lo tanto, es de interés obtener el valor numérico de ocurrencia para cada una ellas.

4.3.7. Definición de parámetros para la selección de las técnicas

Siguiendo la línea del punto anterior, la cantidad de ocurrencias de una técnica de ingeniería de software en los más de 300 papers es un parámetro importante para la selección de las mismas, que serán el insumo para la realización de las guías, pero no único. Es de nuestra importancia también tener un conjunto de guías que sean heterogéneas en cuanto a las áreas de conocimiento del libro SWEBOK [12], para que de esta forma sean lo más abarcativas posibles en cuanto a sus áreas de conocimiento. Del mismo modo, nos interesa que exista

también una heterogeneidad en cuanto al ciclo BML (*Build-Measure-Learn*), de modo de tener también guías que sean tanto para construir, para medir, y para aprender.

4.3.8. Selección de las técnicas de ingeniería de software para las guías de apoyo

Luego de aplicar todos los pasos anteriores, se estará en condiciones para poder seleccionar las técnicas de ingeniería de software, para así desarrollar las guías de apoyo, que se diseñarán y crearán en la próxima fase.

4.4. Fase 2: Diseño de guías de apoyo y del proceso de mentoría

4.4.1. Diseño de guías de apoyo

Previamente a enfocarnos en el contenido de cada una de las guías, se necesitará definir una estructura en común para todas estas guías de apoyo. Definiendo una estructura se obtienen ventajas importantes, como por ejemplo facilitar la usabilidad y búsqueda de información, organizar la información de manera clara, mejorar la comprensión, simplificar el mantenimiento y actualización, y también facilitar la creación de nuevas guías de apoyo en el futuro.

Si bien partimos de una estructura común, cada guía puede tener diferentes particularidades dependiendo de la naturaleza de la misma. Esto puede implicar la necesidad de agregar secciones específicas para adaptarse a dichas particularidades, de este modo mejorando la forma en que los procesos definidos son presentados.

Una vez definida la estructura, lo que queda es obtener información de cada una de las técnicas de ingeniería de software obtenidas como resultado de la fase 1, y crear las guías de apoyo. La información a utilizar debe de ser obtenida de fuentes con respaldo académico y se debe hacer referencia a las mismas.

4.4.2. Diseño de plan de mentoría

Diseñar un plan de mentoría a demanda para complementar e incorporar estas guías de apoyo, con el fin de ayudar a la comprensión de las mismas a los emprendimientos de software, evaluando y mejorando iterativamente tanto este plan como a las guías.

4.5. Fase 3: Validación e implementación de guías y plan de mentoría

4.5.1. Búsqueda y selección de personas para validación de guías

Se necesita encontrar un grupo heterogéneo de personas idóneas para poder tener una validación inicial de las guías. Para eso se deberá contactar a docentes del laboratorio de ingeniería de software, egresados que hayan realizado proyectos de investigación y emprendedores, todos vinculados a Universidad ORT Uruguay. De ahora en más, a este grupo de personas las denominaremos “expertos”.

4.5.2. Agrupamiento de expertos en grupos heterogéneos

Una vez determinadas estas personas idóneas en ingeniería de software que realizarán la validación de las guías, hay que generar grupos con el objetivo de que cada uno de ellos evalúe sólo un subconjunto de las mismas, para que se les haga más amena su evaluación. La idea principal es que dichos grupos sean lo más heterogéneos posibles, es decir, por ejemplo, que los docentes no evalúen las mismas guías, sino que cada grupo de guías tenga diferentes perspectivas de diferentes tipos de expertos en la materia.

4.5.3. Diseño y distribución de encuestas

Para obtener una evaluación inicial y detallada de los expertos en cada uno de los grupos mencionados en la sección anterior, optamos por emplear un enfoque de dos etapas: encuestas y entrevistas individuales. Las encuestas nos permitirán recopilar datos

cuantitativos y obtener una visión general rápida de las opiniones y experiencias de los expertos. Esta información será esencial para identificar patrones y tendencias. Posteriormente, las entrevistas individuales servirán para profundizar en los resultados de la encuesta, permitiéndonos recopilar datos cualitativos y obtener matices más específicos sobre cada tema. La combinación de estos dos métodos proporciona un enfoque más integral y robusto para entender las perspectivas de los expertos involucrados.

Para la realización de las encuestas es necesario considerar la posibilidad de incluir preguntas tanto abiertas como cerradas. Las preguntas abiertas ofrecen a las partes interesadas la oportunidad de expresar sus opiniones y puntos de vista con sus propias palabras, lo cual puede ayudar a identificar problemas que nosotros tal vez no hayamos visto. Las preguntas cerradas pueden darnos datos numéricos que luego pueden analizarse estadísticamente, y así ver patrones y tendencias que no son tan evidentes. En el caso de este último tipo de preguntas, determinar un sistema para poder medir las respuestas de la mejor manera posible.

Por último, y no por eso menos importante, tener en cuenta durante el diseño de la encuesta de definir objetivos claros, de estructurar la encuesta de manera lógica, comenzando en temas más generales y luego ir hacia temas más específicos, de redactar preguntas claras y concisas, sin ambigüedades, y de intentar que no sea tan larga, ya que los encuestados pueden perder el interés haciendo que sus respuestas no sean tan acertadas.

Una vez diseñada la encuesta, distribuirla y esperar a que los expertos las realicen.

4.5.4. Análisis de las encuestas a expertos

Esta etapa no es solamente importante para obtener una primera validación, sino, como nombramos anteriormente, poder generar preguntas que nos ayuden a ahondar aún más en particularidades que consideremos pertinentes.

En líneas generales, debemos utilizar la información obtenida para detectar mejoras para las guías y para ahondar en detalles en la próxima etapa, que nos ayudará a descubrir más mejoras.

4.5.5. Diseño de entrevistas a expertos

Las entrevistas son esenciales para validar nuestras guías de apoyo, ya que permiten obtener retroalimentación directa de las partes interesadas, asegurando que las pautas sean claras, efectivas y aborden nuestras necesidades específicas de manera precisa.

Para poder llevarlas a cabo, es vital que las entrevistas estén preparadas de antemano, definiendo el propósito y la información que se desea obtener. Es también muy importante lograr que las preguntas sean abiertas de forma tal que fomenten la discusión y se profundice en los temas.

Al inicio de esta, solicitar permiso para poder grabar la entrevista para luego analizarla con aún más detalle.

4.5.6. Análisis de las entrevistas a expertos

Tras la realización de las entrevistas individuales, las cuales se llevarán a cabo siguiendo los criterios establecidos en la sección anterior, procederemos a analizar los datos recopilados. Este análisis tiene como objetivo profundizar en las perspectivas y opiniones de los expertos consultados.

4.5.7. Búsqueda de equipos de proyectos

El objetivo en esta instancia es contactarnos con los tutores que están actualmente llevando adelante proyectos del tipo emprendimiento dentro de Universidad ORT Uruguay, comentarles de nuestro trabajo y presentarles tanto el plan de mentoría como las guías, y en caso de que estar de acuerdo, tener reuniones con sus grupos de estudiantes con el afán de que puedan a llevar a cabo este plan, utilizando las técnicas de ingeniería de software que entendamos que cada uno de los equipos necesiten.

4.5.8. Inicio y ejecución del plan de mentoría

Para los equipos que hayan decidido adoptar el plan de mentoría y seguir sus lineamientos definidos, se procederá con su ejecución. Luego, se analizarán los resultados obtenidos, permitiendo extraer conclusiones y asegurando el enriquecimiento del equipo durante el plan.

5. Resultados obtenidos

5.1. Reporte de resultados de fase 1: Identificación y clasificación de prácticas

Para cada uno de los puntos definidos en la sección anterior referidos a la fase de identificación y clasificación de técnicas, denotaremos los resultados intermedios obtenidos hasta llegar al resultado final, que corresponde a las técnicas de ingeniería de software que utilizaremos para la creación de las guías de apoyo.

5.1.1. Preparación de los artículos académicos

Inicialmente lo que hicimos fue ordenar los artículos académicos tanto de manera cronológica ascendente como alfabéticamente, y generamos una hoja de cálculo de forma de facilitar y registrar los descubrimientos y análisis que vayamos a hacer.

En total, hemos obtenido 315 papers, siendo el más antiguo publicado en el año 2000 y el más reciente en 2023.

Esta hoja de cálculo consta de cuatro columnas, siendo ellas, de izquierda a derecha las siguientes:

- **Título**: Es la concatenación del año de publicación del paper, y del título de este.
- **Responsable**: Corresponde al integrante del equipo que tome la responsabilidad inicial de analizar dicho paper.
- **Información recolectada**: Conjunto de técnicas de ingeniería de software que han sido utilizadas por emprendimientos de software, recopiladas luego de la lectura y análisis del paper. Si no se encuentran dichas técnicas, dejar constancia de que esto es así.
- **Observaciones**: Columna dedicada para anotar cualquier observación que sea importante remarcar para las posteriores etapas del proceso.

5.1.2. Análisis de técnicas de software en artículos académicos

En este punto hemos obtenido las técnicas de ingeniería de software que según estos emprendimientos de software agregaron valor a sus procesos. Para lograr esto, cada uno de los integrantes del equipo tomó responsabilidad sobre cada paper, respetando el orden dispuesto en la etapa anterior, y al finalizar el análisis de cada uno de manera individual, se registraron tanto las técnicas de ingeniería de software obtenidas, como si dicho artículo es válido para una posterior revisión cruzada. Dicha validez corresponde a si un artículo académico es rico en contenido en cuanto a lo que nos concierne. Si bien en el relevamiento inicial contabamos con 315 papers, finalmente no de todos ellos pudimos obtener estas técnicas. En el Anexo 20 se puede apreciar una tabla en la cual figuran todos los papers de los cuales pudimos obtener estas técnicas de ingeniería de software.

Al abordar cada paper, comenzamos con un escaneo inicial del título y abstract, para discernir rápidamente su relevancia. Si estos componentes sugerían que estaba entre nuestra temática objetivo, profundizamos inmediatamente en la introducción y conclusión del artículo, ya que proporcionan una clara perspectiva del propósito, metodología y resultados del estudio. Para aquellos documentos de especial relevancia en cuanto a su contenido, realizamos una revisión selectiva de secciones clave, como la metodología o los resultados, priorizando áreas que podrían ofrecer información significativa sin requerir una lectura completa del contenido.

5.1.3. Validación de información recolectada en exactitud y completitud

Como insumo de esta etapa se siguió utilizando la hoja de cálculo de la etapa anterior, a la cual se le agregó una nueva columna llamada “validar”, en la cual se dejaba constancia de que alguno de los resultados extraídos tiene posibilidades de mejora en cuanto a la completitud.

Cada uno de los integrantes revisó la celda correspondiente a “Información Recolectada” de todos los papers que fueron analizados en profundidad por su compañero. En caso de encontrar alguna observación, se dejaba explícita en la columna “validar”.

El integrante del equipo que haya analizado el paper en primera instancia, será también responsable de responder la solicitud de validación, y realizar las modificaciones que considere pertinentes, si correspondiese.

Al finalizar esta etapa, tenemos una versión más refinada de la hoja de cálculo en la cual están todos los papers revisados por alguno de los dos integrantes, y además la información obtenida está validada en cuanto a su que las técnicas registradas están registradas de forma correcta.

5.1.4. Revisión cruzada de artículos académicos

Hemos realizado una revisión cruzada focalizada en 33 papers, elegidos específicamente por su alta relevancia dentro del total de artículos.

En esta etapa, conforme a lo planificado y a lo estipulado en el diseño metodológico, ambos integrantes realizamos revisiones mutuas de los papers que se habían seleccionado para su análisis.

Se generó una nueva columna en la hoja de cálculo llamada “Revisión Cruzada” en la cual se agregan los resultados de este nuevo análisis, o simplemente para manifestar que se está de acuerdo con lo determinado en primera instancia.

Luego se realizó una instancia en la cual recorrimos cada uno de los papers sujetos a este tipo de revisión, y en conjunto se discutió si las observaciones planteadas tenían relevancia, o no, y en caso de que si, actualizar la información recolectada para cada paper.

Al finalizar esta etapa, en la hoja de cálculo contamos con todos los papers analizados, cumpliendo lo planificado en cuanto al alcance de la revisión cruzada, y además discutido y resuelto en conjunto las observaciones encontradas, las cuales fueron mínimas.

5.1.5. Agrupamiento de técnicas de ingeniería de software

Teniendo como insumo las técnicas obtenidas como resultado de los pasos anteriores, con todos los análisis pertinentes, determinamos un procedimiento para agrupar todas las técnicas de ingeniería de software recolectadas, con el objetivo de poder en la próxima etapa cuantificar estas técnicas de la manera más fielmente posible.

Cómo nombrado en la correspondiente etapa del diseño metodológico, lo que buscamos es encontrar términos en común para técnicas que posiblemente estén escritas de distinta manera, o que puedan ser generalizadas en una técnica o actividad que las integre.

Ambos integrantes del equipo trabajamos en conjunto iterando por cada una de estas técnicas en múltiples oportunidades, analizando si cada una de las mismas estaba escrita de manera distinta, aunque representando la misma técnica, o si pudiera agruparse en una técnica más general.

Las técnicas de ingeniería de software que podían agruparse en una técnica más general fueron agrupadas, mientras que técnicas que significaban exactamente lo mismo indistintamente del idioma o de la forma en la cual fueron escritas, fueron normalizadas. Esto puede verse en el Anexo 19.

5.1.6. Contabilidad de técnicas de ingeniería de software obtenidas

En esta etapa contamos con una primera versión de un documento en el cual se encuentran todas las técnicas de ingeniería de software obtenidas, con el mejor nivel de normalización posible al aplicar el proceso de la etapa anterior de forma manual.

Para llevar a cabo esta etapa, desarrollamos un script de Python el cual nos permitió contar las ocurrencias de estas técnicas, y ordenarlas de manera descendente en cuanto a este atributo, ya que estas son uno de los aspectos a tomar en cuenta para poder finalmente seleccionar las técnicas de ingeniería de software. Este script está disponible en el Anexo 13.

Con la ejecución de este script, pudimos comprobar que la normalización realizada en el proceso anterior era mejorable, ya que encontramos en algunos casos, técnicas que cumplían los requisitos para ser normalizadas. Por tanto, decidimos normalizar estas técnicas conforme a lo definido en el paso anterior, y repetir la ejecución del script. Este script lo ejecutamos tres veces hasta quedar conformes con el resultado.

De esta forma, nos quedamos con un listado de técnicas con su respectiva cantidad de ocurrencias.

5.1.7. Definición de parámetros para la selección final de las técnicas

En esta etapa tenemos como insumo la lista con todas las técnicas de ingeniería de software con sus respectivas ocurrencias luego del análisis de los papers.

Usamos como herramienta una nueva hoja de cálculo con las siguientes columnas:

- **Técnica de ingeniería de software:** técnicas ordenadas de manera descendente por número de ocurrencias totales.
- **BML**: Fases del ciclo Build-Measure-Learn.
- **Área de conocimiento SWEBOK**

Luego de haber realizado el tratamiento de las técnicas como se evidenció en los pasos anteriores, ambos integrantes del equipo tuvimos una sesión de trabajo en conjunto para definir, para cada una de estas técnicas de ingeniería de software, a qué fase del ciclo BML y a que área de conocimiento de SWEBOK [12] pertenecen.

Observamos que algunas técnicas altamente valoradas son demasiado extensas y complejas, o muy generales como para condensar en una guía de apoyo enfocada y concreta. Por esta razón, hemos decidido priorizar y promover aquellas técnicas que sí pueden ser incorporadas en una guía de apoyo y aportar valor significativo a un emprendimiento de software, acorde a la dimensión y plazos de este proyecto.

5.1.8. Selección de las técnicas de ingeniería de software para las guías

Al realizar el proceso declarado en las etapas anteriores de esta fase pudimos determinar 12 técnicas de ingeniería de software para poder generar las respectivas guías de apoyo. Estas técnicas son:

- Encuestas
- Entrevistas
- Estudio de casos
- Integración y Despliegue Continuo
- Manejo de deuda técnica
- Mockups
- Observación
- Pruebas A/B
- Prueba de usabilidad
- Prototipos
- Pruebas Unitarias
- Wireframes

Para llegar a este listado tratamos de abordar la mayor cantidad de áreas del conocimiento de SWEBOK [12] y teniendo en cuenta la metodología Lean Startup, específicamente su enfoque de BML (*Build-Measure-Learn*), luego de haber priorizado por número de ocurrencias, siendo este último un factor preponderante.

Pruebas A/B:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de pruebas de software y dentro de BML a la actividad de *Measure*.

Motivación:

Las pruebas A/B, también conocidas como pruebas de división, son una técnica utilizada para evaluar dos versiones diferentes de una página web o una aplicación móvil para determinar cuál de ellas produce mejores resultados en términos de objetivos específicos.

Creemos que es de gran utilidad a la hora de tener que tomar decisiones en emprendimientos de software, ya que seguramente no se tenga la suficiente experiencia para elegir que camino tomar ante una situación puntual. Fue la técnica más frecuente en la revisión de artículos en cuanto a la actividad de pruebas de software.

Prueba de usabilidad:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de pruebas de software y dentro de BML a la actividad de *Measure*.

Motivación:

Las pruebas de usabilidad son una técnica importante en la ingeniería de software que ayuda a mejorar la calidad y la eficiencia del diseño de la interfaz de usuario, lo que puede mejorar la satisfacción del usuario y reducir los costos de desarrollo. Los principales beneficios de incorporar esta actividad son identificar problemas de usabilidad, asegurar la eficacia del diseño, reducir los costos, proporcionar información valiosa para la toma de decisiones y mejorar la satisfacción del usuario.

Fue una de las técnicas más frecuentes en la revisión de artículos en cuanto a la actividad de pruebas de software.

Pruebas Unitarias:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de pruebas de software y de construcción de software, mientras que dentro de BML dentro de la actividad de *Measure*.

Motivación:

Las pruebas unitarias son una técnica de pruebas automatizadas utilizadas en el desarrollo de software, y su principal objetivo es verificar que cada unidad individual de código (generalmente una función o método) funcione correctamente de forma aislada, es decir, sin depender de otras unidades de código o de otras partes del sistema.

Creemos que es de gran utilidad comprender este tipo de pruebas para un emprendimiento de software. siendo sus principales ventajas en su utilización el ayudar a detectar errores en el código de forma temprana, mejorar la calidad del software, facilitar el mantenimiento del código, reducir los costos de desarrollo y permitir la integración continua.

Fue una de las técnicas más frecuentes en la revisión de artículos en cuanto a la actividad de pruebas de software

Estudio de casos:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de requisitos de software y dentro de BML a la actividad de *Learn* principalmente.

Motivación:

La técnica de estudio de casos en ingeniería de software ayuda a los emprendimientos de software a identificar problemas, validar requisitos e identificar mejores prácticas. Esto les permite desarrollar un software que se ajuste a los estándares de la industria, satisfaga las necesidades de sus clientes y les permita mantenerse competitivos.

Además de esto vimos que en la revisión fue muy utilizada por este tipo de emprendimientos.

Prototipo:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de requisitos de software y diseño de software principalmente, y dentro de BML de *Build*.

Motivación:

La técnica de prototipado ayuda a los emprendimientos de software a crear rápidamente versiones preliminares de su software y obtener retroalimentación temprana de los usuarios. Esto les permite definir los requisitos y especificaciones del software, identificar problemas tempranamente, iterar rápidamente en el desarrollo, y atraer tanto a inversores como a clientes potenciales. En resumen, es una técnica muy útil para ahorrar tiempo y dinero en el desarrollo y en mejorar la calidad del software.

Aparece como técnica ampliamente usada en la revisión que realizamos.

Mockups:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de requisitos de software y diseño de software principalmente, y dentro de BML de *Build*.

Motivación:

Los mockups son representaciones visuales estáticas o interactivas de una interfaz de usuario o diseño gráfico, que se utilizan para ilustrar cómo se verá y funcionará un producto o servicio antes de que se construya o desarrolle completamente.

Creemos que es una herramienta que les será de mucha utilidad a nuestros interesados a la hora de probar conceptos, identificar problemas, además de ahorrar tiempo y recursos.

También esta técnica tuvo varias ocurrencias en la revisión, es ampliamente utilizada por los emprendimientos de software.

WireFrames:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de requisitos de software y diseño de software principalmente, y dentro de BML de *Build*.

Motivación:

Los wireframes son representaciones visuales esquemáticas de una interfaz de usuario o diseño gráfico que se utilizan para ilustrar la estructura, disposición y flujo de información de un producto o servicio.

Fue encontrada en en la revisión, siendo utilizada por varios emprendimientos de software, y creemos que resultará útil a la hora de planificar y diseñar la estructura, así como también la disposición de la interfaz de usuario.

Más allá de lo anterior tiene una gran potencialidad en cuanto a que permiten una colaboración más eficaz entre los miembros del equipo y los *stakeholders*, al proporcionar una representación visual clara y concisa de la estructura y flujo de información de un producto o servicio.

Entrevistas:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de requisitos de software y dentro de BML a la actividad de *Measure* principalmente.

Motivación:

Las entrevistas en ingeniería de software son una técnica utilizada para recolectar información y obtener perspectivas, opiniones y requisitos de los usuarios, clientes o expertos en un proyecto de software. La información recopilada en las entrevistas es importante para mejorar la calidad del software y desarrollar soluciones efectivas que se adapten a las necesidades del negocio.

Encuestas:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de requisitos de software y dentro de BML a la actividad de *Measure* principalmente.

Motivación:

Las encuestas en los emprendimientos de software son una herramienta de investigación que se utiliza para recopilar información estructurada de un grupo de personas. Las encuestas pueden ser utilizadas para comprender mejor las necesidades y preferencias de los clientes, medir la satisfacción de estos, realizar investigaciones de mercado y obtener información sobre la competencia. Al obtener retroalimentación directa de los usuarios, se puede mejorar la calidad y la satisfacción del cliente, adaptar su producto o servicio para satisfacer mejor las necesidades del mercado, y desarrollar estrategias de *marketing* y ventas más efectivas. Es importante diseñar adecuadamente las encuestas para garantizar la validez y la fiabilidad de los datos recopilados, y analizar cuidadosamente los resultados para tomar decisiones informadas.

Observación:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de requisitos de software y dentro de BML, a la actividad de *Measure* principalmente.

Motivación:

La técnica de observación en ingeniería de software puede ser de gran utilidad para los emprendimientos de software, porque les permite comprender mejor las necesidades y desafíos de los usuarios desde una etapa temprana del desarrollo del producto. Esto puede ayudar a diseñar soluciones de software que satisfagan mejor las necesidades de los usuarios, lo que a su vez puede mejorar la satisfacción del cliente y aumentar las posibilidades de éxito del producto. Nos parece adecuado incluirla ya que en la revisión encontramos que es una técnica utilizada y otro aspecto importante es que la observación es una técnica de bajo costo, efectiva para obtener información

valiosa sobre el comportamiento de los usuarios en el ambiente en el que se utiliza el software y los procesos involucrados.

Manejo de deuda técnica:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de mantenimiento de software y calidad de software, y dentro de BML a la actividad de *Measure* principalmente.

Motivación:

El manejo de la deuda técnica es importante para los emprendimientos de software porque les ayuda a reducir costos, aumentar la eficiencia en el desarrollo de su producto de software, y mantenerse ágiles y flexibles. Además, la gestión de la deuda técnica puede mejorar la calidad del software, reducir el tiempo de inactividad y aumentar las posibilidades de éxito del producto.

Si bien tenemos claro que en este tipo de emprendimientos sus comienzos no suelen darle mucha importancia a este aspecto, consideramos que puede ser muy útil entender las implicancias de no atenderla y los beneficios que pueden tener al implementar un manejo de deuda técnica.

En la revisión encontramos que se marca con énfasis la importancia de atender este aspecto.

Integración y Despliegue Continuo:

Dentro de SWEBOK está comprendida dentro del área de conocimiento de mantenimiento de software, proceso de ingeniería de software, gestión de la configuración de software, herramientas y métodos de la ingeniería de software, mientras que dentro de BML a la actividad de *Build* principalmente.

Motivación:

La metodología de CI/CD es una práctica muy beneficiosa para los emprendimientos de software, ya que les permite desarrollar y entregar software de manera más rápida, eficiente y escalable. Al utilizar CI/CD, los desarrolladores pueden automatizar los procesos de prueba, integración y entrega de software, lo que les permite centrarse en escribir código de alta calidad en lugar de realizar tareas repetitivas y manuales.

5.2. Reporte de resultados del diseño de guías de apoyo y del plan de mentoría

En esta etapa se logró tener definida una estructura estandarizada para todas las guías, así como la realización de estas y también la especificación y creación del plan de mentoría.

5.2.1. Diseño de guías de apoyo

Para este proceso nos basamos principalmente en nuestras experiencias como estudiantes, complementado con nuestras vivencias en el plano laboral, para determinar cuáles son las secciones que mayormente aportan valor.

A continuación, detallamos cada una de las secciones junto con una descripción de las razones para incluirlas en nuestras guías:

- **Introducción**: Esta ofrece una panorámica inicial, sentando las bases para las secciones posteriores. Asimismo, puede ayudar a captar la atención y el interés del lector desde el inicio.

- **¿Por qué debería incluirla en mi emprendimiento de software?:** Estos emprendimientos operan con recursos limitados y a menudo deben priorizar sus acciones. Esta sección responde a la pregunta fundamental: "¿Por qué debería invertir tiempo, esfuerzo y recursos en esto?". La idea es mostrar el valor que les puede aportar a sus procesos aplicar esta técnica.

- **¿En qué condiciones puedo aplicar esta técnica?:** Define el contexto o las circunstancias ideales para aplicar la técnica. Sirve para ayudar al lector a identificar si su situación particular se alinea con la aplicación efectiva de lo que se está presentando.

- **Proceso de desarrollo**: Es fundamental brindar un camino claro y estructurado para que el lector pueda poner en práctica lo que se describe en la guía. Esta sección garantiza que el lector tenga una hoja de ruta paso a paso para implementar de manera efectiva la técnica.

- **Consideraciones para aplicar la técnica**: Mientras que el proceso de desarrollo ofrece los pasos a seguir, las consideraciones ofrecen una guía de qué no hacer o aspectos a tener en cuenta durante la implementación. Los emprendimientos de software, al ser entidades en crecimiento y aprendizaje, se beneficiarán enormemente de estas consideraciones, evitando errores comunes y garantizando una aplicación más eficiente de la técnica.

- **Herramientas Sugeridas**: Esta sección es opcional y sugiere algunas herramientas que ayuden a la aplicación de la técnica en contextos particulares.

- **Dónde leer más**: Proporcionar fuentes y referencias es crucial para legitimar la información presentada y ofrecer a los emprendimientos de software la oportunidad de profundizar en temas específicos, lo que puede fortalecer la confianza del lector en el contenido.

Haciendo uso tanto de libros como de artículos académicos, con la intención de abordar las temáticas con el mayor respaldo académico posible, elaboramos en conjunto guías para todas las técnicas obtenidas como resultado del proceso de análisis de artículos académicos.

Para poder ver en más detalle cada una de estas guías, ver los anexos correspondientes a las mismas que se encuentran desde el Anexo 1 hasta el Anexo 12.

5.2.2. Diseño del plan de mentoría

5.2.2.1. Diseño de la estructura del plan de mentoría

La creación y diseño de este plan de mentoría surge de la necesidad de acompañar al equipo de trabajo en la utilización de las guías de apoyo, facilitando así el entendimiento como la aplicación de estas. Nos inspiramos en el enfoque delineado por Linda Philips-Jones [32] y enriquecimos esta propuesta con nuestra experiencia académica y laboral, haciendo el ejercicio de colocarnos en el lugar del equipo de trabajo para entender mejor qué necesidades pueden tener.

Aspiramos a que este plan, más allá de ser una simple guía, se convierta en una herramienta práctica que favorezca el fortalecimiento de los equipos y los ayude a alcanzar sus objetivos.

La estructura que hemos diseñado para el plan de mentoría consiste en las siguientes secciones:

- **Introducción**: Esta sección es importante para establecer el contexto y la relevancia del plan. Antes de entrar en detalles técnicos o pasos específicos, es crucial para los lectores entender la razón de ser del plan y cómo se sitúa en el enfoque general de la ingeniería de software.

- **Roles**: Definir roles claros desde el principio evita confusiones y malentendidos. Define responsabilidades y expectativas para cada una de las partes involucradas. Al tener un entendimiento claro de su rol, cada parte sabe qué se espera de ella y cómo debe interactuar con las demás.

- **Objetivos**: En esta sección se deja en claro la finalidad del proceso definido para todas las partes involucradas durante la aplicación del plan de mentoría.

- **Plan de trabajo:** Esta sección detalla la ejecución del proceso paso a paso. Sin un plan estructurado, hay riesgos de omisiones, confusiones o repeticiones innecesarias. Aquí, cada actividad, desde la introducción hasta la evaluación final, se planifica y organiza, garantizando un proceso fluido y eficiente. Esta sección está a su vez compuesta por las siguientes subsecciones:

- **Reunión 1 - Introducción y evaluación:** Esta reunión sienta las bases para todo el plan de mentoría. Permite a las partes conocerse, identificar las necesidades específicas del equipo y establecer objetivos claros para la mentoría. Además, al determinar la logística para futuras reuniones, se espera lograr un flujo continuo en cuanto a la comunicación.

- **Identificación de técnicas:** Luego de haber completado la reunión anterior, el mentor analizara la situación particular para recomendar qué técnicas de las disponibles podrían ajustarse a sus necesidades, además de tener las opiniones o preferencias expresadas por el equipo en cuanto al uso de las guías disponibles.

- **Reunión 2 - Introducción a las guías de apoyo y plan de seguimiento:** Una vez identificadas las técnicas adecuadas mediante una discusión entre mentores y equipo de trabajo, se establece un plan de seguimiento, garantizando que haya un proceso sistemático para evaluar el progreso y hacer ajustes si es necesario.

- **Ejecución del plan de seguimiento:** Sin una ejecución práctica y la aplicación de las guías, la mentoría sería meramente teórica. Esta fase asegura que el equipo ponga en práctica lo aprendido y reciba retroalimentación constante, lo cual es esencial para el aprendizaje y la mejora.

- **Reunión de cierre - Evaluación final:** Como en cualquier proceso, es crucial evaluar los resultados al final. Esta reunión permite al equipo de trabajo y al mentor discutir el progreso, identificar logros y áreas de mejora, y recopilar comentarios para refinar el proceso en el futuro, así como el contenido de las guías de apoyo.

- **Generación de Informe:** Documentar los hallazgos, recomendaciones y oportunidades de mejora no solo es valioso para el equipo actual, sino también para futuras mentorías. Este informe actúa como un recurso que puede ser utilizado para adaptar y mejorar el proceso a lo largo del tiempo.

- **Logística de reuniones:** La claridad en la modalidad, horarios y métodos de contacto, asegura que las reuniones sean efectivas y se minimicen malentendidos y pérdidas de tiempo.

- **Lo que el mentor puede esperar del equipo de trabajo y lo que el equipo de trabajo espera del mentor:** Establecer expectativas claras para ambas partes evita malentendidos y conflictos. Al detallar lo que cada parte puede esperar de la otra, se establece un marco de respeto mutuo y colaboración efectiva.

Siguiendo la estructura anteriormente mencionada, generamos un plan de mentoría con todas las definiciones necesarias el cual puede apreciarse en la siguiente subsección

5.2.2.2. Plan de mentoría

Tras detallar la necesidad y fundamento detrás de nuestro enfoque en el diseño del plan de mentoría, abordaremos a continuación su estructura y contenido.

Introducción

La mentoría es una herramienta muy útil para el desarrollo profesional, si lo llevamos al campo de la ingeniería de software su potencialidad está en permitir el crecimiento y aprendizaje continuo de todas las partes.

Para llevar a cabo esta mentoría se cuenta con una serie de guías de ayuda, con el objetivo de proporcionar una estructura clara y concreta sobre la aplicación práctica de determinadas técnicas de ingeniería de software.

Roles

Mentor: Una persona con experiencia, habilidades y conocimientos en un área específica que está dispuesta a compartirlos con otra persona. El mentor brinda orientación, consejos y retroalimentación al equipo de trabajo para ayudarlo a alcanzar sus objetivos.

Equipo de trabajo: Un grupo de personas que busca aprender y recibir orientación en un área específica de alguien con más experiencia. El mismo está dispuesto a recibir retroalimentación, aprender de la experiencia del mentor y utilizarla para su desarrollo personal y profesional.

Objetivos

Un objetivo de esta actividad es lograr obtener retroalimentación valiosa sobre la utilización de las guías, con la intención de descubrir oportunidades de mejora tanto en las mismas como en el proceso, y que las mismas sean plasmadas en sucesivas instancias de mentoría, generando así un proceso de mejora continua.

Otro objetivo es que el equipo de trabajo logre con la aplicación de las guías y tenga el acompañamiento de los mentores, mejorando su producto, así como sus procesos de ingeniería de software.

Plan de trabajo

1. Reunión 1: Introducción y evaluación:

- Presentación del mentor y del equipo de trabajo.
- Evaluación de las necesidades del equipo en cuanto a ingeniería de software.
- Identificación de las áreas donde trabajar y objetivos específicos para la mentoría
- Determinación de la fecha, hora y modalidad (ver logística de reuniones) de la próxima reunión.
- Elaboración de acta de la reunión por parte del mentor y compartida al equipo de trabajo por mail.

2. Identificación de técnicas

Identificación de las técnicas de ingeniería de software más adecuadas para el equipo por parte del mentor, basándose en sus objetivos y recursos disponibles.

3. Reunión 2: Introducción a las guías de apoyo y plan de seguimiento

- Presentación y justificación de las guías técnicas seleccionadas por el mentor, que se ajustan a las necesidades obtenidas en la reunión 1, siempre abiertos a la discusión entre las partes.
- Definición de una estrategia de seguimiento, en donde el equipo de trabajo se sienta cómodo y no se interfiera con su dinámica de trabajo.
- Determinación del período de tiempo en el cual se llevará a cabo la mentoría.
- Determinación de la fecha, hora y modalidad (ver logística de reuniones) de la primera reunión definida en la estrategia de seguimiento.
- Elaboración de acta de la reunión por parte del mentor y compartida al equipo de trabajo por mail.

4. Ejecución del plan de seguimiento

Esta actividad se trata de la aplicación sistemática de las actividades definidas en el plan de seguimiento.

5. Reunión de cierre: Evaluación

- Evaluación del progreso del equipo de trabajo en cuanto a la implementación de las técnicas de ingeniería de software.
- Presentación de formulario de retroalimentación al equipo de trabajo, referente al proceso de mentoría, el contenido de las guías de ayuda, y oportunidades de mejora en cualquier aspecto.

6. Generación de Informe

Esta actividad es llevada a cabo por el mentor luego de que el proceso de mentoría termine, y consiste en la generación de un informe sobre las posibilidades de mejora tanto del proceso de mentoría como de las guías de ayuda, dado la retroalimentación obtenida en la reunión de cierre y su experiencia personal al aplicar el proceso.

Proceso de mejora:

- Dada la información obtenida durante la ejecución del plan de seguimiento y la reunión final, mejorar las guías y el proceso de mentoría en cuanto sea posible.
- En caso de que no se disponga de tiempo, pero, aun así, haber encontrado oportunidades de mejora, documentarlo para que otro mentor pueda evaluarlo y continuar el proceso de mejora continua.

Diagrama de plan de trabajo

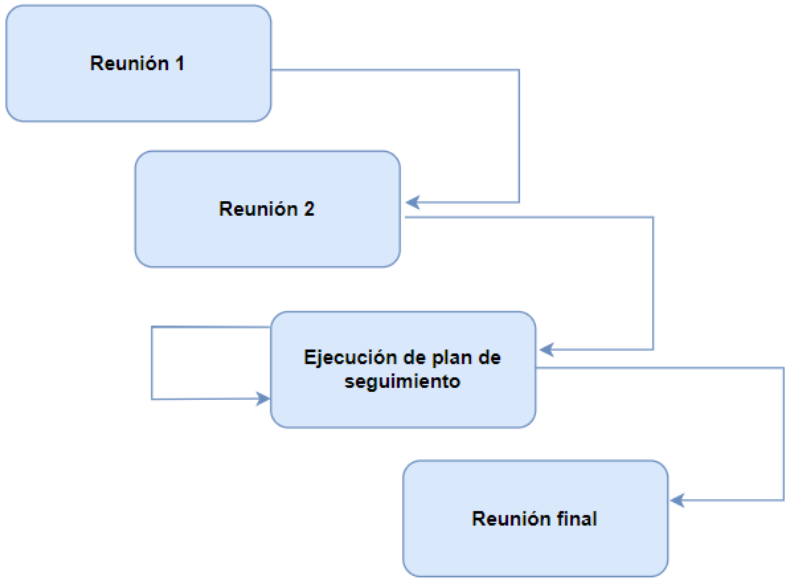

Ilustración 1. Diagrama que denota la ejecución del plan de mentoría

Logística de reuniones

Las reuniones podrán ser tanto de manera presencial como remota, adaptándose de la mejor manera a las actividades y posibilidades de los interesados.

Se podrá definir las particularidades que el equipo de trabajo crea conveniente en cuanto a la modalidad de trabajo, por ejemplo:

- Horarios
- Días
- Solo un integrante de los interesados participará de las reuniones.
- Método de contacto como Whatsapp, Teams, correo electrónico u otros
- Etc

Lo que el mentor puede esperar del equipo de trabajo

- Aceptar la relación de manera temporal, durante un periodo determinado.
- Tener reuniones periódicas en persona u *online*.
- Pedir sugerencias o consejos.
- Ser escuchado. Aplicar al menos algunos de los consejos y hacerle saber los resultados.
- Mantener los compromisos asumidos.
- Mantener la confianza.

Lo que el equipo de trabajo puede esperar del mentor

- Tener reuniones periódicas en persona u *online*.
- Proporcionar buenos consejos sobre sus actividades y desarrollo profesional.

- Mantener confidencias entre las partes.
- Dar seguimiento a los compromisos asumidos.
- Ayudar a resolver conflictos.
- Proporcionar comentarios honestos pero justos y diplomáticos.

5.3. Reporte de validación e implementación de guías y plan de mentoría

5.3.1 Búsqueda y selección de expertos para validación de guías

En nuestra búsqueda de colaboradores idóneos para ofrecer un análisis crítico de nuestras guías, tanto en contenido como en estructura y técnicas empleadas, identificamos tres perfiles que se alineaban con nuestras expectativas: docentes del laboratorio de ingeniería de software, egresados que han llevado a cabo proyectos de investigación relacionados con emprendimientos de software, y emprendedores en el ámbito del software, todos ellos vinculados a la Universidad ORT Uruguay.

Inicialmente, contactamos a seis de ellos, obteniendo respuesta de cinco, con quienes iniciamos el proceso de validación de las guías. Sin embargo, debido a demoras tanto en el llenado de formularios como en algunas respuestas de la comunicación por correo electrónico, optamos por ampliar nuestro alcance. Así, decidimos ampliar el espectro y terminamos por contactar a 23 personas adicionales que habían realizado proyectos con el Centro de Innovación y Emprendimientos (CIE). De este último grupo, lamentablemente, sólo una persona nos respondió.

En este punto contamos con un docente del laboratorio de ingeniería de software, tres expertos relacionados con emprendimientos y dos relacionados a proyectos de investigación los cuales estaban dispuestos a participar de esta instancia.

El contacto se realizó por correo electrónico donde explicamos el motivo de nuestro contacto, comentamos las características de nuestro proyecto, y cuál era la motivación de solicitar su retroalimentación. Aunque intentamos contactar a un amplio espectro de profesionales, no todos mostraron interés en participar.

5.3.2. Formación de grupos, diseño de encuestas y respuestas de expertos

Para garantizar que nuestras guías sean relevantes y útiles, iniciamos un estudio para evaluarlas, que hemos diseñado para los emprendimientos de software en el área de ingeniería de software.

Inicialmente diseñamos y distribuimos a los expertos una encuesta que abordó diversos aspectos, entre ellos, la estructura de las guías, la presentación del proceso de desarrollo, la relevancia, completitud y claridad del contenido. Estos aspectos son cruciales para garantizar que las guías sean no sólo informativas, sino también aplicables en escenarios reales.

A continuación, presentamos una breve descripción de los expertos que amablemente contribuyeron con sus opiniones:

Experto	**Descripción**
Mario Camerota	Emprendedor graduado de la Universidad ORT Uruguay y co-fundador de Colectate.
Andrés Bentos	Egresado con un proyecto de investigación centrado en emprendimientos de software.
Rodrigo Stratta	Emprendedor graduado de la Universidad ORT Uruguay y co-fundador de Larifamos.com.
Darío Macchi	Docente del Laboratorio de Ingeniería de Software.
Santiago Ingold	Parte del equipo del emprendimiento de software Black Pitbull, centrado en la sistematización de procesos de ciberseguridad.

Pablo Benítez	Egresado con un proyecto de investigación centrado en emprendimientos de software.

Tabla 1. Descripción de expertos contactados durante la investigación

5.3.2.1. Creación de grupos heterogéneos

Nos tomamos el tiempo de considerar cuidadosamente el volumen de material proporcionado a cada experto en el proceso de nuestra encuesta, optando por limitarlo a cuatro guías por persona. Esta decisión no fue tomada a la ligera, sino que es el resultado de un esfuerzo por mantener un equilibrio entre la necesidad de recopilar información detallada, y el deseo de evitar que el proceso se vuelva demasiado abrumador para los participantes.

Entendemos que las encuestas, si bien son una herramienta crucial para recoger información valiosa, pueden resultar tediosas y pesadas si se presentan en grandes volúmenes. Queremos garantizar que cada persona se sienta motivada y comprometida durante todo el proceso, evitando que la densidad de las guías suponga una carga o disminuya su entusiasmo.

Para esto formamos tres subgrupos distintos, cada uno de ellos incluyendo cuatro guías de apoyo distintas. Como está establecido en el diseño metodológico, cada uno de estos grupos está conformado por distintos tipos de expertos.

A continuación, detallaremos la estructura de estas asignaciones:

	Integrantes	**Técnicas**
Grupo de expertos 1	Mario Camerota Andrés Bentos	Pruebas AB Encuestas Mockups CI/CD
Grupo de expertos 2	Darío Macchi Santiago Ingold	Pruebas de Usabilidad Entrevistas Prototipos Observación

Grupo de expertos 3	Pablo Benítez Rodrigo Stratta	Estudio de Casos Wireframes Pruebas Unitarias Manejo de Deuda Técnica

Tabla 2. Asignación de grupos de expertos y técnicas

5.3.2.2. Diseño y distribución de encuestas

Siguiendo con el contacto establecido con los expertos, enviamos a cada uno, correspondiente a los grupos previamente definidos, las cuatro guías mencionadas junto con un enlace al formulario para recopilar sus comentarios.

Estos formularios pueden verse en el Anexo 14, Anexo 15 y Anexo 16 para los grupos 1, 2 y 3 respectivamente.

Utilizamos la escala de Likert en las encuestas para evaluar las respuestas de los participantes. La puntuación para cada pregunta del formulario se encuentra en el rango entre 1 y 5. En esta escala, 1 se interpreta como "muy mal" y 5 como "muy bien". Entre estos extremos, el 2 representa "mal", 3 es considerado "neutral", y 4 significa "bien".

De esta manera, ofrecimos a los encuestados una gama de opciones que les permitió expresar con precisión su opinión o experiencia respecto a la pregunta en cuestión.

5.3.2.3. Respuestas de expertos

Experiencias generales a todos los grupos

En la primera sección del formulario, planteamos dos preguntas de carácter general, diseñadas para obtener una perspectiva amplia sobre el material presentado. Estas interrogantes abordan la estructura y el proceso de desarrollo tal como se describen en las guías. Luego de estas preguntas iniciales, el cuestionario prosigue con una serie de ítems específicos centrados en la validación de diversas técnicas, estas serán abarcadas en las próximas secciones.

Experto	**¿Cómo considera la estructura de las guías?** Por estructura entendemos a las secciones de las mismas y su orden.	**¿Cómo evalúa la presentación del proceso de desarrollo en las guías?** Por proceso de desarrollo entendemos la sección donde se define la aplicación de la técnica en sí.
Mario Camerota	4	5
Andrés Bentos	5	4
Darío Macchi	5	4
Santiago Ingold	4	5
Pablo Benitez	5	5
Rodrigo Stratta	5	4

Tabla 3. Resultados de preguntas generales realizadas a expertos

Experiencia con grupo de expertos 1

	Mario Camerota	**Andrés Bentos**
¿Encuentra que el contenido de la guía de **pruebas A/B** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	4	3
¿Considera que la guía de **pruebas A/B** es completa en términos de los aspectos que cubre?	5	4
¿Cómo evalúa la claridad de las explicaciones de la guía de **pruebas A/B**?	5	4
¿Encuentra que el contenido de la guía de **encuestas** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	5	5
¿Considera que la guía de **encuestas** es completa en términos de los aspectos que cubre?	3	5
¿Cómo evalúa la claridad de las explicaciones de la guía de **encuestas**?	4	4
¿Encuentra que el contenido de la guía de **mockups** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	3	5

¿Considera que la guía de **mockups** es completa en términos de los aspectos que cubre?	2	4
¿Cómo evalúa la claridad de las explicaciones de la guía de **mockups**?	4	4
¿Encuentra que el contenido de la guía de **CI/CD** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	2	4
¿Considera que la guía de **CI/CD** es completa en términos de los aspectos que cubre?	4	4
¿Cómo evalúa la claridad de las explicaciones de la guía de **CI/CD**?	4	4

Tabla 4. Resultados de preguntas específicas realizadas al grupo 1

Experiencia con grupo de expertos 2

	Darío Macchi	**Santiago Ingold**
¿Encuentra que el contenido de la guía de **pruebas de usabilidad** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	5	3
¿Considera que la guía de **pruebas de usabilidad** es completa en términos de los aspectos que cubre?	4	4
¿Cómo evalúa la claridad de las explicaciones de la guía de **pruebas de usabilidad**?	5	5
¿Encuentra que el contenido de la guía de **entrevistas** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	5	4
¿Considera que la guía de **entrevistas** es completa en términos de los aspectos que cubre?	4	4
¿Cómo evalúa la claridad de las explicaciones de la guía de **entrevistas**?	5	5
¿Encuentra que el contenido de la guía de **prototipos** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	5	4
¿Considera que la guía de **Prototipos** es completa en términos de los aspectos que cubre?	4	4

¿Cómo evalúa la claridad de las explicaciones de la guía de **prototipos**?	5	5
¿Encuentra que el contenido de la guía de **observación** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	5	4
¿Considera que la guía de **observación** es completa en términos de los aspectos que cubre?	4	4
¿Cómo evalúa la claridad de las explicaciones de la guía de **observación**?	5	5

Tabla 5. Resultados de preguntas específicas realizadas al grupo 2

Experiencia con grupo de expertos 3

	Pablo Benítez	**Rodrigo Stratta**
¿Encuentra que el contenido de la guía de **estudio de casos** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	3	4
¿Considera que la guía de **estudio de casos** es completa en términos de los aspectos que cubre?	5	4
¿Cómo evalúa la claridad de las explicaciones de la guía de **estudio de casos**?	4	5
¿Encuentra que el contenido de la guía de **wireframes** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	5	4
¿Considera que la guía de **wireframes** es completa en términos de los aspectos que cubre?	5	4
¿Cómo evalúa la claridad de las explicaciones de la guía de **wireframes**?	5	4
¿Encuentra que el contenido de la guía de **pruebas unitarias** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	5	5
¿Considera que la guía de **pruebas unitarias** es completa en términos de los	5	3

aspectos que cubre?		
¿Cómo evalúa la claridad de las explicaciones de la guía de **pruebas unitarias**?	5	4
¿Encuentra que el contenido de la guía de **manejo de deuda técnica** es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?	5	4
¿Considera que la guía de **manejo de deuda técnica** es completa en términos de los aspectos que cubre?	4	4
¿Cómo evalúa la claridad de las explicaciones de la guía de **manejo de deuda técnica**?	5	4

Tabla 6. Resultados de preguntas específicas realizadas al grupo 3

5.3.3. Diseño de entrevistas a expertos

Posterior a que los expertos completaran el formulario, solicitamos tener una reunión para realizar una entrevista con cada uno de ellos, y así lograr potenciar la información recolectada de la encuesta.

Para realizar la entrevista nos basamos en la guía de entrevista, la cual puede verse en Anexo 2, que nosotros diseñamos, y de este modo, ya tuvimos un acercamiento a su utilización real.

En base a los resultados obtenidos en el formulario, diseñamos entrevistas específicas a cada entrevistado, enfocadas en las preguntas que más nos podían aportar información, ya sea para confirmar ideas como para profundizar aspectos que, en la encuesta cerrada, la valoración numérica nos indicaba que había un área de mejora para trabajar.

Preparación de entrevistas

A continuación, presentamos las preguntas que fueron el hilo conductor de la entrevista en las cuales permitimos que el entrevistado pudiera expresarse lo más libremente posible, fomentando así la discusión y el intercambio de ideas. Estas están estructuradas en forma de tabla para facilitar la presentación en este documento.

Mario Camerota
¿Qué le pareció la estructura de las guías?
Respecto a la guía de Encuestas, ¿Hay algún aspecto que le parece relevante para mejorar la completitud de esta?
Respecto a la guía de Mockups, ¿Hay algún aspecto que le parece relevante para mejorar la completitud de esta?
¿Qué lo llevó a considerar que la técnica de mockups no es tan relevante?
¿Qué lo llevó a considerar que la técnica de CI/CD no es tan relevante?
¿Hay alguna información adicional que agregaría a las guías para hacerlas más útiles para emprendimientos de software?
¿Hubo algún contenido que le resultara confuso o poco claro?
¿Sugiere agregar algún otro tipo de información a las guías?
¿Sugiere alguna otra técnica de ingeniería de software que debería incluirse en estas guías?
¿Qué opinión tiene sobre el proyecto que estamos llevando a cabo?

Tabla 7. Preguntas preparadas para entrevista con Mario Camerota

Andrés Bentos
¿Qué le pareció la estructura de las guías?
Respecto a la guía de Pruebas A/B, ¿Por qué consideras que no es tan aplicable o relevante?
¿Hay alguna información adicional que agregaría a las guías para hacerlas más útiles para emprendimientos de software?
¿Hubo algún contenido que le resultara confuso o poco claro?
¿Sugiere agregar algún otro tipo de información a las guías?

¿Sugiere alguna otra técnica de ingeniería de software que debería incluirse en estas guías?
¿Qué opinión tiene sobre el proyecto que estamos llevando a cabo?

Tabla 8. Preguntas preparadas para entrevista con Andrés Bentos

Darío Macchi
¿Qué le pareció la estructura de las guías?
¿Hay alguna información adicional que agregaría a las guías para hacerlas más útiles para emprendimientos de software?
¿Hubo algún contenido que le resultara confuso o poco claro?
¿Sugiere agregar algún otro tipo de información a las guías?
¿Sugiere alguna otra técnica de ingeniería de software que debería incluirse en estas guías?
¿Qué opinión tiene sobre el proyecto que estamos llevando a cabo?

Tabla 9. Preguntas preparadas para entrevista con Darío Macchi

Rodrigo Stratta
¿Qué le pareció la estructura de las guías?
Respecto a la guía de Encuestas, ¿Hay algún aspecto que le parece relevante para mejorar la completitud de la misma?
¿Hay alguna información adicional que agregaría a las guías para hacerlas más útiles para emprendimientos de software?
¿Hubo algún contenido que le resultara confuso o poco claro?
¿Sugiere agregar algún otro tipo de información a las guías?
¿Sugiere alguna otra técnica de ingeniería de software que debería incluirse en estas guías?

¿Qué opinión tiene sobre el proyecto que estamos llevando a cabo?

Tabla 10. Preguntas preparadas para entrevista con Rodrigo Stratta

Pablo Benítez
¿Qué le pareció la estructura de las guías?
¿Qué lo llevó a considerar que la técnica de Estudio de Casos no es tan relevante?
¿Hay alguna información adicional que agregaría a las guías para hacerlas más útiles para emprendimientos de software?
¿Hubo algún contenido que le resultara confuso o poco claro?
¿Sugiere agregar algún otro tipo de información a las guías?
¿Sugiere alguna otra técnica de ingeniería de software que debería incluirse en estas guías?
¿Qué opinión tiene sobre el proyecto que estamos llevando a cabo?

Tabla 11. Preguntas preparadas para entrevista con Pablo Benítez

5.3.4. Realización de entrevistas a expertos

Una vez definida la estructura anterior, coordinamos con cada uno de los expertos una fecha y hora que les resultará cómoda para tener la entrevista, siendo en todos los casos las mismas realizadas de forma virtual. Con la autorización de los entrevistados, grabamos todas las entrevistas salvo la de Rodrigo Stratta, en la cual tuvimos problemas técnicos. No obstante, logramos tomar apuntes durante esta entrevista, que luego utilizamos para hacer su correspondiente análisis.

Para la ejecución y análisis de la entrevista, seguimos ajustados a los lineamientos marcados en la guía de entrevistas, intentando fielmente aplicar el paso a paso.

Para facilitar el análisis, luego de cada una de las entrevistas, realizamos una transcripción a partir de las grabaciones. Estas transcripciones pueden verse en el Anexo 17.

5.3.5. Búsqueda de equipos emprendedores y presentación de guías y mentoría

A partir del listado de proyectos que están en curso o comenzando que son del tipo emprendedor en Universidad ORT Uruguay, obtuvimos cuales son los grupos a los cuales podíamos contactar.

Este listado tenía la particularidad de ser acotado, ya que solo cuatro proyectos cumplían con la característica buscada.

Nuestra estrategia se basó en primera instancia en contactar a los tutores de dichos proyectos, ya que nos pareció de orden plantearles el tema a ellos y tener una primera aprobación; En caso de que estuvieran de acuerdo proceder a contactar a los estudiantes directamente.

El contacto con dichos tutores se realizó vía email, en donde explicamos nuestra intención y enviamos adjuntas dos de las guías de apoyo para que puedan ver de qué se tratan, y además enviamos el plan de mentoría que planeamos ejecutar con ellos en caso de que así lo deseen.

También en dicho correo electrónico dejamos planteada la posibilidad de tener una reunión con dicho tutor y explicar en profundidad la modalidad, así como despejar cualquier duda que tuviera.

En la totalidad de los casos donde obtuvimos respuesta, que fueron cuatro de los cinco contactados, los tutores se mostraron muy dispuestos, manifestando que creían de utilidad aplicar este plan, así como que sus equipos tutorados tuvieran acceso a las guías de apoyo.

En algunos casos, se optó por tener reuniones para profundizar conceptos. Estas sesiones resultaron ser muy enriquecedoras. Creemos haber despejado todas las dudas y, además, pudimos discutir con los tutores acerca del plan de mentoría. De estas conversaciones obtuvimos valiosa retroalimentación, no solo del mismo, sino también sobre el proyecto en general. También recibimos comentarios acerca de la importancia de iniciativas como esta, ya que fomentan la colaboración entre diferentes equipos de proyectos.

En este punto estábamos en condiciones de contactar a los equipos de tesis para contarles sobre nuestro trabajo, ofrecer el acceso al material de las guías de apoyo generadas, así como la posibilidad de un plan de mentoría a demanda.

Contactamos a todos los equipos de tesis, en donde les explicamos que ya habíamos discutido las características de la propuesta con sus tutores, y les solicitamos si podíamos acceder a una reunión para profundizar. Para ganar fluidez y que lleguen a dichas reuniones con un acercamiento a las características de la propuesta, les enviamos previamente a las mismas como adjunto en el correo electrónico de contacto, una de las guías y el plan de mentoría que deseamos aplicar con cada uno de ellos.

Efectuamos las reuniones con cada uno de los grupos de forma individual, en donde en las mismas expusimos cuáles eran las 12 técnicas que habían sido seleccionadas para tener una guía de apoyo. Los equipos se manifestaron interesados, al menos en acceder al material, y quedaron de confirmar por correo electrónico si también les interesaba acceder al plan de mentoría.

5.3.6. Inicio y ejecución del plan de mentoría

Los equipos contactados tienen particularidades dispares, algunos equipos estaban comenzando su proyecto, mientras que otros ya estaban en etapas más avanzadas.

Esto nos significó una complicación, ya que los equipos que recién comenzaban aún no estaban aplicando procesos que pudieran utilizar algunas de las guías y otros en etapas más avanzadas ya habían realizado alguno de los procesos. A pesar de eso, la gran mayoría de las técnicas consideramos que podrían haberse aplicado en algunos de esos proyectos.

Luego de estas instancias de reuniones, notamos un decremento en la receptividad por parte de los equipos, la comunicación se volvió muy frustrada. Como equipo analizamos esta situación y llegamos a una posible conclusión de que el desencadenante principal fue este desfase de las distintas etapas entre nuestro proyecto y los proyectos candidatos, lo que comprometió la aplicación del plan de mentoría, así como el tiempo acotado para ejecutar

dicho plan, teniendo en cuenta que nuestra intención era llegar a obtener conclusiones en el tiempo acotado de este proyecto de grado.

También consideramos que pudo haber otros factores particulares en cada equipo, como falta de tiempo o simplemente no estar a fin de compartir detalles de sus proyectos. Al respecto intentamos comunicarnos para lograr llegar a la causa raíz de esta comunicación frustrada y obtuvimos el mismo resultado, no pudiendo aclarar la situación.

5.3.7. Validación de guías por retrospectiva

Habiendo considerado la situación anterior como un riesgo para el proyecto, decidimos desarrollar una nueva estrategia, y nos enfocamos en intentar validar solo las guías de apoyo, pero esta vez en retrospectiva con los grupos de proyecto que están en proceso, nombrados con anterioridad, que ya hayan aplicado algunas de estas técnicas. La idea consiste en comparar cómo estos equipos aplicaron estas técnicas y contrastar sus procesos con los que planteamos nosotros en cada una de nuestras guías de apoyo.

Para ello, decidimos nuevamente crear un formulario para poder obtener este tipo de validaciones, teniendo en cuenta los lineamientos establecidos en la guía de ayuda que hemos creado para las encuestas. Si bien tenemos claro que un formulario para una encuesta debe de ser lo más específico y conciso posible para que el encuestado no pierda el foco y pierda el interés, decidimos poner todas las técnicas de las guías que hemos realizado, aclarando a dicho encuestado que responda las preguntas sobre las técnicas que, si haya utilizado, ignorando las otras.

Idealmente, habríamos priorizado la realización de entrevistas en lugar de encuestas. Sin embargo, debido a la limitada respuesta obtenida para colaborar con nuestro proyecto, determinamos que este enfoque nos permitiría obtener esta valiosa validación.

Este formulario incluye doce secciones, en las cuales cada sección corresponde a cada una de las técnicas que hemos obtenido. Todas estas secciones contienen las preguntas con una misma estructura, pero cada una de ellas es específica para la técnica correspondiente. En el siguiente ejemplo, puede verse el caso del cuestionario sobre la técnica de pruebas A/B. El resto del formulario está compuesto por secciones análogas a la siguiente.

Pregunta	Forma de evaluación
¿Qué tan similar considera que es el proceso de aplicación de la técnica de pruebas A/B, descrito en nuestra guía, en comparación con la manera en la que ustedes la implementaron en su proyecto?	La valoración se encuentra en el rango de 1 y 5. En esta escala, el 1 se interpreta como "muy diferente" y el 5 como "muy similar". Entre estos extremos, el 2 representa "diferente", el 3 es considerado "neutral", y el 4 significa "similar".
¿Cuál fue el mayor desafío que enfrentaron al implementar la técnica de pruebas A/B en su proyecto? Por favor, describa cualquier problema o dificultad que hayan encontrado.	Pregunta abierta
¿Cómo mejorarías la guía de pruebas A/B? Por favor, proporcione sugerencias específicas o áreas que considera podrían mejorarse en nuestra guía.	Pregunta abierta
¿Tiene algún comentario adicional sobre la técnica de pruebas A/B o la guía que le proporcionamos? Por favor, comparta cualquier observación o comentario adicional que pueda tener.	Pregunta abierta

Tabla 12. Ejemplo de sección de encuesta de validación por retrospectiva

5.3.8. Distribución de encuesta de validación en retrospectiva

Dado lo anterior, decidimos dirigir esta encuesta hacia los grupos de estudiantes con los que hemos establecido contacto previo y que en la actualidad están desarrollando proyectos de tipo emprendedor. A pesar de no haber logrado la colaboración deseada en relación con el plan de mentoría y la aplicación práctica de nuestras guías de apoyo, consideramos que, al invitarlos a completar una encuesta retrospectiva, podríamos hacerle más atractiva la

propuesta y, por tanto, obtener una valiosa retroalimentación para seguir validando las guías de apoyo.

En concreto, le enviamos el formulario de validación por retroalimentación a tres grupos, y con el grupo restante acordamos con su tutor hacer otro tipo de validación. Esta diferenciación se realizó debido a que los primeros tres grupos ya habían pasado por procesos donde habían aplicado algunas técnicas de las cuales realizamos una guía de apoyo y por tal motivo tenía sentido esta instancia de validación en retrospectiva, y el restante grupo manifestó tener expectativas de poder aplicar alguna de las mismas, por eso haremos una instancia de retroalimentación luego de que se efectivice la utilización de nuestras guías. Este caso particular se profundizará más adelante en este documento.

La suma de los integrantes de los tres grupos contactados da un total de 11 personas, las cuales todas recibieron la invitación a participar del formulario en retrospectiva. Amablemente, dos de estas 11 personas nos respondieron el formulario, siendo ambas de equipos distintos.

No obstante, consideramos que necesitábamos un número mayor de respuestas, por lo que decidimos obtener información sobre los proyectos de tesis del tipo emprendimiento de los últimos años.

Felizmente pudimos obtener un listado con esta información, comprendiendo el período desde agosto de 2016 hasta agosto del año 2020, en donde obtuvimos los contactos de un total de 23 personas.

De estas, dos personas respondieron la encuesta. Por tanto, pudimos obtener cuatro validaciones por retrospectiva en total, las cuales serán analizadas a continuación.

5.3.9. Encuestas de validación en retrospectiva

Después de contactar en total a 34 personas para solicitar su participación en esta instancia de validación en retrospectiva, logramos contar con el compromiso y las contribuciones de cuatro de ellos. Donde dos de ellos están actualmente llevando adelante sus proyectos del tipo emprendedor, mientras que las dos restantes lo han realizado en años anteriores.

Sin más, a continuación, se encuentra una tabla en la cual denotamos cuales técnicas fueron utilizadas por los participantes, ya que no necesariamente aplicaron todas las técnicas que se encuentran en nuestras guías.

A continuación, se definen los participantes de la encuesta:

- **Participante 1 - Miguel Nuñez:** Al momento de su participación en el análisis por retrospectiva, se encontraba realizando su proyecto de emprendedurismo en Universidad ORT Uruguay.
- **Participante 2 - Constanza Curzio:** Egresada con proyecto final de emprendedurismo en Universidad ORT Uruguay.
- **Participante 3 - Andrés Juan Montaldo:** Al momento de su participación en el análisis por retrospectiva, se encontraba realizando proyecto de emprendedurismo en Universidad ORT Uruguay.
- **Participante 4 - Gastón Donadio:** Egresado con proyecto final de emprendedurismo en Universidad ORT Uruguay.

	Miguel Nuñez	**Constanza Curzio**	**Andrés Juan Montaldo**	**Gastón Donadio**
Pruebas A/B				
Prueba de usabilidad				
Test Unitarios	Se utilizó	Se utilizó	Se utilizó	Se utilizó
Estudio de casos				Se utilizó
Prototipo	Se utilizó			Se utilizó
Mockups				Se utilizó
Wireframes				
Entrevistas			Se utilizó	Se utilizó
Observación		Se utilizó		
Encuestas	Se utilizó	Se utilizó		Se utilizó

Manejo de deuda técnica		Se utilizó		Se utilizó
CI/CD		Se utilizó	Se utilizó	Se utilizó

Tabla 13. Técnicas aplicadas por participantes de acuerdo con su experiencia previa

A continuación, detallamos las preguntas que se realizaron a los participantes, respecto a cada una de las técnicas que utilizaron:

Pregunta 1: ¿Qué tan similar considera que es el proceso de aplicación de la técnica descrita en nuestra guía, en comparación con la manera en la que ustedes la implementaron en su proyecto?

Pregunta 2: ¿Cuál fue el mayor desafío que enfrentaron al implementar la técnica en su proyecto? Por favor, describa cualquier problema o dificultad que hayan encontrado.

Pregunta 3: ¿Cómo mejorarías la guía? Por favor, proporcione sugerencias específicas o áreas que considera podrían mejorarse en nuestra guía.

Pregunta 4: ¿Tiene algún comentario adicional sobre la técnica o la guía que le proporcionamos? Por favor, comparta cualquier observación o comentario adicional que pueda tener.

Estudio de casos	
Pregunta 1	
Gastón	4
Pregunta 2	
Gastón	"*Obtener la información de cosas tan específicas no es tan fácil en general. En nuestro caso utilizamos un conjunto de tecnologías poco exploradas lo cual nos dificultó justamente esto.*"
Pregunta 3	

Gastón	"*Mencionaría algo sobre la dificultad de encontrar la información sobre cosas tan particulares.*"

Tabla 14. Respuestas a preguntas sobre la guía de apoyo de estudio de casos

Test unitarios	
Pregunta 1	
Gastón	5
Miguel	5
Constanza	5
Andrés	Sin valoración
Pregunta 2	
Gastón	"*No encontramos dificultad en esta técnica. El equipo ya tenía experiencia trabajando de esta forma y no fue difícil aplicarlo al proceso de desarrollo.*"
Miguel	"*Nada*"
Constanza	Sin valoración
Andrés	"*Como comentaba en la parte de CI/CD, nuestro mayor desafío fue hacer pruebas de componentes de IA (nunca antes habíamos hecho). Fue más que nada un desafío técnico, tuvimos que salir a buscar cómo probar algo así.*"
Pregunta 3	
Gastón	"Podría mencionar muy anecdóticamente las estrategias de pruebas como TDD, BDD u otras"

Miguel	"*Librerías más usadas para lenguaje como java, nodejs, c# con plantillas ejemplos*"
Constanza	Sin valoración
Andrés	"*Esta muy completa, y fue exactamente los pasos que seguimos para hacer nuestras pruebas*."
Pregunta 4	
Gastón	"*Cualquier proyecto que quiera tener calidad, necesita utilizar esta técnica en su desarrollo.*"
Miguel	"*No*"
Constanza	Sin valoración
Andrés	Sin valoración

Tabla 15. Respuestas a preguntas sobre la guía de apoyo de pruebas unitarias

Prototipo	
Pregunta 1	
Gastón	5
Miguel	4
Pregunta 2	
Gastón	"*La dificultad que tuvimos a la hora de aplicar esta técnica, fue saber hasta dónde iba el prototipo, es decir, siempre queríamos darle más funcionalidades de las que eran necesarias.*"
Miguel	"*Qué prototipo usar según el cliente*."
Pregunta 3	

Miguel	"*Qué prototipo usar según el cliente*."
Pregunta 4	
Gastón	"*Si bien los prototipos sirven para validar productos, también sirve en caso de productos muy grandes, para ir entregando valor más rápidamente.*"
Miguel	"*No*"

Tabla 16. Respuestas a preguntas sobre la guía de apoyo de prototipo

Mockups	
Pregunta 1	
Gastón	5
Pregunta 2	
Gastón	"*En este caso aplicamos al 100% la técnica de mockups y no tuvimos dificultad*."
Pregunta 3	
Gastón	Sin valoración.
Pregunta 4	
Gastón	"*Esta técnica nos sirvió realmente, tanto para definir requerimientos como para explicar a las personas externas qué era lo que queríamos hacer. La recomiendo rotundamente.*"

Tabla 17. Respuestas a preguntas sobre la guía de apoyo de mockups

Entrevistas	
Pregunta 1	
Gastón	5

Andrés	5
Pregunta 2	
Gastón	"*El mayor desafío fue encontrar a las personas adecuadas para entrevistar. En nuestro caso se trataba de expertos sobre el negocio. También el tema de tiempos fue un poco complicado ya que los expertos solo podían en los horarios que nosotros dedicamos a nuestros trabajos.*"
Andrés	"*El mayor desafío que tuvimos al realizar las entrevistas fue conseguir gente para entrevistar. No nos fue fácil conseguir gente con e-commerces a quien podamos entrevistar para validar la idea, aunque la gente en sí está dispuesta a dedicarte un rato. Lo que hicimos fue salir a buscar gente por linkedin. Y nos funcionó bien. Creo que en su guía no está explicitado esto, y estaría bueno agregarlo: métodos para conseguir gente para entrevistar; potenciales usuarios.*"
Pregunta 3	
Gastón	"*Yo creo que algo muy importante en la entrevista es tener una parte estructurada, pero mayoritariamente dejar hablar a la otra persona, llevarlo a un lado que se sienta cómodo (en la conversación) y pueda expresarse libremente. Si bien dice que las entrevistas no son siempre las adecuadas dependiendo de la situación, yo creo que es de las técnicas más enriquecedoras.*"
Andrés	"*En general seguimos los mismos pasos excepto la documentación de requisitos; muchas veces no salen requisitos de las entrevistas sino más bien conceptos generales que luego nosotros usamos para identificar casos de uso y requisitos funcionales.*"
Pregunta 4	
Gastón	Sin valoración.
Andrés	Sin valoración.

Tabla 18. Respuestas a preguntas sobre la guía de apoyo de entrevistas

Observación	
Pregunta 1	
Constanza	5
Pregunta 2	

Constanza	Sin valoración
Pregunta 3	
Constanza	Sin valoración
Pregunta 4	
Constanza	Sin valoración

Tabla 19. Respuestas a preguntas sobre la guía de apoyo de observación

Encuestas	
Pregunta 1	
Gastón	5
Miguel	4
Pregunta 2	
Gastón	"*El desafío más grande que encontramos fue que la gente se lo tome en serio. Muchas veces a las personas les aburre hacer encuestas y no le dedican la suficiente energía para dar buenas respuestas. Entiendo que acá fallamos nosotros probablemente por el tipo de encuestas.*"
Miguel	"*Ninguna*"
Constanza	"*La difusión. Hacerlo llegar a la mayor cantidad de personas posibles.*"
Pregunta 3	
Gastón	"*Creo que está muy acertada la guía. Y destacó que es importante segmentar bien los usuarios y tener claro qué información queremos obtener.*"
Miguel	"*Ninguna*"
Constanza	"*Capaz dejaría más claro el tema de no formular la pregunta esperando cierta respuesta por parte de los encuestados, o no formularla de tal manera que decir una negativa sea*

	mal visto"
Pregunta 4	
Gastón	"*Es muy importante saber que las encuestas no sirven para todo, hay que saber cuándo aplicarlas y hacerlas lo más amigables posibles porque sino la gente se aburre.*"
Miguel	"*Ninguna*"

Tabla 20. Respuestas a preguntas sobre la guía de apoyo de encuestas

Manejo de deuda técnica	
Pregunta 1	
Gastón	3
Constanza	4
Pregunta 2	
Gastón	"*La mayor dificultad es dedicar un tiempo para resolver la deuda. Siempre hay un conflicto de intereses, por un lado, está el software funcionando con su deuda técnica correspondiente, pero por otro lado siempre se quiere mejorar la solución actual y resolver esas deudas técnicas.*"
Constanza	Sin valoración.
Pregunta 3	
Gastón	"*Yo no creo que la deuda técnica afecte al usuario como se menciona, sino que se trataría de un bug, y eso no lo considero deuda técnica.*"
Constanza	"*Documente la deuda técnica". En esta parte capaz agregaría que es muy beneficioso ir documentando la deuda técnica que encontramos, aunque en ese momento no tengamos tiempo/recursos para solucionarla.*"
Pregunta 4	
Gastón	"*Desde el lado de ingeniería la deuda técnica nunca termina, siempre queremos mejorar y mejorar el producto que tenemos, es difícil saber hasta dónde dedicarle tiempo.*"
Constanza	"*Por nuestra parte, a veces asumimos deuda técnica a sabiendas para cumplir con ciertos deadlines que nos imponemos (en general para probar algo en un ambiente más real y ver*

	si íbamos por el buen camino), por lo que íbamos documentando todo en el proceso."

Tabla 21. Respuestas a preguntas sobre la guía de apoyo de manejo de DT

CI/CD	
Pregunta 1	
Gastón	4
Constanza	5
Andrés	4
Pregunta 2	
Gastón	"*Yo creo que la mayor dificultad es realizar la configuración inicial, ya que luego no lleva mucho trabajo*."
Constanza	Sin valoración.
Andrés	"*Nuestro desafío actual que tenemos con CI/CD es el hecho de automatizar nuestras pruebas. Nuestras pruebas unitarias tienen la particularidad de que, al testear componentes de IA, utilizan también IA. Es decir, tenemos un set de preguntas base que wisello tiene que contestar bien (ej: Soy un niño de 5 años y quiero tomar whey -> wisello tiene que recomendarte que no lo hagas). Pero la respuesta de wisello no es fija, entonces tenemos que usar GPT para identificar si el resultado de la prueba está bien o no. Y eso cuesta plata, entonces no lo podemos correr cada vez como parte de nuestro pipeline*."
Pregunta 3	
Gastón	"*Lo veo bastante bien, lo único que se me ocurre es que, en la parte de automatizar pruebas, también se pueden automatizar cosas como chequeos de código como formateadores, etc. Para mantener los estándares.*"
Constanza	Sin valoración.
Andrés	"*En general, seguimos los mismos pasos.*"
Pregunta 4	

Gastón	Sin valoración.
Constanza	Sin valoración.
Andrés	Sin valoración.

Tabla 22. Respuestas a preguntas sobre la guía de apoyo de CI/CD

5.3.10. Validación de guías a través del uso práctico

Darío Macchi, también tutor de un equipo que está desarrollando su proyecto de fin de carrera, nos indicó que sus estudiantes estaban considerando la implementación de algunas de nuestras guías de apoyo. Esto debido a la fase del proyecto en que se encontraban y a las necesidades específicas que enfrentaban en ese momento, particularmente en áreas como entrevistas, wireframes, mockups y prototipos. Luego de que él les presentase las guías, nos informó que el equipo efectivamente había adoptado algunas de ellas. Posteriormente, nos facilitó el contacto de los estudiantes para que pudiéramos dialogar directamente con ellos.

La salvedad es que dicho equipo estaba realizando un proyecto de fin de carrera en conjunto con la industria, en lugar de ser del tipo emprendimiento, que es el tipo de proyecto que estábamos más interesados en evaluar, dadas sus similitudes con un emprendimiento de software.

No obstante, esto nos resultó interesante, ya que lamentablemente no pudimos cumplir esta etapa de validación en la práctica con proyectos de tipo emprendimiento, como comentamos en secciones anteriores.

5.3.10.1. Preparación y ejecución de entrevista con estudiantes de proyecto

A continuación, les mostramos las preguntas que guiaron nuestra entrevista con los integrantes de este equipo, Diego Amarilla y Nicolás Praderi. Es importante mencionar que, aunque las usamos como referente, también brindamos espacio para que el entrevistado se expresara con total libertad, incentivando el diálogo y el intercambio de perspectivas.

Las hemos organizado en formato de tabla para una mejor visualización en este documento.

Preguntas sobre las guías aplicadas
¿Qué técnicas decidieron implementar?
¿Cómo fue el proceso de implementación? ¿Fue sencillo seguir las guías?
¿Notaron algún impacto inmediato después de la implementación?
¿Hubo algún obstáculo o desafío inesperado durante la implementación?
¿Sintieron que la guía estaba completa o que faltaba información crucial?
¿Hay algo en las guías que consideran que se pueda mejorar o aclarar?
Preguntas sobre las guías no aplicadas
¿Por qué decidieron no implementar las demás técnicas?
¿Hubo algún término o sección en las guías no implementadas que les resultara confuso?
¿Hay alguna técnica que consideren implementar en el futuro cercano?
Utilidad de las guías
¿Recomendarían estas guías a otros emprendimientos de software?
¿Hay algún tema o técnica de ingeniería de software que les gustaría que se cubriera en futuras guías?

¿Tienen algún otro comentario o sugerencia para mejorar el proyecto?

Tabla 23. Preparación de entrevista sobre aplicación práctica de guías

Una vez realizada la anterior preparación para la entrevista, nos comunicamos para coordinarnos para llevar la entrevista a cabo con dicho equipo. La transcripción de la misma se encuentra en el Anexo 18**.**

6. Análisis y discusión de resultados

6.1. Análisis de encuesta de validación de guías de apoyo por expertos

Esta encuesta abordó tanto temas generales sobre las guías de apoyo, así como también temas más específicos en cuanto a cada una de las técnicas presentes en las guías.

6.1.1. Análisis de preguntas sobre consideraciones generales de las guías

Inicialmente, de acuerdo con Tabla 3, en la cual figuran las valoraciones de todos los expertos en cuanto a las preguntas generales sobre las guías de apoyo, pudimos determinar las siguientes consideraciones respecto a la estructura de las guías y a la presentación del proceso de desarrollo.

Estructura de las guías:

De los seis expertos encuestados, cuatro otorgaron la calificación máxima, mientras que los restantes dos otorgaron una calificación de 4 en relación a la estructura de las guías. Esto demuestra un alto grado de satisfacción con respecto a cómo están organizadas las secciones y su orden.

Presentación del proceso de desarrollo:

Tres expertos calificaron la presentación del proceso de desarrollo con un 5, y tres lo hicieron con un 4. Esto refleja una percepción muy buena sobre la sección donde se define la aplicación de la técnica.

6.1.2. Análisis de preguntas sobre consideraciones específicas de las guías

Como establecimos anteriormente, a cada uno de los grupos de expertos formados, los cuales pueden apreciarse en la Tabla 2, no solamente le hicimos preguntas relacionadas a aspectos generales de las guías de apoyo, sino también sobre aspectos más específicos.

Comenzando con el primer grupo, conformado por Mario Camerota y Andrés Bentos, nos pareció de gran importancia conocer respuestas a las preguntas que figuran en la Tabla 4. De las mismas realizamos el siguiente análisis.

Guía de Pruebas AB: En cuanto a la relevancia de la guía de Pruebas AB para emprendimientos de software, Mario Camerota otorgó una puntuación de 4, mientras que Andrés Bentos otorgó un 3. Aunque estas puntuaciones en promedio son ligeramente positivas, están por debajo de las puntuaciones de otros aspectos, lo que podría indicar que en sus experiencias personales como emprendedores, esta técnica no fue de las más relevantes.

Guía de Encuestas: Ambos participantes calificaron la relevancia de la guía de encuestas con puntuaciones altas, otorgando una valoración de 5. Sin embargo, Mario Camerota calificó la completitud y claridad de la guía de encuestas con un 3 y un 4 respectivamente, mientras que Andrés Bentos otorgó puntuaciones de 5 y 4. Esto podría indicar que, aunque la guía de encuestas es relevante, puede haber margen de mejora en su completitud y claridad.

Guía de Mockups: La relevancia de la guía de mockups para los emprendimientos de software obtuvo una puntuación de 3 de Mario Camerota y un 5 de Andrés Bentos. Además, Mario Camerota otorgó una puntuación de 2 a la completitud de esta guía, y Andrés Bentos otorgó un 4. Esto puede indicar que la guía de mockups puede no ser tan relevante o completa para estos encuestados en particular.

Guía de CI/CD: Ambos participantes otorgaron puntuaciones de 2 y 4 a la relevancia de la guía de CI/CD. Además, Mario Camerota y Andrés Bentos otorgaron puntuaciones de 4 y 4 a la completitud y claridad de esta guía. Esto sugiere que la guía de CI/CD en general es completa y clara para los participantes, en cuanto a la relevancia se ven resultados dispares.

En cuanto al segundo grupo, conformado por Darío Macchi y Santiago Ingold, en esta oportunidad nos interesó conocer respuestas a las preguntas que figuran en la Tabla 5. De las mismas realizamos el siguiente análisis.

Guía de Pruebas de Usabilidad: Respecto a la relevancia de la guía de Pruebas de Usabilidad para emprendimientos de software, Darío Macchi otorgó una puntuación de 5, mientras que Santiago Ingold otorgó un 3. Esto sugiere que puede haber cierta variación en la percepción de la relevancia de esta guía.

Guía de Entrevistas: Ambos participantes calificaron la relevancia de la guía de Entrevistas para emprendimientos de software con puntuaciones altas, con un 5 de Darío Macchi y un 4 de Santiago Ingold. Ambos calificaron la completitud y claridad de la guía de Entrevistas con un 4 y un 5 respectivamente, lo que nos indica que esta guía particular está muy bien en ese aspecto.

Guía de Prototipos: La relevancia de la guía de Prototipos para los emprendimientos de software obtuvo una puntuación de 5 de Darío Macchi y un 4 de Santiago Ingold. Además, Darío Macchi otorgó una puntuación de 5 a la completitud de esta guía, y Santiago Ingold otorgó un 4. Esto indica que los encuestados tienen una opinión muy buena sobre esta guía.

Guía de Observación: En cuanto a la relevancia de la guía de Observación, ambos participantes otorgaron puntuaciones de 5 (Darío Macchi) y 4 (Santiago Ingold). Sin embargo, para la completitud y claridad de esta guía, Darío Macchi otorgó puntuaciones de 4 y 5, y Santiago Ingold otorgó un 4 en ambas. Esto sugiere que la guía de Observación es relevante para los encuestados, y al interactuar con la guía la encontraron completa y clara.

Finalmente, en cuanto al tercer grupo, conformado por Pablo Benitez y Rodrigo Stratta, en esta oportunidad nos interesó conocer respuestas a las preguntas que figuran en la Tabla 6. De las mismas realizamos el siguiente análisis.

Guía de Estudio de Casos: En cuanto a la relevancia de la guía de Estudio de Casos para emprendimientos de software, Pablo Benítez otorgó una puntuación de 3, mientras que

Rodrigo Stratta otorgó un 4. Aunque estas puntuaciones en promedio son apenas positivas, están por debajo de las puntuaciones de otros aspectos.

Guía de Wireframes: Ambos participantes calificaron la relevancia, completitud y claridad de la guía de Wireframes para emprendimientos de software con puntuaciones altas, con un 5 de Pablo Benítez y un 4 de Rodrigo Stratta. Esto sugiere que la guía de Wireframes es relevante y aplicable para estos emprendimientos.

Guía de Pruebas Unitarias: Aunque Pablo Benítez otorgó una puntuación máxima de 5 a la relevancia, completitud y claridad de la guía de Pruebas Unitarias, Rodrigo Stratta otorgó una puntuación de 3 a la relevancia y 4 a la completitud y claridad. Esto puede indicar que, aunque la guía de Pruebas Unitarias puede ser completa y clara para algunos usuarios, puede no serlo para todos, y puede no ser tan relevante para algunos de nuestros encuestados.

Guía de Manejo de Deuda Técnica: Ambos participantes otorgaron puntuaciones de 4 y 5 a la relevancia, completitud y claridad de la guía de Manejo de Deuda Técnica. Esto sugiere que la guía de Manejo de Deuda Técnica es relevante, completa y clara para los participantes.

6.2. Análisis de entrevistas de validación de guías de apoyo por expertos

Como se detalla en la sección 5.3.2, a partir de las respuestas recabadas en las encuestas a expertos, optamos por realizar entrevistas para profundizar en aquellas respuestas que por su potencial, podrían proporcionar mayor información. Esto nos permitirá entender las razones detrás de cada respuesta, ya sea para esclarecer las justificaciones de opiniones negativas y positivas, o para formular preguntas adicionales si fuese necesario.

De las entrevistas a cada uno de los expertos, las cuales pueden verse sus transcripciones en el Anexo 17, realizamos un análisis y pudimos determinar los comentarios más destacados presentes a continuación:

Análisis de entrevista a Andrés Bentos

Puntos favorables:

La visión de Andrés hacia las guías es en general positiva. Valora la claridad y la forma en la que están escritas, mencionando que son fácilmente entendibles y van paso a paso. Además, subraya la complementariedad de las técnicas mencionadas. Esto refuerza la utilidad del compendio en su conjunto. En cuanto al aporte de estas guías, Andrés resalta que siempre hay aspectos positivos en cualquier cosa que se haga, ya que las interpretaciones pueden variar entre las personas. Considera que consolidar información de manera práctica, concisa y clara, especialmente con un adecuado toque académico, es valioso, sobre todo cuando se trata de organizar equipos o tomar decisiones sobre la construcción de un producto o servicio.

Observaciones y oportunidades de mejora:

Andrés enfatiza la necesidad de ser cautelosos al aplicar pruebas A/B, especialmente en emprendimientos de software que pueden no estar lo suficientemente avanzados. Argumenta que tales pruebas pueden ser costosas y que el riesgo asociado al implementar una funcionalidad que podría no ser bien recibida por el público es significativo. Esta perspectiva sugiere que, si bien la técnica es útil, su aplicación debe ser cuidadosamente considerada. En cuanto a la accesibilidad de las guías, Andrés propone incluir información adicional, como un glosario o terminología que facilite la comprensión a personas no técnicas. Esta idea viene respaldada por su observación sobre emprendedores sin un socio tecnológico, quienes podrían beneficiarse enormemente de una guía más inclusiva. Finalmente, Andrés reflexiona sobre la importancia de que cualquier aporte o guía, sirva de soporte y no entorpezca el proceso de emprendimiento. Esto enfatiza la importancia de ser prácticos y directos, sin perder de vista que el objetivo principal es entender y satisfacer las necesidades del cliente.

Análisis de entrevista a Darío Macchi

Puntos favorables:

Darío manifestó una alta valoración sobre la estructura uniforme que caracteriza a las guías. Esta cohesión entre las guías de apoyo no solo facilita la navegación y referencia a distintas secciones, sino que, como él mismo lo mencionó, brinda un marco de referencia a quienes, en contextos académicos o emprendedores, necesitan guiar a equipos hacia un objetivo, como en su caso particular guiando estudiantes desde su rol de tutor. Además, la legibilidad y simplicidad con la que se presenta la información no solo optimiza la experiencia de lectura, sino que también permite un rápido consumo del contenido, haciendo a las guías herramientas efectivas y eficientes.

Observaciones y oportunidades de mejora:

A pesar de la percepción generalmente positiva que tiene Darío de las guías, hizo hincapié en la necesidad de incorporar referencias más puntuales y detalladas, ya que actualmente referenciamos la bibliografía en general y no tanto a la página puntual ni una correspondencia directa. Estas referencias servirían para expandir y profundizar en temas que, por su naturaleza o extensión, no se abordan completamente en las guías, evitando así que el lector tenga que acudir a búsquedas adicionales. En adición a esto, Darío puso de manifiesto la relevancia de incluir técnicas adicionales, como el análisis de competidores. Dicha técnica no solo ofrecería una perspectiva valiosa sobre el panorama competitivo de un sector, sino que también proporcionaría herramientas prácticas, como la revisión de opiniones de usuarios en tiendas de aplicaciones, para entender mejor las demandas y percepciones del mercado.

Análisis de entrevista a Mario Camerota

Puntos favorables:

Mario resaltó la familiaridad que sintió con las guías, indicando que le parecieron valiosas debido a su formación y experiencia previa en la carrera de Licenciatura en Sistemas en Universidad ORT Uruguay. A pesar de que algunas áreas estuvieron detalladas más allá de lo que él mismo había considerado, reconoció el valor de esos detalles, describiendo las guías

como excelentes. En cuanto a las encuestas, Mario comprendió y validó su importancia, señalando que utilizaron esta técnica, ya que estaban conscientes de la necesidad que ellos pretendían atender, y que lo que buscaban era validar la aceptación de una herramienta para gestionar colectas.

Observaciones y Oportunidades de mejora:

A pesar de la apreciación generalmente positiva de las guías, Mario indicó que, para emprendimientos emergentes con recursos limitados, la inversión en ciertas áreas, como la automatización a través de CI/CD, puede no ser viable en las primeras etapas. Sugirió agregar un enfoque en la experiencia del usuario (UX) dado que la retención y conversión de usuarios es crítica para un emprendimiento de software. Un punto importante que Mario mencionó es la posibilidad de clasificar las guías según las etapas del emprendimiento, para que los emprendedores puedan identificar fácilmente cuáles son las más relevantes para ellos en un momento dado. Esta clasificación ayudaría a los emprendedores a no sentirse abrumados con la cantidad de información y a seleccionar herramientas más adecuadas a su contexto y necesidades. Por último, recalcó la relevancia de introducir estas técnicas de ingeniería de software en incubadoras o espacios similares, ya que muchos emprendedores pueden no estar familiarizados con ellas y podrían beneficiarse enormemente de su aplicación.

Análisis de entrevista a Pablo Benítez

Puntos favorables:

El entrevistado destacó la consistencia de la estructura de las guías, lo que facilita su comprensión y seguimiento. Asimismo, notó que ciertas guías incorporan recomendaciones de herramientas, lo que considera una ventaja. Apreció la claridad en la presentación de cada guía, especialmente en las secciones de introducción y proceso, reconociendo la utilidad de tener definidos los posibles usos y la secuencia de pasos a seguir. También mencionó que la guía del estudio de casos, aunque no le resultó novedosa, le pareció interesante su aplicación a la ingeniería de software, ya que generalmente la había visto relacionada a temas de negocio.

Observaciones y oportunidades de mejora:

El entrevistado señaló que algunas guías carecían de recomendaciones de herramientas y consideró que esto podría facilitar aún más la implementación de las técnicas. Sugirió, por ejemplo, la inclusión de herramientas para la guía de Wireframes. Aunque valoró la estructura y contenido de las guías, señaló que el proceso del estudio de casos le pareció complejo y se cuestionó sobre el tiempo que podría llevar su aplicación completa. Otro punto para mejorar sería la inclusión de números de página para facilitar la navegación y referencia. Además, destacó que las guías están orientadas a un público técnico, lo que podría limitar su utilidad para personas con menos experiencia o conocimientos en el área. Por último, propuso la incorporación de nuevas guías, como una relacionada con sistemas de control de versiones no sólo para código sino también para documentación, y otra sobre procesos de desarrollo.

Análisis de entrevista a Rodrigo Stratta

Puntos favorables:

La estructura y completitud de las guías fueron bien recibidas por Rodrigo, quien las encontró claras y fáciles de leer. También resaltó la utilidad del proyecto, considerándolo un valioso recurso para quienes inician y desconocen las técnicas de ingeniería de software disponibles. Además, mostró interés en revisar más guías, y se ofreció a brindar retroalimentación adicional una vez que las haya examinado.

Observaciones y oportunidades de mejora:

Rodrigo sugiere agregar información sobre la automatización de las pruebas de QA, dado que eso les ha sido útil en su experiencia. También mencionó que al inicio de su proyecto tuvieron un cuello de botella financiero, sugiriendo que podría ser útil incluir alguna herramienta o guía financiera, en especial para proyectos que manejan dinero. Por último, hizo notar que hay técnicas con las que no está familiarizado, como por ejemplo wireframes y otras que no ha aplicado, siendo una de ellas las pruebas A/B, lo que podría indicar una oportunidad para hacer estas técnicas más accesibles o brindar más información sobre su relevancia.

6.3. Análisis de encuestas de validación de guías de apoyo en retrospectiva

A continuación, figura el análisis realizado a los resultados obtenidos de este tipo de validación para cada una de las guías en las cuales tuvimos respuesta.

Guía de apoyo sobre pruebas unitarias

Como se aprecia en la Tabla 15, los resultados de la encuesta revelan que los procesos definidos en la guía de pruebas unitarias tuvieron la valoración máxima en cuanto a similitud a los procesos aplicados por todos los encuestados que respondieron esta pregunta. En lo que respecta a los desafíos enfrentados en la implementación de pruebas unitarias, las respuestas de los participantes fueron heterogéneas. Mientras un individuo señaló no haber encontrado dificultades, atribuible quizás a la experiencia previa del equipo, otro indicó un desafío técnico específico relacionado con la prueba de componentes de Inteligencia Artificial.

La encuesta también arrojó recomendaciones valiosas para mejorar la guía, como por ejemplo la inclusión de estrategias adicionales de pruebas como TDD (*Test driven development*) y BDD (*Behavior driven development*). Uno de los participantes incluso reafirmó la utilidad y completitud de la guía, lo que fortalece su relevancia y aplicabilidad en su estado actual. Si bien uno de los participantes no proporcionó comentarios adicionales, el otro destacó la importancia crítica de implementar pruebas unitarias para asegurar la calidad en cualquier proyecto de desarrollo de software.

Guía de apoyo sobre estudio de casos

Según las respuestas presentes en la Tabla 14, en términos de la similaridad entre la implementación del equipo y la guía, se registró una única respuesta con una calificación de 4 sobre 5. Esta alta puntuación señala una buena alineación entre la guía y el enfoque de implementación del equipo. En lo que se refiere a los desafíos experimentados, la dificultad en la obtención de información específica fue identificada como una barrera.

La única sugerencia de mejora para la guía se relaciona directamente con este desafío, recomendando que se aborde el tema de la dificultad para encontrar información en casos particulares. Este aspecto se intenta abarcar en la sección ‘Proceso de desarrollo’, específicamente en la planificación de la recopilación de datos de nuestra guía de ayuda de estudio de casos.

Guía de apoyo sobre prototipo

En la Tabla 16, podemos ver que, en relación con la técnica de desarrollo de Prototipos, la encuesta revela un alto grado de alineación entre la guía proporcionada y las prácticas implementadas en los proyectos, con calificaciones de 4 y 5 sobre 5. Esto sugiere que la guía es en general efectiva y aplicable en contextos prácticos.

En el ámbito de los desafíos, se identificaron dos puntos. Primero, la dificultad de definir el prototipo en función de las expectativas o requerimientos del cliente. Segundo, el reto de determinar el alcance adecuado del prototipo, evitando la sobre ingeniería o la inclusión de más funcionalidades de las estrictamente necesarias para el propósito del prototipo. Ambos desafíos apuntan a la necesidad de ofrecer lineamientos más precisos en la guía sobre la toma de decisiones en función del cliente y el alcance del proyecto. Con respecto a esto, nuestras guías de apoyo están pensadas para tener un enfoque más general, por lo que estos casos particulares resultan ser de dificultoso abordaje.

En cuanto a las sugerencias para la mejora de la guía, se planteó la importancia de ofrecer orientación sobre qué tipo de prototipo es el más adecuado en función de las necesidades específicas del cliente. Este punto puede estar directamente relacionado con los desafíos mencionados y podría constituir un valioso añadido a futuras ediciones de la guía.

Respecto a los comentarios adicionales, aunque uno de los participantes no ofreció comentarios, el otro extendió el valor de la técnica de prototipado más allá de la mera validación de productos. Destacó su utilidad para entregar valor de forma más rápida en el contexto de proyectos de gran envergadura, lo que podría ser un aspecto a considerar para enriquecer la guía con una perspectiva más amplia sobre la versatilidad de esta técnica.

Guía de apoyo sobre mockups

Según la Tabla 17, la única respuesta recibida para la pregunta sobre la similitud entre la guía y la implementación en el proyecto dio una calificación de 5 sobre 5.

El participante comentó que esta técnica no solo resultó útil para definir los requerimientos del proyecto, sino que también sirvió como una herramienta de comunicación efectiva para interactuar con personas externas al equipo de desarrollo. El participante recomienda rotundamente el uso de Mockups, lo que subraya la utilidad de esta técnica en el ámbito del desarrollo de proyectos.

Guía de apoyo sobre entrevistas

Como podemos ver en la Tabla 18 en lo que respecta a la similitud entre lo dispuesto en la guía y la implementación en sus proyectos, ambas respuestas otorgaron una puntuación de 5 sobre 5.

Uno de los desafíos más prominentes fue encontrar personas adecuadas para entrevistar, una barrera que ambos participantes experimentaron. Además, el manejo del tiempo se destacó como otro desafío, particularmente cuando se trata de sincronizar horarios con expertos en el área.

En cuanto a las sugerencias para mejorar la guía, una destacaba la importancia de documentar no sólo requisitos específicos sino también conceptos generales que podrían ser útiles más adelante para identificar casos de uso y requisitos funcionales. El segundo participante hizo hincapié en la necesidad de un enfoque más abierto durante las entrevistas, permitiendo que los entrevistados se expresen de manera más abierta y cómoda. Estos dos aspectos están contemplados en esta guía de apoyo.

Guía de apoyo sobre observación

En cuanto a los resultados obtenidos que figuran en la Tabla 19, obtuvimos una puntuación de 5 sobre 5 en el aspecto de similitud entre procesos. Sin embargo, es relevante notar que no

se proporcionaron respuestas a las preguntas sobre desafíos, sugerencias para mejorar la guía o comentarios adicionales.

Dicho esto, la falta de retroalimentación más detallada limita nuestra capacidad para realizar análisis más profundos o sugerir mejoras. Aunque el alto grado de similitud es un resultado positivo, la ausencia de detalles adicionales hace difícil evaluar áreas de mejora específicas para la guía en lo que respecta a esta técnica.

Guía de apoyo sobre encuestas

Según la Tabla 20, en relación con la técnica de encuestas, los datos recopilados revelan una alta coincidencia entre la guía proporcionada y cómo se implementaron las encuestas en los proyectos, con calificaciones de 4 y 5 sobre 5.

En cuanto a los desafíos, surgieron principalmente dos áreas de preocupación. La primera es la dificultad de difundir la encuesta de manera efectiva para llegar a un número suficiente de participantes, lo que es crucial para la representatividad de los datos. La segunda área de preocupación es asegurar que los participantes tomen en serio la encuesta y proporcionen respuestas de alta calidad. Ambos desafíos han sido abordados en nuestra guía de apoyo, pero no en tanta profundidad debido a que estos desafíos dependen del contexto del cual se quiera obtener información.

En relación con las propuestas de mejora para la guía, se resaltó la necesidad de especificar técnicas para prevenir sesgos al formular preguntas en encuestas. Aunque nuestra guía de apoyo señala la importancia de evitar dichos sesgos, no proporciona métodos concretos para lograrlo. A pesar de ser un aspecto que potenciaría la calidad de los resultados, decidimos no incluirlo en esta versión con el objetivo de mantener la guía concisa y de estructura clara

En el apartado de comentarios adicionales, aunque uno de los participantes no ofreció observaciones, el otro resaltó que las encuestas no son la herramienta adecuada para todos los contextos y que hay que saber cuándo y cómo implementarlas de manera efectiva. Este aspecto es tratado en la guía de apoyo correspondiente a la técnica.

Guía de apoyo sobre manejo de deuda técnica

Como se visualiza en la Tabla 21**,** en relación con la técnica de manejo de deuda técnica, las respuestas a la encuesta indican un nivel de alineación entre medio y bueno, entre la guía proporcionada y la implementación en los proyectos, con calificaciones de 3 y 4 sobre 5.

En el contexto de los desafíos, el principal obstáculo identificado es la dificultad de equilibrar el tiempo dedicado a resolver la deuda técnica con otras prioridades del proyecto. Este desafío resalta la complejidad inherente al manejo de deuda técnica, que a menudo está en conflicto con las presiones de tiempo y los objetivos a corto plazo.

Con respecto a las sugerencias para mejorar la guía, se plantearon dos puntos clave. El primero es la importancia de documentar la deuda técnica incluso cuando no se pueda resolver de inmediato, lo cual podría ser una adición útil en futuras ediciones de la guía. El segundo punto desafía la noción de que la deuda técnica afecta directamente al usuario, argumentando que tal impacto sería más apropiadamente clasificado como un *bug*.

En el ámbito de comentarios adicionales, se destacó que, en algunos casos, se asume deuda técnica de manera consciente para cumplir con plazos específicos, lo que resalta la complejidad de la toma de decisiones en esta área.

Guía de apoyo sobre integración y despliegue continuo

Por último, referido a la Tabla 22, en relación con la técnica de CI/CD (Integración Continua/Despliegue Continuo), las respuestas a la encuesta muestran una fuerte alineación entre las prácticas descritas en la guía y la implementación real en los proyectos, con calificaciones de 4 y 5 sobre 5.

En cuanto a los desafíos, se identificaron dos áreas específicas. La primera es la complejidad asociada con la automatización de pruebas, ya que en su caso utilizan una herramienta de terceros que cobra por tráfico, por lo que incluirla en su pipeline de CI/CD aumenta sustancialmente los costos. El segundo desafío se centra en la configuración inicial,

señalando que, una vez superado este obstáculo, el mantenimiento y operación del sistema de CI/CD se vuelve considerablemente más manejable.

Respecto a las sugerencias para mejorar la guía, se percibe un alto grado de satisfacción con el contenido actual.

6.4. Análisis de guías de apoyo a través del uso práctico

La entrevista al equipo de proyecto proporciona una validación significativa sobre la calidad y aplicabilidad de las guías de apoyo de mockups y prototipos. A pesar de su familiaridad con estas técnicas, las guías sirvieron no sólo como un refuerzo sino también como una herramienta que complementó y enriqueció su proceso.

El equipo trabajó codo a codo con el cliente en la realización de los mockups, en el cual ambas partes quedaron satisfechas con el resultado obtenido, lo que es un testimonio del valor práctico de las guías. Este nivel de integración y adaptabilidad muestra que las guías no son solamente teóricas, sino altamente prácticas y aplicables en situaciones reales de proyectos.

Otro punto destacado es la apreciación del equipo por la flexibilidad de las guías. Consideran que, mientras que las guías proporcionan una dirección y estructura claras, también dejan espacio para la adaptabilidad y personalización según las necesidades específicas del proyecto. Esta flexibilidad es esencial, ya que cada proyecto es único y requiere un enfoque adaptado a sus particularidades.

El hecho de que el equipo, teniendo experiencia previa, aún encontrara valor en las guías es muy significativo. Sugerir que las guías serían aún más enriquecedoras para alguien que no tiene experiencia previa con dicha técnica es una fuerte validación de su claridad, accesibilidad y utilidad.

Por último, aunque el equipo haya aplicado solamente dos guías de apoyo, el simple hecho de que consideren la aplicación de más guías y planeen su posible implementación en el futuro, habla del valor percibido y la confianza en las guías proporcionadas.

6.5. Respuesta a preguntas de investigación

1. **¿Cuáles son las técnicas de ingeniería de software más utilizadas en emprendimientos de software?**

La respuesta a esta pregunta ha sido construida a través de un proceso riguroso y detallado delineado en la fase 1 de nuestra investigación. En primer lugar, preparamos y ordenamos los artículos académicos para poder manejar este gran volumen de información, lo cual detallamos en la sección 5.1.1. Posteriormente, en la sección 5.1.2, analizamos utilizando la técnica de revisiones rápidas a todos los artículos, lo que nos permitió obtener una imagen clara de qué técnicas son utilizadas por los emprendimientos.

Una vez recopilada esta información, procedimos a validar su completitud, como se describe en la sección 5.1.3. Fue esencial para nosotros garantizar que la información recopilada fuera representativa. Además, llevamos a cabo una revisión cruzada de los artículos académicos, la cual puede verse en la sección 5.1.4.

Luego, en la sección 5.1.5, agrupamos las técnicas de ingeniería de software, seguido por un proceso en el cual contamos las ocurrencias de cada una de ellas. Esto puede apreciarse en la sección 5.1.6. Esta fase nos proporcionó una comprensión de las técnicas más prevalentes y su relevancia en el panorama de los emprendimientos de software.

Finalmente, después de definir parámetros específicos para la selección, el cual puede verse en la sección 5.1.7, llegamos a una lista final de técnicas de ingeniería de software que consideramos de suma importancia para los emprendimientos de software. Estas técnicas, presentadas en la sección 5.1.8, son:

- Encuestas
- Entrevistas
- Estudio de casos
- Integración y Despliegue Continuo

- Manejo de deuda técnica
- Mockups
- Observación
- Pruebas A/B
- Prueba de usabilidad
- Prototipos
- Pruebas Unitarias
- Wireframes

Consideramos que la aplicación de estas técnicas seleccionadas en un emprendimiento puede ofrecer ventajas significativas en términos de eficiencia y efectividad para emprendimientos de software.

2. **¿Qué estructura deberían tener las guías de apoyo sobre las técnicas de ingeniería de software más utilizadas?**

Según definimos en el diseño metodológico, específicamente en la sección 4.4.1 y que luego llevamos a cabo obteniendo como resultado lo dispuesto en la sección 5.2.1, donde definimos la estructura de las guías de la siguiente manera:

- **Introducción**: Esta ofrece una panorámica inicial, sentando las bases para las secciones posteriores. Asimismo, puede ayudar a captar la atención y el interés del lector desde el inicio.
- **¿Por qué debería incluirla en mi emprendimiento de software?**: Estos emprendimientos operan con recursos limitados y a menudo deben priorizar sus acciones. Esta sección responde a la pregunta fundamental: "¿Por qué debería invertir tiempo, esfuerzo y recursos en esto?". La idea es mostrar el valor que les puede aportar a sus procesos aplicar esta técnica.

- **¿En qué condiciones puedo aplicar esta técnica?:** Define el contexto o las circunstancias ideales para aplicar la técnica. Sirve para ayudar al lector a identificar si su situación particular se alinea con la aplicación efectiva de lo que se está presentando.

- **Proceso de desarrollo**: Es fundamental brindar un camino claro y estructurado para que el lector pueda poner en práctica lo que se describe en la guía. Esta sección garantiza que el lector tenga una hoja de ruta paso a paso para implementar de manera efectiva la técnica.

- **Consideraciones para aplicar la técnica**: Mientras que el proceso de desarrollo ofrece los pasos a seguir, las consideraciones ofrecen una guía de qué no hacer o aspectos a tener en cuenta durante la implementación. Los emprendimientos de software, al ser entidades en crecimiento y aprendizaje, se beneficiarán enormemente de estas consideraciones, evitando errores comunes y garantizando una aplicación más eficiente de la técnica.

- **Herramientas Sugeridas**: Esta sección es opcional y sugiere algunas herramientas que ayuden a la aplicación de la técnica en contextos particulares.

- **Dónde leer más**: Proporcionar fuentes y referencias es crucial para legitimar la información presentada y ofrecer a los emprendimientos de software la oportunidad de profundizar en temas específicos, lo que puede fortalecer la confianza del lector en el contenido.

A partir del análisis derivado de las encuestas y entrevistas realizadas al grupo de expertos, así como las encuestas realizadas a emprendedores de forma retrospectiva y conclusiones recolectadas del uso práctico, se identificaron diversos puntos cruciales que respaldan la estructura de las guías de apoyo sobre técnicas de ingeniería de software.

Los expertos han mostrado un alto grado de satisfacción con la estructura actual de las guías. De hecho, una mayoría otorgó la calificación máxima y el resto una buena valoración, lo que refleja un considerable respaldo a la organización y secuencia de las secciones. Esta cohesión

y uniformidad en el formato ofrecen a los usuarios un patrón predecible al interactuar con ellas.

La claridad de las guías fueron aspectos subrayados. La fácil comprensión y el desglose paso a paso del contenido son cruciales para la eficacia de las guías.

Varios expertos sugirieron la inclusión de recomendaciones específicas de herramientas. Estas recomendaciones pueden ser una ventaja, brindando a los usuarios no solo el conocimiento teórico sino también soluciones prácticas para implementar dichas técnicas.

Para garantizar la accesibilidad del contenido a una audiencia más amplia, es importante incorporar elementos que faciliten la comprensión a personas no técnicas, tales como glosarios. Para esta instancia no lo consideramos, pues nuestro público objetivo fueron personas con las cualidades técnicas necesarias, pero podría ser una buena adición para quienes continúen este trabajo en el futuro.

La incorporación de referencias detalladas fue otro punto enfatizado. Estas referencias permitirían a los lectores profundizar en temas específicos si lo desean, agregando profundidad al contenido sin sobrecargar la guía principal.

Una propuesta interesante fue clasificar las guías según etapas o fases específicas del proceso de desarrollo o emprendimiento. Esto podría ayudar a los usuarios a seleccionar la información más relevante según su situación actual.

3. **¿Qué nivel de satisfacción reportan quienes interactuaron con las guías respecto al contenido generado?**

Dadas las técnicas de ingeniería de software determinadas anteriormente, y también dada la estructura definida para las guías, respaldados tanto en artículos académicos como bibliografía de prestigio, generamos a cada una de las guías, de las que posteriormente recibimos retroalimentación valiosa de muchos interesados.

Basándonos en la retroalimentación obtenida de expertos en el campo y en comentarios retrospectivos, estamos en una posición privilegiada para afirmar con confianza, la riqueza y robustez del contenido de nuestras guías de apoyo sobre técnicas de ingeniería de software.

Varios de nuestros revisores han señalado la familiaridad y consistencia en la estructura de las guías, facilitando su seguimiento y comprensión. Esta familiaridad permite a los usuarios sentirse en sintonía con el contenido presentado. Algunos han mencionado la apreciación de los detalles adicionales proporcionados en áreas específicas, que a pesar de ir más allá de lo comúnmente considerado, enriquecen notablemente la experiencia de aprendizaje.

Un aspecto bien valorado ha sido la claridad en la presentación, especialmente en las secciones introductorias y en la exposición del proceso a seguir. Este nivel de detalle es esencial para guiar a los usuarios a través de la secuencia adecuada de pasos, y ha sido destacado por su utilidad, especialmente para aquellos que están iniciándose en la ingeniería de software.

Otro comentario recurrente ha sido que sería de importancia marcar a las guías según la etapa del emprendimiento en que puedan usarse. Esta observación trata sobre la necesidad de introducir estas técnicas en incubadoras, maximizando su impacto al abordar a emprendedores que podrían no estar familiarizados con ellas. Este comentario es de gran importancia pues una de las motivaciones principales detrás de este proyecto radica en la usual carencia en las incubadoras de apoyo en cuanto a orientación específica en ingeniería de software.

Es necesario mencionar que, si bien las guías están orientadas principalmente a un público técnico, varias de las personas con las que hemos interactuado para la validación de estas han dejado constancia de que su diseño y estructura permiten una rápida lectura y comprensión. La cohesión entre las guías, junto con una navegación fácil y una presentación legible y simplificada, optimiza el tiempo de lectura, haciendo que el contenido sea efectivo y eficiente. Otra de las sugerencias es incluir referencias bibliográficas más específicas en lugar de generalidades sobre las fuentes consultadas es algo que definitivamente consideraremos.

Al mirar hacia el futuro, reconocemos la importancia de seguir refinando, mejorando y ampliando el catálogo de guías de apoyo.

La retroalimentación ha sido tremendamente positiva, y nos complace saber que nuestras guías han tenido una recepción tan satisfactoria.

4. **¿Cómo se puede definir y especificar un proceso de mentoría sobre prácticas de ingeniería de software orientado a emprendimientos de software?**

El diseño del plan de mentoría que ideamos para orientar a emprendimientos de software, el cual puede verse en la sección 5.2.2.2, consideramos que es especialmente adecuado para su propósito, y esto se puede atribuir a varios factores del plan. Primero, se basa en un enfoque de investigación robusto, pues las técnicas de ingeniería de software propuestas se derivan de un gran trabajo de relevamiento, así como de validación con expertos y egresados de proyectos de emprendimiento que pudieron hacerlo en retrospectiva. Este enfoque garantiza que las técnicas seleccionadas no solo son relevantes, sino que también han demostrado ser efectivas.

La definición clara de roles y expectativas es otro elemento fundamental. Al especificar lo que se espera tanto del equipo de trabajo como del mentor, aseguramos un nivel de compromiso y entendimiento mutuo. Esto alinea los esfuerzos de todas las partes involucradas hacia objetivos comunes, optimizando los resultados de la mentoría.

Una característica importante del plan es el enfoque en la mejora continua. Al priorizar la retroalimentación continua y la adaptabilidad, garantizamos que el plan de mentoría sea relevante, evolucionando a medida que cambian las necesidades y circunstancias del emprendimiento. Esta adaptabilidad se refleja también en la flexibilidad logística que ofrece el plan, considerando diversas modalidades de reunión que se ajustan a las realidades cambiantes de los emprendimientos en el mundo actual.

Además, este plan no solo se centra en la mentoría en sí, sino que se proyecta más allá. La inclusión de una etapa de evaluación y retroalimentación al finalizar el proceso de mentoría, seguida de la generación de informes y un proceso de mejora, refleja una visión a largo plazo. Reconocemos que la mentoría no es un evento único, sino un proceso dinámico y en constante evolución, con la expectativa de que pueda cumplir varios ciclos de aplicación y mejora en el futuro.

En esencia, este plan de mentoría ha sido diseñado específicamente para apoyar a emprendimientos de software en la implementación de técnicas de ingeniería que nuestro equipo ha identificado como valiosas. Su estructura y enfoque garantizan una mentoría

relevante, adaptativa y sostenida en el tiempo, siendo una herramienta muy importante para cualquier emprendimiento que busque optimizar sus procesos de software.

Si bien la intención inicial era durante la duración de nuestro proyecto, poder aplicar este proceso de mentoría con algunos de los equipos que están cursando en este momento su proyecto final de carrera del tipo emprendimiento, lamentablemente no pudimos llevarlo a cabo, por las razones que explicamos a continuación.

Los equipos con los que interactuamos presentaron distintas fases de desarrollo. Mientras algunos estaban en sus etapas iniciales, aún sin la necesidad de aplicar técnicas de ingeniería de software, otros ya tenían procesos más avanzados. Esta heterogeneidad representó un reto. A raíz de nuestras interacciones, percibimos una reducción en la disposición colaborativa de los equipos y una comunicación menos fluida. Consideramos que las diferencias entre las etapas de nuestro proyecto y las de dichos equipos pudieron influir en esta situación, complicando la ejecución de nuestro plan de mentoría debido al periodo limitado de nuestro proyecto académico.

Además, pensamos que hubo otros factores individuales que afectan a cada equipo, como la falta de disponibilidad. Intentamos alcanzar un entendimiento para identificar la causa principal del rompimiento en la comunicación, pero nos encontramos con la misma barrera, sin lograr esclarecer completamente la situación.

A pesar de estos desafíos, reconocemos el valor de cada interacción.

Aunque no logramos una implementación práctica de nuestro plan de mentoría, recibimos una respuesta positiva de los docentes a quienes se lo presentamos. Creemos firmemente en el potencial de este enfoque y, dada su naturaleza, aspiramos a que se aplique y refine en futuras instancias. Estamos convencidos de que, con el tiempo y la adaptación, tanto el plan como las guías seguirán evolucionando y resultarán valiosos para futuros emprendimientos de software.

7. Conclusiones

Esta investigación nos ha permitido profundizar sobre un tema de gran relevancia en el ámbito de la ingeniería de software. No solo nos adentramos en un área de nuestro interés, sino que también adquirimos un valioso conocimiento que enriqueció nuestra perspectiva profesional. Estamos convencidos de que los hallazgos y herramientas desarrolladas serán de gran aporte para emprendedores en el ámbito del software.

Durante nuestro estudio, buscamos respuestas a preguntas que permitieran comprender y fomentar la mejora en cuanto a los procesos de ingeniería de software de los emprendimientos de software en general. Inicialmente, nos enfocamos en encontrar las técnicas de ingeniería de software más utilizadas en tales emprendimientos. Posteriormente, nos propusimos definir la estructura base de las guías de apoyo destinadas a estas técnicas, con el objetivo de que resultaran prácticas, accesibles y relevantes para los emprendedores.

Una vez creadas las guías, nos propusimos medir el nivel de satisfacción de quienes interactuaron con ellas. Para garantizar que nuestras contribuciones fueran de la más alta calidad, sometimos nuestras guías de apoyo a la opinión de expertos, integrado por docentes de laboratorio de ingeniería de software, egresados que han llevado a cabo proyectos de investigación sobre emprendimientos de software, y emprendedores en el ámbito del software. Este tipo de validación fue la que más información nos ha dado. Adicionalmente a estos expertos, se realizó una validación en retrospectiva con egresados que han llevado a cabo emprendimientos y con estudiantes que, en ese momento, estaban llevando a cabo proyectos de final de carrera del tipo emprendedor. Esta retrospectiva se basó en comparar lo dispuesto en nuestras guías, con lo que los encuestados han aplicado en sus proyectos del tipo emprendedor. También se incluyó una validación en aplicación real de algunas de las guías propuestas por un equipo de estudiantes que estaba realizando su proyecto final en conjunto con la industria. Todas las personas vinculadas a la validación de todo nuestro trabajo están vinculadas a la Universidad ORT Uruguay.

En concreto, hemos generado un compendio de guías de apoyo que desglosan las 12 técnicas de ingeniería de software más usadas para emprendimientos de software. Estas guías no solo proporcionan detalles sobre cada técnica, sino que también abordan aspectos cruciales que

han sido altamente valorados por los usuarios. Entre las secciones más destacadas y apreciadas se encuentran por ejemplo un apartado sobre consideraciones para aplicar la técnica en particular, otra sección específica donde se manifiesta el proceso de desarrollo, un apartado dedicado a clarificar en qué condiciones se puede aplicar cada técnica y una reflexión sobre las ventajas de aplicar dichas esta técnica.

Ante la evidente necesidad de acompañar a los emprendedores de una manera estructurada en sus procesos de ingeniería de software, nos planteamos cómo establecer un plan de mentoría adaptado a dichos procesos de los emprendimientos de software. Esta reflexión nos llevó a diseñar un enfoque de mentoría robusto, dando más importancia a la transmisión efectiva de conocimientos y destrezas en ingeniería de software, a través de la utilización y entendimiento de las guías de apoyo. Estas indagaciones fundamentales no solo dieron forma a nuestra investigación, sino que también nos dotaron de una comprensión más rica sobre la temática.

En cuanto a este plan de mentoría, diseñado para orientar a emprendimientos de software, refleja un compromiso con los aspectos técnicos y la adaptabilidad, basándose en la colaboración continua. Su estructura, que se centra en roles definidos, retroalimentación, mejora constante y una visión a largo plazo, lo convierte en una herramienta potencialmente muy valiosa para emprendimientos que buscan fortalecer la aplicación de técnicas de ingeniería de software. A pesar de no haberse podido aplicar de manera práctica durante la duración del proyecto, el plan ha recibido respuestas positivas del ámbito académico. Estos desafíos, lejos de desalentar, han brindado perspectivas valiosas y reafirman la necesidad de adaptabilidad. Se espera que, a medida que se presenten nuevas oportunidades, este plan de mentoría evolucione y continúe beneficiando a futuros emprendimientos en el área de software.

8. Lecciones aprendidas

A lo largo de esta investigación relacionada a la ingeniería de software aplicada a emprendimientos de software, hemos extraído diversas lecciones que consideramos de gran relevancia para futuros estudios y proyectos en la materia.

Primero, esta carencia representa un problema significativo, ya que deja a los emprendimientos de software sin una guía adecuada en una de las áreas cruciales para su éxito: la ingeniería de software.

Esto nos incentivó a intentar colaborar en dicho aspecto mediante la creación de guías de apoyo, así como el diseño de un plan de mentoría que colabora a atender esta carencia.

La retroalimentación, como segundo punto, se presenta como una herramienta esencial. Las perspectivas ofrecidas por aquellos que interactuaron con nuestras guías fueron invaluables, arrojando luz sobre áreas de mejora y consolidación. Esto nos llevó a una tercera lección: la importancia de la diversidad de perspectivas. Al interactuar con docentes, estudiantes y emprendedores, comprendimos que cada punto de vista enriquece y complementa el producto final.

Una cuarta lección, pero no menos importante, fue comprender que la flexibilidad es la clave. Los emprendimientos de software operan en un entorno dinámico y cambiante, por lo que nuestras guías deberían ser un reflejo de esa realidad, es decir suficientemente adaptables, pero ofreciendo una dirección clara.

Sin embargo, como toda investigación, enfrentamos desafíos, en especial con el plan de mentoría. Uno de los principales problemas que encontramos fue la dificultad de hallar personas dispuestas a colaborar. Resultó ser especialmente frustrante encontrar quien respaldara la validación de las guías de apoyo y, aún más, aquellos dispuestos a involucrarse de lleno en el plan de mentoría. Creemos que la cantidad acotada de proyectos de emprendedurismo en curso actualmente dentro de la Universidad ORT Uruguay, pudo haber agravado esta situación. A pesar de estos contratiempos, es importante destacar que sí hubo algunas personas que generosamente ofrecieron su tiempo y experiencia para colaborar y validar nuestras guías. Estas valiosas contribuciones reforzaron nuestra convicción en la

importancia del proyecto. Estos obstáculos, lejos de desanimarnos, nos enseñaron la importancia de la resiliencia y adaptabilidad en el proceso investigativo.

Finalmente, aunque comenzamos con el objetivo de diseñar guías, terminamos con una comprensión más rica y profunda del panorama de la ingeniería de software en emprendimientos de software. Esta última lección reafirma que la verdadera esencia de la investigación no radica sólo en los resultados, sino también en el aprendizaje y descubrimiento que implica el proceso.

Este proyecto ha sido más que una serie de hallazgos. Ha sido un proceso de aprendizaje y crecimiento que nos ha equipado con lecciones que trascenderán este estudio y que nos servirán en futuras investigaciones y emprendimientos.

9. Amenazas a la validez del estudio

En estudios cualitativos, según Hernández Sampieri y colaboradores [33], es necesario reemplazar los típicos criterios de amenazas a la validez por aquellos más pertinentes a esta modalidad de investigación. Estos autores señalan que expertos en el campo han propuesto criterios que buscan una correspondencia con la confiabilidad, validez y objetividad de la investigación cuantitativa. Si bien estos criterios han sido reconocidos por muchos investigadores, otros los han descartado.

9.1. Dependencia

Se refiere al nivel en el cual diversos investigadores, al recopilar y analizar datos similares, obtienen resultados comparables. Según Saumure y Given [34], y Franklin y Ballau [35], existen dos tipos de dependencia: a) interna, que se relaciona con el grado en que, al menos dos investigadores, derivan categorías parecidas a partir de los mismos datos; y b) externa, que concierne al grado en que diferentes investigadores identifican categorías similares en un ambiente y periodo idénticos, pero recolectando datos propios.

Con respecto a la dependencia interna, experimentamos notoriamente este aspecto durante la instancia de revisión cruzada de artículos académicos, en donde con esta acción buscamos asegurarnos de que la información recolectada en primera instancia era la totalidad de información relevante disponible en dichos artículos. Luego de realizar esta revisión cruzada en los documentos seleccionados, llegamos a que los resultados podrían considerarse iguales, ya que los hallazgos encontrados en esta revisión fueron mínimos. En cuanto al criterio de dependencia externa no se pudo verificar, ya que no existen otros investigadores que hayan obtenido datos bajo las mismas condiciones, es decir, en el mismo entorno y periodo.

9.2. Transferencia

Este punto hace alusión a si el investigador ha comprendido en su totalidad y profundidad las experiencias de los participantes, especialmente las relacionadas con la formulación del problema. La validez en este aspecto se ha fortalecido de varias maneras. En primer lugar, la grabación de todas las entrevistas, a excepción de una debido a problemas técnicos. Estas

grabaciones proporcionaron una fuente fiable para revisar y reproducir las conversaciones en cualquier momento, asegurando no omitir ninguna información esencial y capturar la esencia completa de las respuestas de los entrevistados.

Siguiendo nuestra guía de entrevistas, el uso de preguntas abiertas permitió a los entrevistados expresarse con detalle, garantizando una mayor comprensión de sus experiencias y percepciones. Además, se emplearon técnicas de escucha activa, como el parafraseo y el resumen, para garantizar una completa comprensión de las respuestas del participante.

Conscientes de la posibilidad de sesgos, se hizo un esfuerzo para abordar las entrevistas con una mente abierta para mantener un enfoque equilibrado e imparcial durante todo el proceso.

9.3. Transferencia y aplicabilidad de resultados

Según Hernández Sampieri y colaboradores [33], este criterio no busca generalizar los resultados a una población extensa. En un estudio cualitativo, el objetivo no es esa generalización, sino que ciertos aspectos o la esencia de los hallazgos puedan ser relevantes o aplicables en diferentes contextos.

Con respecto a esto, tanto los expertos, como equipos de proyectos encuestados y entrevistados, pertenecen a Universidad ORT Uruguay, por lo que no está asegurada la validez de las conclusiones fuera de este ámbito.

10. Propuestas de trabajos futuros de investigación

Al finalizar esta investigación, emergen algunas cuestiones que sugieren la posibilidad de trabajos a futuro. Estas inquietudes, que podrían ampliar la comprensión sobre el tema, se presentan a continuación:

- Poner en práctica el plan de mentoría y debido a la naturaleza iterativa del proceso, documentar resultados, mejoras y conclusiones para mejorar futuras ejecuciones del plan.

- Diseñar e implementar un sistema de control de versiones para las guías de apoyo. Esto tendrá como beneficio que, si un documento se corrompe o se pierde, se podrá recuperar una versión anterior, así como acceder a justificaciones que motivaron determinados cambios.

- Incluir un glosario o terminología adicional en las guías para personas no técnicas, para ampliar el público objetivo.

- Analizar e incluir en el contenido de todas las guías de apoyo, herramientas sugeridas para la aplicación de estas.

- Si bien las guías de apoyo contienen una sección dedicada para especificar en qué condiciones puedo aplicar dicha técnica, es relevante se especifique claramente en qué etapa o etapas del proyecto agregan mayor valor.

- Relevar el nivel académico con el cual suelen llegar los estudiantes a la realización del proyecto final de su carrera, con la finalidad de determinar de forma ajustada que nivel de lenguaje técnico deben manejar las guías de apoyo.

- De las sugerencias y experiencias recolectadas durante el proyecto, se desprende la necesidad de generación de guías de apoyo relacionadas con la mejora en la experiencia de usuario y análisis de competidores, se recomienda validar dichas necesidades y consecuentemente realizar la generación.

11. Referencias bibliográficas

[1] N. Patel, “90% of Startups Fail: Here’s what you Need to Know about the 10%”, 2015. [Online]. Available: https://www.forbes.com/sites/neilpatel/2015/01/16/90-of-startups-will-fail-heres-what-you-need-to-know-about-the-10/?sh=3cee84366792. Accessed: Oct. 12, 2022

[2] T. Eisenmann, Ed., *Why Start-ups Fail,* USA: Currency, 2021.

[3] C. Giardino, M. Unterkalmsteiner, N. Patemoster, T. Gorschek, and P. Abrahamsson, “Software Development in Startup Companies: The Greenfield Startup Model,” *IEEE Transactions on Software Engineering,* vol. 42, pp. 586-604, 2016.

[4] M. Kajko-Mattsson and N. Nikitina, "From Knowing Nothing to Knowing a Little: Experiences Gained from Process Improvement in a Start-Up Company," *2008 International Conference on Computer Science and Software Engineering*, Wuhan, China, 2008, pp. 617-621.

[5] A. Buffa and D. Febles, “Habilidades técnicas y blandas en startups de software”, Tesis de grado, Universidad ORT Uruguay, Montevideo, 2019.

[6] S. Peralta and G. Perez, “Inicio y evolución de las prácticas y actividades de ingeniería en startups de software”, Tesis de grado, Universidad ORT Uruguay, Montevideo, 2019.

[7] A. S. Gutterman, *Entrepreneurship*, 1st ed. New York, NY, USA: Business Expert Press, 2018.

[8] T. Byers, R. Dorf and A. Nelson, *Technology ventures: from idea to enterprise*, 5th ed. New York, NY, USA: McGraw-Hill Education, 2018.

[9] N. Shanbhag and E. Pardede, “Development Frameworks for Software Startups: A Literature Review,” In E. Gupta (Ed.), *Emerging Technologies for Innovation Management in the Software Industry*, IGI Global, pp. 1-43, 2022.

[10] E. McGowan, “From Early to Acquired: What Are the Stages of a Startup?” [Online]. Available: https://www.startups.com/library/expert-advice/startup-stages, Accessed: Oct. 20, 2022.

[11] I. Sommerville, *Software Engineering*, 9th ed. Pearson, 2011.

[12] P. Bourque and R. Fairley, “Guide to the software engineering Body of Knowledge, V3.0” [Online]. Available: www.swebok.org. Accessed: Oct. 20, 2022.

[13] “ISO/IEC TR 19759:2005”. [Online]. Available: https://www.iso.org/standard/33897.html. Accessed: Oct. 20, 2022.

[14] S. Blank, M. Andreessen, R. Hoffman, and W. Sahlman, *HBR’s 10 Must Reads on Entrepreneurship and Startups*. Brighton, Massachusetts, USA: Harvard Business Review Press, 2018.

[15] E. Ries, *The Lean Startup: How Today’s Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses,* Crown Currency, 2011.

[16] M. Adey, F. Early, and M. Foster, *The mentors handbook*. London, UK: Herts Tech, 1999.

[17] C. Conway, *Strategies for Mentoring: A Blueprint for Successful Organizational Development*. Toronto, Canada: Wiley, 1998.

[18] S. Hezlett and S. Gibson, “Linking mentoring and social capital: Implications for career and organization development,” *Advances in Developing Human Resources,* 3rd ed., vol. 9, 2007.

[19] S. Jeong and S. Park, “Mentoring in the Human Resource Development Context”, in *The Wiley International Handbook of Mentoring,* B. Irby and et al, Eds., Hoboken, New Jersey, USA: 2020.

[20] J. Starr, *The Mentoring Manual: Your Step by Step Guide to Being a Better Mentor,* 2nd ed., Ft Pr, 2014.

[21] P. Phan and S. Mian, Eds., “Business Incubation and Incubator Mechanisms”, in *Technology entrepreneurship and business incubation: theory, practice, lessons learned*, London, UK: Imperial College Press, 2016.

[22] A. Yusubova and B. Clarysse, “Success Factors of Business Accelerators in Three European Cities: Paris, London, Berlin”, in *Technology entrepreneurship and business incubation: theory, practice, lessons learned*, P. Phan and et al, Eds., London, UK: Imperial College Press, 2016.

[23] N. Tripathi and M. Oivo, “The Roles of Incubators, Accelerators, Co-working Spaces, Mentors, and Events in the Startup Development Process”, in *Fundamentals of Software Startups*, A. Nguyen-Duc and et al, Eds., Berlin, Germany: Springer, 2020.

[24] R. Busulwa, N. Birdthistle, and S. Dunn, Eds., *Startup Accelerators. A field guide*, 1.a ed. Hoboken, New Jersey, New York, USA: Wiley, 2020.

[25] E. St-Jean and J. Audet, “The role of mentoring in the learning development of the novice entrepreneur”, in *The International Entrepreneurship and Management Journal*, vol. 8, D. Audretsh, G. Castrogiovanni, and S. Roig, Eds., Springer, 2012, pp. 119-140.

[26] J. Chrisman, E. McMullan, and J. Hall, “The influence of guided preparation on the long-term performance of new ventures”, in *Journal of Business Venturing*, vol. 20, 2005, pp. 769-791.

[27] R. Zozimo, S. Jack, and E. Hamilton, “Entrepreneurial learning from observing role models”, *in Entrepreneurship and Regional Development*, vol. 29, 2017, pp. 889-911.

[28] H. Travassos and M. Felderer, “Rapid Reviews in Software Engineering”, in *Contemporary Empirical Methods in Software Engineering*, B. Cartaxo, G. Pinto, and S. Soares, Eds., Switzerland: Springer Nature, 2020.

[29] T. Winters, T. Manshreck and H. Wright, Eds., *Software engineering at Google. Lessons learned from programming over time*. O’Reilly, 2020.

[30] L. Hatley, M. Al-Freih, and B. Bannan, “I’m Just Guiding You: An Exploration of Software Design Mentorship within a Software Engineering Firm”, in *Journal of Software Engineering and Applications*, vol. 11, 2018.

[31] N. Richter and T. Schildhauer, “Startup Clinics: Applied Research and ‘First Aid’ for Early Stage Startups”, in *Entrepreneurial Innovation and Leadership*, N. Richter and et al, Eds., Berlin, Germany: Springer, 2018.

[32] L. Philips-Jones, *The Mentors Guide - How to be a Kind of Mentor You Once Had or Wish You’d Had*. California, USA, 2003.

[33] R. Hernández Sampieri and C. Fernández Collado, Eds., *Metodología de la investigación*, 6th ed. Mexico: McGraw-Hill, 2014.

[34] L. Given and K. Saumure, Eds., *The SAGE Encyclopedia of Qualitative Research Methods.* California, USA: SAGE Publications, 2008.

[35] C. Franklin and M. Ballau, “Reliability and validity in qualitative research”, in *Social work: Research and evaluation. Quantitative and qualitative approaches*, R. Grinnell and Y. Unrau, Eds., New York, USA: Oxford University Press, 2005, pp. 438-449.

ANEXO 1 - Guía de encuestas

Introducción

Las encuestas son un medio de obtener información de muchas personas, de forma anónima y en un tiempo relativamente breve. Una encuesta es un conjunto de preguntas escritas que se formulen a las partes interesadas. Las encuestas pueden recoger información que luego se puede utilizar para mejorar un producto al aportar ideas y opiniones que no se obtienen fácilmente mediante otras técnicas de elicitación.

¿Por qué debería incluirla en mi emprendimiento de software?

El uso de encuestas como técnica de consulta tiene varias ventajas:

- Las encuestas que utilizan preguntas cerradas pueden proporcionar datos numéricos que pueden analizarse estadísticamente. Esto puede ayudar a identificar patrones y tendencias que podrían no ser evidentes a partir de datos cualitativos por sí solos. Por ejemplo, en una encuesta se puede pedir a los encuestados que valoren su satisfacción con un producto o servicio en una escala del 1 al 10. Los datos numéricos resultantes pueden analizarse en función de la satisfacción de los encuestados. Los datos resultantes pueden analizarse para identificar áreas en las que se necesitan mejoras.

- Las encuestas con preguntas abiertas ofrecen a las partes interesadas la oportunidad de expresar sus opiniones y puntos de vista con sus propias palabras. Esto puede ayudar a identificar problemas o preocupaciones que podrían no haber previsto. Por ejemplo, en una encuesta se puede pedir a los encuestados que describan con sus propias palabras su experiencia con un determinado producto o servicio. Los datos cualitativos resultantes pueden aportar información valiosa sobre las necesidades y preferencias de los clientes.

- Las encuestas son una forma relativamente rápida, eficaz y barata en comparación a otros métodos, de recabar información de un gran número de interesados. Los encuestados pueden completar las encuestas a su propio ritmo, lo que las hace más convenientes que otras técnicas de obtención de información, como las entrevistas o los grupos de discusión.

- Las encuestas pueden distribuirse por internet, lo que las convierte en una forma eficaz de obtener información de partes interesadas dispersas geográficamente.

- Las encuestas tienen el potencial de generar una gran cantidad de datos, lo que puede ser útil para identificar patrones y tendencias. Sin embargo, es importante tener en cuenta que analizar grandes cantidades de datos requiere tiempo y recursos.

- En comparación con otras técnicas de obtención de datos como las entrevistas, las encuestas suelen requerir menos tiempo y son menos costosas de administrar. Esto las convierte en una forma rentable de recabar información de un gran número de personas.

¿En qué condiciones puedo aplicar esta técnica?

La técnica de encuestas puede aplicarse en diversas condiciones. Estas condiciones incluyen:

- Cuando se necesita recopilar información de un gran número de interesados.

- Cuando necesite recopilar datos de forma anónima.

- Cuando necesite recopilar datos de forma cuantitativa y cualitativa.

- Cuando los encuestados están geográficamente dispersos.

- Cuando sea necesario recopilar datos sobre un tema o cuestión en concreto.

- Cuando se dispone de tiempo o recursos limitados.

- Cuando desee ahorrar tiempo y reducir los conflictos de agenda permitiendo a los interesados realizar la encuesta tengan disponibilidad.

Proceso de desarrollo

Para garantizar el éxito de la encuesta, es necesario prepararla minuciosamente para obtener la información necesaria y, al mismo tiempo, reducir al mínimo el tiempo que necesitan los encuestados para completarla. Deben tener en cuenta los siguientes pasos:

1. **Definir el objetivo de la encuesta y el grupo destinatario:** Identificar los objetivos y el probable grupo de usuarios a encuestar.

2. **Elegir el tipo de encuesta adecuado:**

    - Preguntas cerradas: En este tipo de preguntas, se pide al encuestado que seleccione entre las respuestas disponibles. Las preguntas cerradas son útiles cuando se conocen los temas, pero no el abanico de respuestas de los usuarios a los mismos. Las respuestas a preguntas cerradas son más fáciles de analizar que las obtenidas con preguntas abiertas.

    - Preguntas abiertas: En este tipo de preguntas, el encuestado es libre de responder a la pregunta como desee. Las preguntas abiertas pueden proporcionar respuestas cualitativas más detalladas y aportar ideas y opiniones que no se obtienen fácilmente con otras técnicas de obtención.

    - El tipo de encuestas también puede ser un conjunto de preguntas abiertas y preguntas cerradas, pudiendo así adecuar la encuesta hacia los objetivos de la misma.

3. **Seleccione el grupo de muestra:** Considere tanto el tipo de encuesta como el número de personas del grupo de usuarios identificado para determinar si se debe encuestar a todo el grupo. Los métodos de muestreo estadístico pueden ayudar a garantizar que los resultados de la encuesta no estén sesgados.

4. **Seleccionar los métodos de distribución y recogida:** Determine el modo de comunicación adecuado para cada grupo de muestreo.

5. **Proyectar el nivel de respuesta deseado:** Determine qué tasa de respuesta sería aceptable y considere la posibilidad de ofrecer incentivos para aumentar las tasas de respuesta.

6. **Determine si la encuesta debe apoyarse con entrevistas individuales:** Dado que una encuesta puede no proporcionar la profundidad de datos que pueden obtenerse de entrevistas individuales, considere la posibilidad de realizar entrevistas antes y después de la encuesta para obtener un mayor nivel de detalle. Para saber como hacer buenas entrevistas, considere ver la guía que hemos realizado.

7. **Redactar las preguntas de la encuesta:**

    - Comunicar la finalidad de la encuesta y explicar los objetivos a las partes interesadas.

    - Centrar todas las preguntas en los objetivos declarados y asegurarse de que cada objetivo esté respaldado por un conjunto completo de preguntas.

    - Que la encuesta sea breve, idealmente con menos de 10 preguntas, fácil y rápida de completar en no más de cinco o diez minutos.

    - Utilice una redacción clara y concisa, evite las preguntas dobles o negativas, y evite las estructuras ramificadas complejas o las preguntas incómodas.

8. **Prueba la encuesta:** Realizar una prueba de usabilidad y ajustar la encuesta en función de los resultados.

9. **Distribuye la encuesta:** Distribuye la encuesta según lo dispuesto en el punto 4.

10. **Documente los resultados de la encuesta:** Recopile y analice las respuestas.

Consideraciones para aplicar la técnica

- Defina claramente el objetivo y el alcance de la encuesta.

- Utilizar un lenguaje claro y conciso en las preguntas de la encuesta.

- Realice una prueba piloto de la encuesta con un pequeño grupo de interesados antes de distribuirla más ampliamente.

- Analice detenidamente los resultados, busque patrones y tendencias que puedan servir de base para las actividades de análisis.

- Hacer lo posible para evitar ambigüedad tanto en las preguntas como en las posibles respuestas, si corresponde.

- Evite el sesgo, usted puede estar inducido por su propio sesgo, intente librarse de este al momento de redactar las preguntas.

Herramientas Sugeridas

- **Google Forms**: Con esta herramienta puedes crear y analizar encuestas desde un dispositivo móvil o un navegador web sin tener que usar ningún software especial. Puedes ver los resultados al instante, en el momento en que se envían, y organizarlos en gráficos para consultarlos fácilmente. Del mismo modo, los encuestados pueden realizar la encuesta fácilmente.

Donde leer más

- IIBA, *A Guide to the Business Analysis Body of Knowledge (*Version 2.0*)*, Toronto, Ontario, Canada: International Institute of Business Analysis, 2008.
- A. Van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley & Sons, 2009.
- K. Wiegers and J. Beatty, *Software Requirements*, 3rd ed., Microsoft Press, 2013.

ANEXO 2 - Guía de entrevistas

Introducción

Las entrevistas son un tipo de técnica de elicitación en la que el entrevistador y el entrevistado discuten de forma abierta para tener mayor conocimiento de las necesidades de los usuarios a los que se quiere llegar. Existen distintos tipos de entrevistas, estructuradas y no estructuradas. Las entrevistas estructuradas son las que el entrevistador tiene un conjunto predefinido de preguntas y busca respuestas a las mismas, mientras que las entrevistas no estructuradas son en las que, sin preguntas predefinidas, el entrevistador y el entrevistado debaten de forma abierta.

El éxito de una entrevista depende de varios factores, como el nivel de conocimientos del entrevistador sobre el tema, la experiencia del entrevistador en la realización de entrevistas, la habilidad del entrevistador para documentar las conversaciones, la disposición del entrevistado a proporcionar información relevante, el grado de conocimiento del entrevistado sobre lo que se quiere y espera del sistema objetivo, y la relación entre el entrevistador y el entrevistado.

¿Por qué debería incluirla en mi emprendimiento de software?

Las entrevistas tienen varias ventajas como técnica de elicitación, entre ellas:

- Fomenta la participación y establece una buena relación con la parte interesada.
- Técnica sencilla y directa que puede utilizarse en diversas situaciones.
- Permite al entrevistador y al participante mantener conversaciones y explicaciones completas sobre las preguntas y respuestas.
- En algunas circunstancias permite observar el comportamiento no verbal.
- El entrevistador puede hacer preguntas de seguimiento y de sondeo para confirmar su propia comprensión.

¿En qué condiciones puedo aplicar esta técnica?

Las entrevistas pueden aplicarse como técnica para obtener información de las partes interesadas en diversos procesos de ingeniería de software. Las entrevistas son especialmente útiles cuando la información necesaria es subjetiva o difícil de cuantificar, o cuando es importante tener en cuenta el punto de vista de las partes interesadas. Las entrevistas también pueden servir para aclarar requisitos, identificar riesgos y validar suposiciones.

Sin embargo, es importante tener en cuenta que las entrevistas no siempre son la técnica más adecuada para cualquier situación. Factores como las limitaciones de tiempo, la disponibilidad de las partes interesadas y la complejidad de la información que se busca deben tenerse en cuenta a la hora de decidir si se recurre o no a las entrevistas como técnica.

Proceso de desarrollo

A continuación, se detallan los pasos para aplicar las entrevistas como técnica de elicitación:

1. Preparación de la entrevista:

- Determinar qué información se necesita y qué preguntas deben formularse para obtenerla.
- Identifique a las partes interesadas que tengan conocimientos o experiencia relevantes para los requisitos que se quieren obtener.
- Elabore una lista de preguntas abiertas que fomenten el debate y permitan obtener información detallada. Las preguntas cuyas respuestas son simplemente sí o no, no aportan información de calidad.
- Programe entrevistas con las partes interesadas y proporcionarles cualquier información o material de referencia necesario.

2. Realización de la entrevista:

- Cree un entorno cómodo para el debate y establezca una buena relación con las partes interesadas para fomentar una comunicación abierta.
- Utilice preguntas abiertas para fomentar el debate y obtener información detallada.
- Utilizar técnicas de escucha activa para garantizar una comprensión precisa de las respuestas de las partes interesadas. Algunas técnicas de escucha activa pueden ser el parafraseo, el resumen y realizar preguntas para clarificar alguno de los conceptos manejados. El parafraseo consiste en decir lo que ha dicho el interlocutor con tus propias palabras, sin cambiar el sentido del mensaje. El parafraseo puede ayudar a aclarar información, confirmar la comprensión y demostrar al interlocutor que estás escuchando activamente y participando en la conversación. En cuanto a resumir, este consiste en repetir los puntos clave o las ideas principales de lo que ha dicho el orador, de forma concisa y organizada.
- Haga preguntas de seguimiento según sea necesario para aclarar las respuestas u obtener información adicional.

3. Analizar los datos de las entrevistas:

- Revisar las notas o grabaciones de las entrevistas para identificar temas y patrones comunes.
- Organizar las conclusiones en categorías o grupos en función de similitudes o diferencias.
- Validar los hallazgos con las partes interesadas para garantizar su exactitud e integridad.

4. Documentar los requisitos:

- A partir de la información recopilada en las entrevistas, documentar los requisitos utilizando un formato adecuado para el proyecto (por ejemplo, historias de usuario, casos de uso, requisitos funcionales).

- Asegúrese de que los requisitos son claros, concisos, completos y comprobables.

Donde leer más

- IEEE Computer Society, "Guide to the Software Engineering Body of Knowledge (Version 3.0)," 2014.
- IIBA, *A Guide to the Business Analysis Body of Knowledge (*Version 2.0*)*, Toronto, Ontario, Canada: International Institute of Business Analysis, 2008.
- K. Wiegers and J. Beatty, *Software Requirements*, 3rd ed., Microsoft Press, 2013.

ANEXO 3 - Guía de estudio de casos

Introducción

El estudio de casos es una técnica que se centra en la comprensión profunda y detallada del caso estudiado. En lugar de buscar generalizaciones, el estudio de casos busca comprender cómo funciona un caso particular en su contexto específico. Por lo tanto, esta técnica es especialmente útil para explorar preguntas complejas y difíciles que no pueden ser abordadas fácilmente por otros métodos de investigación.

¿Por qué debería incluirla en mi emprendimiento de software?

- El Estudio de Casos puede ayudar a identificar y solucionar problemas de software antes de que se conviertan en problemas mayores. Esto puede mejorar la calidad del software y reducir la cantidad de errores que los usuarios experimentan.
- Ayuda a identificar y solucionar problemas de software en una etapa temprana, puede reducir costos asociados con el mantenimiento y la reparación de errores, ahorrando costos en este proceso.
- Ayuda a validar la calidad y eficacia de una solución propuesta antes de implementarla. Favorece evitar problemas y errores en la implementación del software.

¿En qué condiciones puedo aplicar esta técnica?

- Cuando un emprendimiento de software se enfrenta a un problema, puede ser de mucha utilidad conocer cómo otras empresas se enfrentan a una situación similar.
- Si se pretende validar una hipótesis en el mercado.
- Cuando se pretende analizar los riesgos de una acción que se quiere llevar a cabo.

Proceso de Desarrollo

1. **Diseñar el estudio de caso:**

    - Seleccione un caso o casos específicos que sean relevantes para su investigación.
    - Defina las preguntas de investigación y establezca criterios de selección.
    - Determine los límites del caso, incluyendo el contexto, los participantes y las actividades involucradas.

2. **Planificar la recopilación de datos:**

    - Seleccione fuentes de datos relevantes para su estudio, como entrevistas, observación directa, análisis documental y experimentación.
    - Planifique cómo recopilar los datos y cómo los validará.
    - Considere la validez interna y externa al seleccionar fuentes de datos.

      La validez interna se refiere a la capacidad del estudio de brindar las conclusiones buscadas, mientras que la validez externa se refiere a la generalización de las conclusiones a otros entornos o situaciones similares.

3. **Recopilar datos:**

    - Recopile los datos según lo planificado en el paso anterior.
    - Asegúrese de validar los datos recopilados para garantizar su precisión y confiabilidad.
    - Utilice múltiples fuentes de datos para aumentar la validez del estudio.

4. **Analizar los datos:**

    - Organice y analice los datos recopilados según las preguntas de investigación definidas en el paso 1.
    - Utilice técnicas apropiadas para analizar diferentes tipos de datos (por ejemplo, entrevistas).
    - Considere la validez interna y externa al analizar los datos.

5. **Interpretar los resultados:**

    - Interprete los resultados del análisis y extrapolar conclusiones relevantes para su investigación.
    - Compare sus resultados con los resultados de otros estudios de casos y teorías existentes.
    - Considere la validez interna y externa al interpretar los resultados.

6. **Informar sobre el estudio:**

    - Escriba un informe detallado sobre su estudio, incluyendo información sobre el diseño del estudio, la recopilación y análisis de datos, así como las conclusiones obtenidas.
    - Utilice un formato claro y conciso para presentar sus resultados.
    - Considere la validez interna y externa al informar sobre el estudio.

7. **Validar el informe:**

    - Valide su informe en la medida de lo posible con expertos en la materia para garantizar que sea preciso y relevante.
    - Considere la retroalimentación de los expertos y realice cambios en su informe según sea necesario.

- Asegúrese de que su informe cumpla con los estándares éticos y profesionales para la investigación en ingeniería de software.

Consideraciones para aplicar la técnica:

- Definir claramente el problema de investigación y los objetivos del estudio de casos.
- Utilizar la mayor cantidad de fuentes posibles, sin descuidar su origen y que tan validables son. Utilice fuentes validables.
- Presentar los resultados del estudio de manera clara y lo menos ambigua posible.
- Evaluar críticamente la calidad del estudio de casos, teniendo en cuenta su validez interna y externa.

Donde leer más

- R. K. Yin, *Case Study Research: Design and Methods*, 5th ed. Sage Publications, 2014.
- P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, 2012.
- A. Cockburn, *Applying Use Cases: A Practical Guide*, 2nd ed. Addison-Wesley Professional, 2000.

ANEXO 4 - Guía de integración y despliegue continuo

Introducción

CI/CD son un conjunto de prácticas de desarrollo de software donde el foco está en la automatización y la mejora continua del proceso de integración y entrega de software.

CI (*Continuous Integration*) se refiere a la práctica de integrar el código de diferentes desarrolladores en un repositorio compartido de manera frecuente. Esto ayuda a detectar errores y conflictos temprano en el proceso de desarrollo.

CD (*Continuous Delivery*) se refiere a la práctica de automatizar todo el proceso de entrega del software, desde la construcción hasta la implementación. Esto ayuda a acelerar la entrega del software y reducir los errores.

¿Por qué debería incluirla en mi emprendimiento de software?

- Al automatizar las pruebas y la entrega del software, se pueden detectar errores y problemas más rápidamente.
- Si se logra automatizar todo el proceso de entrega del software, desde la construcción hasta la implementación, se puede acelerar significativamente el tiempo necesario para entregar nuevas características y correcciones de errores.
- La entrega de nuevas características y correcciones de errores más rápidamente y con mayor calidad, se puede mejorar la satisfacción del cliente.
- Al utilizar herramientas para integrar continuamente el código de diferentes desarrolladores en un repositorio compartido, se puede fomentar una mayor colaboración entre equipos.
- Disminuye la carga sobre el equipo de trabajo, ya que gran parte de las tareas se automatizan mejorando así la eficiencia.

¿En qué condiciones puedo aplicar esta técnica?

- Todos los equipos involucrados deben trabajar en estrecha colaboración.
- El proceso debe estar claramente documentado y el equipo debe estar dispuesto a seguir estos lineamientos.
- Disponer de los procesos de automatización necesarios operativos y funcionales, además de la infraestructura necesaria.

Proceso de Desarrollo

1. **Automatizar la construcción del software**: En este punto se deberá establecer un repositorio de código y automatizar la interacción con él.
2. **Configurar un servidor de integración continua**: Es necesario para que aquí se lleve a cabo la integración de código automatizada, se realizará la compilación, ejecutarán las pruebas automatizadas y se notificará a los interesados como finalizó este proceso.
3. **Configurar pruebas automatizadas**: Escriba pruebas automatizadas, esto permite encontrar errores y poder iterar rápidamente para solucionar los inconvenientes.
4. **Configurar un flujo de entrega continua**: Mediante esta acción se entregará código de manera continua, esto potencia en gran medida la posibilidad de entregar nuevas funcionalidades de forma fluida y automatizada.

   En este punto también debe tener configurada su infraestructura de despliegue, es el medio por el cual lograra entregar código a los usuarios.
5. **Mejorar continuamente**: Realice mejoras continuas en su proceso de CI/CD utilizando métricas, retroalimentación del usuario y miembros del equipo.

Consideraciones para aplicar la técnica:

- En lugar de intentar implementar todo el proceso de CI/CD de una sola vez, es mejor comenzar con pequeños cambios y agregar más funcionalidades a medida que el equipo se sienta cómodo.
- La automatización es clave para el éxito de CI/CD. Automatizar las pruebas, la compilación y la implementación puede ahorrar tiempo y reducir errores.
- Es importante proporcionar retroalimentación continua sobre el proceso de CI/CD para identificar áreas problemáticas y mejorar continuamente el proceso.
- La implementación exitosa de CI/CD requiere una colaboración efectiva entre los equipos de desarrollo, operaciones y pruebas.
- Analizar las opciones de herramientas que existen en el mercado, hay muchas herramientas disponibles para ayudar en la implementación de CI/CD. Es importante elegir las herramientas adecuadas que se adapten a las necesidades específicas del equipo.

Donde leer más

- J. Humble, D. Farley, S. Matyas, A. Glover, and E. Tavela, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 2011.
- G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2016.
- J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 2010.

ANEXO 5 - Guía de manejo de deuda técnica

Introducción

El manejo de la deuda técnica es un proceso continuo que implica la identificación, priorización y reducción de la deuda técnica en un sistema de software. La deuda técnica se refiere a los compromisos técnicos que se han tomado en el desarrollo del software y que pueden tener un impacto negativo en su calidad o capacidad para evolucionar en el futuro.

El manejo efectivo de la deuda técnica requiere una comprensión profunda de las prácticas y herramientas de ingeniería de software, así como una cultura organizacional que valore la excelencia técnica. Se debe trabajar para identificar y priorizar los elementos críticos de la deuda técnica, establecer planes para abordarlos y monitorear continuamente el progreso.

El manejo adecuado de la deuda técnica puede ayudar a mejorar la calidad del software, reducir los costos a largo plazo y aumentar la satisfacción del usuario. Sin embargo, si no se gestiona adecuadamente, puede tener un impacto negativo en el rendimiento del sistema, logrando el efecto contrario, e inclusive quitarles la motivación a los propios desarrolladores.

¿Por qué debería incluirla en mi emprendimiento de software?

La aplicación de un proceso para gestionar la deuda técnica puede aportar varias ventajas, entre ellas:

- Al abordar la deuda técnica, se pueden corregir errores y mejorar la calidad del software, lo que puede aumentar la satisfacción del usuario.
- La gestión adecuada de la deuda técnica puede ayudar a reducir los costos a largo plazo asociados con el mantenimiento y evolución del sistema.

- Al reducir la deuda técnica, se puede aumentar la capacidad del sistema para evolucionar y adaptarse a las necesidades cambiantes.
- La gestión adecuada de la deuda técnica puede ayudar a mejorar el rendimiento del equipo al reducir el tiempo dedicado al mantenimiento y aumentando el tiempo dedicado a nuevas funcionalidades.
- Permite justificar deliberadamente las decisiones cuando se incurre en deuda técnica, de forma tal de estar totalmente seguros de los pasos a realizar.
- Al gestionar adecuadamente la deuda técnica, potencialmente permitirá que la deuda se pague en el futuro.
- Mejora los atributos de calidad del sistema como el rendimiento y la modularidad.

¿En qué condiciones puedo aplicar esta técnica?

Las organizaciones deben gestionar la deuda técnica en las siguientes condiciones.

- Si la deuda técnica está causando problemas significativos, como la reducción de la productividad, el aumento de los costos de mantenimiento o la disminución de la calidad, puede ser necesario gestionarla.
- Si la deuda técnica está obstaculizando la capacidad de una organización para innovar y ofrecer nuevas características o funcionalidades, puede ser necesario gestionarla.
- Si la deuda técnica está afectando a la satisfacción del cliente con un producto o servicio de software, puede ser necesario gestionarla.
- Si abordar la deuda técnica proporciona beneficios significativos, como la mejora de la calidad, el aumento de la productividad o la reducción de los costos de mantenimiento, por lo que puede ser necesario gestionarla.

Proceso de desarrollo

1. Identificar los elementos de la deuda técnica: El primer paso en la gestión de la deuda técnica es identificar los elementos que deben abordarse. Esto puede incluir el código que es difícil de entender o modificar, el código que es propenso a defectos, o código que no está funcionando bien.

2. Priorizar los elementos de la deuda técnica: Una vez que haya identificado los elementos de deuda técnica, debe priorizarlos en función de factores como la gravedad, la complejidad, el riesgo, su impacto en el sistema y el costo de abordarlos. Esto le ayudará a asegurarse de que está centrando sus esfuerzos en las áreas más importantes.

3. Documente la deuda técnica: Es importante documentar cada elemento de deuda técnica en un registro u otro sistema de seguimiento. Esta documentación debe incluir información como la naturaleza de la deuda, su gravedad e impacto en el sistema, y cualquier solución propuesta.

4. Evaluar el estado de desarrollo del software: Para abordar la deuda técnica con eficacia, es importante evaluar el estado actual del desarrollo de software en su organización. Para evaluar el estado actual del desarrollo de software en su organización, puede realizar revisiones o auditorías de código, analizar métricas de calidad del código como la cobertura de código o la complejidad ciclomática, o revisar los estándares de codificación y las mejores prácticas.

5. Elabore un plan para abordar la deuda técnica: Una vez que haya identificado y priorizado los elementos de la deuda técnica y evaluado el estado de desarrollo del software, debe construir un plan para abordar cada elemento de su lista. Este plan debe incluir metas y objetivos específicos para cada elemento, así como plazos y recursos necesarios.

6. Abordar la deuda técnica mediante la refactorización: La refactorización es un enfoque común para abordar la deuda técnica, y consiste en realizar cambios en el código con el fin de mejorar su calidad y mantenimiento. Al refactorizar el código, se sugiere utilizar

herramientas de automatización de pruebas siempre que sea posible y realizarlas de manera exhaustiva después de cada cambio.

7. Abordar la deuda técnica intencionada: La deuda técnica intencionada se contrae intencionadamente para cumplir objetivos a corto plazo. Es importante identificar los elementos de deuda técnica intencional para poder gestionarlos adecuadamente junto con los no intencionales.

8. Supervise el progreso: Cuando empiece a abordar los elementos de deuda técnica, es importante supervisar el progreso y realizar un seguimiento de los cambios a lo largo del tiempo. Esto se puede hacer mediante la revisión periódica de métricas tales como las tasas de defectos, así como el seguimiento de los cambios en la calidad del código y el rendimiento del sistema.

9. Mejorar continuamente: Por último, es importante reconocer que la gestión de la deuda técnica es un proceso continuo que requiere una mejora continua en el tiempo. A medida que aborda los elementos de deuda técnica existentes y evita que se acumulen otros nuevos, debe seguir perfeccionando sus procesos y estrategias de gestión de la deuda técnica para garantizar la salud y sostenibilidad a largo plazo del sistema.

Herramientas Sugeridas

SonarQube: SonarQube es una plataforma de análisis de código y gestión de calidad que ayuda a los desarrolladores a identificar la deuda técnica, *code smells* y las vulnerabilidades de seguridad. Ofrece una serie de herramientas, como el análisis, la cobertura y el análisis estático de código.

Jira: Jira es una herramienta de gestión de proyectos que puede utilizarse para realizar un seguimiento de la deuda técnica. Permite a los equipos crear tickets para la deuda técnica, asignarlos a los desarrolladores y realizar un seguimiento del progreso.

Donde leer más

- P. Kruchten, R. Nord, and I. Ozkaya, *Managing Technical Debt: Reducing Friction in Software Development.* Addison-Wesley Professional, *2019*
- M. Fowler, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 2018.
- N. Ernst, R. Kazman, and J. Delange, *Technical Debt in Practice: Practical Methods and Metrics for Software Teams*. The MIT Press, 2019.

ANEXO 6 - Guía de mockups

Introducción

El diseño de interfaces de usuario es un proceso crítico en el desarrollo de productos digitales. Una interfaz bien diseñada puede mejorar significativamente la experiencia del usuario y aumentar la adopción y retención del producto. Sin embargo, diseñar una interfaz efectiva puede ser un desafío, especialmente cuando se trata de equilibrar las necesidades del usuario con los requisitos técnicos y comerciales.

Una forma efectiva de diseñar y refinar interfaces de usuario es a través del uso de mockups. Los mockups son representaciones visuales de la interfaz que permiten a los diseñadores y desarrolladores probar diferentes diseños y funcionalidades antes de comenzar la implementación. Los mockups pueden ser estáticos o interactivos, y pueden variar en complejidad desde bocetos simples hasta prototipos completamente funcionales.

Una técnica popular para crear mockups es el *paper prototyping*, que utiliza una versión en papel de la interfaz para simular interacciones con los usuarios. El paper prototyping es una forma rápida y fácil de diseñar y refinar interfaces de usuario antes de comenzar la implementación. Esta técnica permite a los diseñadores probar diferentes diseños, iterar rápidamente sobre las ideas y obtener comentarios valiosos antes de invertir tiempo y recursos en el desarrollo.

¿Por qué debería incluirla en mi emprendimiento de software?

- Los mockups son una herramienta visual que permite a los diseñadores y equipos de desarrollo probar la apariencia, estructura y funcionalidad de un sitio web o aplicación antes de su lanzamiento.
- Son útiles para explorar decisiones de diseño visual antes de tener que experimentar las consecuencias de un mal código.

- Los mockups son una fase transicional entre los Wireframes y los prototipos, lo que les permite ser un espacio para experimentar visualmente. (Ver guías de wireframes y prototipos).
- Si se pasa por alto la fase de mockup, puede llevar a resultados visuales deficientes en el proceso de diseño y desarrollo.
- Utilizar mockups puede ayudar a identificar problemas y realizar ajustes antes del lanzamiento, lo que puede ahorrar tiempo y recursos en el futuro.
- Los mockups pueden ayudar a comunicar claramente su mensaje y mejorar la experiencia del usuario final.

¿En qué condiciones puedo aplicar esta técnica?

Los mockups resultan muy útiles cuando se necesita validar una idea de un producto o servicio enfocándose en la apariencia visual, permitiendo definir cómo se verá, sin llegar a probar una funcionalidad cómo se logra al aplicar un prototipo.

Otro caso de uso es cuando se necesita mostrar a inversores como se verá el sistema antes de desarrollarlo, así como permitir ver la aceptación de los usuarios a las interfaces.

Todas esas situaciones tienen la ventaja de que se desarrollan con un costo bajo y en un tiempo relativamente corto.

Proceso de Desarrollo

1. **Identificar los objetivos del proyecto y las necesidades del usuario:** Antes de comenzar a crear mockups, es importante tener una comprensión clara de los objetivos del proyecto y las necesidades del usuario. Esto puede incluir la identificación de los problemas que el sitio o aplicación debe resolver, así como la comprensión de las expectativas y preferencias del usuario.

2. **Establecer la estructura y el flujo del sitio o aplicación**: Una vez que se han identificado los objetivos y necesidades, es útil crear bocetos y Wireframes para establecer la estructura básica del sitio o aplicación. Los bocetos pueden ser dibujados a mano o creados digitalmente, mientras que los Wireframes son representaciones más detalladas de la estructura del sitio o aplicación. (Ver guía sobre wireframes)

3. **Crear mockups para explorar decisiones de diseño visual y realizar pruebas con usuarios**: Después de establecer la estructura básica, es hora de crear mockups para explorar decisiones de diseño visual. Los mockups pueden ser creados utilizando herramientas dedicadas o herramientas web, y deben incluir elementos visuales como colores, tipografía e imágenes. También es importante realizar pruebas con usuarios en esta etapa para asegurarse de que el diseño sea intuitivo y fácil de usar.

4. **Utilizar los resultados de las pruebas para iterar y mejorar el diseño antes del lanzamiento:** Finalmente, después de realizar pruebas con usuarios, es importante utilizar los resultados para iterar y mejorar el diseño antes del lanzamiento. Esto puede incluir ajustes en la paleta de colores, cambios en la disposición de los elementos visuales o mejoras en la navegación del sitio o aplicación.

Consideraciones para aplicar la técnica:

- Mantén los mockups simples y enfocados en la funcionalidad.
- Utilizar una paleta de colores limitada para evitar distracciones.
- Asegúrate de que los elementos visuales sean coherentes en todo el diseño.
- Utiliza texto realista para simular cómo se verá el contenido final.
- Asegúrate de que los botones y otros elementos interactivos sean fácilmente identificables y accesibles.

Donde leer más

- C. Snyder, *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. Morgan Kaufmann Publishers, 2003.
- J. Tidwell, *Designing Interfaces: Patterns for Effective Interaction Design*, 2nd ed. O'Reilly Media, 2010.
- E. Bjarnason, "Prototyping practices in software startups: Initial case study results," in *Proceedings of the 18th International Conference on Agile Software Development*, 2016.

ANEXO 7 - Guía de observación

Introducción

La observación se basa en el estudio de las personas en el desempeño de su trabajo (en nuestro caso, utilizando una solución de software), y en ocasiones se denomina *job shadowing*. Por ejemplo, algunas personas tienen su rutina de trabajo tan arraigada que les cuesta explicar lo que hacen o por qué lo hacen. El observador puede tener que verlos realizar su trabajo para entender el flujo de trabajo. En determinados proyectos, es importante conocer los procesos actuales para evaluar mejor las modificaciones que pueden ser necesarias.

Existen dos enfoques básicos para la técnica de observación:

1. **Pasivo**: En este enfoque, el observador observa al usuario trabajando, pero no hace preguntas. El observador escribe notas sobre lo que ve, pero por lo demás se mantiene al margen, como si fuera invisible. El observador espera hasta que se haya completado todo el proceso antes de hacer preguntas.

2. **Activo**: En este enfoque, el observador interactúa con el usuario y le hace preguntas mientras realiza sus tareas. El observador puede hacer preguntas aclaratorias o buscar información adicional para comprender mejor cómo se hacen las cosas. Este enfoque es más interactivo que la observación pasiva y puede aportar información más detallada sobre el funcionamiento de los procesos.

¿Por qué debería incluirla en mi emprendimiento de software?

La observación como técnica de obtención de requerimientos tiene varias ventajas, entre ellas:

- La observación permite ver cómo se realiza realmente el trabajo en el entorno laboral, lo que puede ayudar a identificar problemas y oportunidades de mejora que pueden no ser evidentes mediante otras técnicas de elicitación. Esta visión de primera mano

también puede ayudar a comprender el contexto en el que se realiza el trabajo, incluidas las limitaciones o desafíos que pueden afectar a la forma en que se realiza el mismo.

- La observación es especialmente útil a la hora de documentar detalles sobre los procesos actuales o si el proyecto pretende mejorar o cambiar un proceso actual. Al observar a los usuarios mientras realizan sus tareas, se puede comprender mejor cómo se hacen las cosas e identificar áreas de mejora.

- Al observar a las personas mientras realizan sus tareas, se pueden identificar áreas de mejora que pueden no ser evidentes a través de otras técnicas de elicitación, así como también reducir el sesgo. Por ejemplo, pueden detectar ineficiencias en la forma de realizar el trabajo u oportunidades para racionalizar los procesos.

- La observación activa puede proporcionar información más detallada sobre el funcionamiento de los procesos que la observación pasiva. La observación activa consiste en hacer preguntas e interactuar con los expertos en la materia para comprender mejor su trabajo, mientras que la observación pasiva consiste simplemente en observar lo que hacen las personas sin interactuar con ellas.

¿En qué condiciones puedo aplicar esta técnica?

La observación es una técnica adecuada cuando se documentan detalles sobre los procesos actuales o si el proyecto pretende mejorar o cambiar un proceso actual. Puede utilizarse en las siguientes condiciones:

- Cuando se necesita comprender cómo se utiliza realmente el sistema.

- Cuando se necesita identificar problemas y oportunidades de mejora que pueden no ser evidentes a través de otras técnicas de elicitación.

- Cuando necesite comprender mejor cómo se hacen las cosas e identificar áreas de mejora.

Proceso de desarrollo

Los pasos para llevar a cabo buenas observaciones pueden dividirse en tres categorías: Preparación, Observación y por último Documentación y confirmación. A continuación, se detalla cada categoría:

1. Preparación:

- Identificar al usuario o los usuarios que se van a observar.
- Definir los objetivos de la observación.
- Determinar el alcance de la observación.
- Obtener el permiso de los usuarios a observar.
- Fijar una hora para la observación que sea conveniente para ambas partes.
- Preparar cualquier material o equipo necesario para la observación.

2. Observación:

- El observador se presenta a la persona observada y le explica que no se cuestiona su trabajo, sino que está allí para observar y documentar procesos como forma de aportar al análisis de requerimientos.
- Observar y documentar cómo se trabaja realmente en el entorno laboral, incluida cualquier comunicación informal o forma en que las personas se desenvuelven realmente en el sistema que pueda no estar documentada en ninguna parte.
- Tomar notas detalladas durante la observación, incluyendo cualquier pregunta o preocupación que surja durante este proceso.

3. Documentación y confirmación:

- Revise las notas tomadas durante la observación y aclare cualquier punto poco claro con quien corresponda.

- Documente todas las conclusiones de la observación de forma clara y concisa, incluidas las hipótesis formuladas durante el proceso.

- Confirmar cualquier suposición realizada durante este proceso con quien corresponda.

Donde leer más

- IIBA, *A Guide to the Business Analysis Body of Knowledge (*Version 2.0*)*, Toronto, Ontario, Canada: International Institute of Business Analysis, 2008.
- K. Wiegers and J. Beatty, *Software Requirements*, 3rd ed., Microsoft Press, 2013
- A. Van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley & Sons, 2009.

ANEXO 8 - Guía de pruebas A/B

Introducción

Las pruebas A/B son una técnica de experimentación que se utiliza generalmente en sitios web o aplicaciones, donde se muestran diferentes variaciones a diferentes personas y se mide qué variación es la más efectiva. Estas pruebas están compuestas por dos elementos; una variante a probar, también llamada tratamiento, y el control, que es una variante especial que corresponde a la versión existente en la que ejecutar la comparación.

Por ejemplo, si a cada visitante del sitio web se le muestra aleatoriamente una de estas variaciones y lo hace durante el mismo período de tiempo, entonces ha creado un experimento controlado del cual obtener resultados y tomar decisiones de acuerdo con las métricas definidas.

¿Por qué debería incluirla en mi emprendimiento de software?

- Utilizando las pruebas A/B se pueden ver distintas clases de problemas, como de usabilidad y diseño de un producto, permitiendo así solucionarlos, mejorarlos, y a fin de cuentas, que el usuario tenga una mejor experiencia.
- Al utilizar pruebas A/B se pueden realizar cambios que pueden ser significativos sin tener que rediseñar completamente, además de permitir sacar conclusiones de forma temprana sobre si un cambio es efectivo o no, permitiendo decidir si se desea realizar su implementación.
- En lugar de suposiciones o intuiciones sin sustento, utilizando pruebas A/B se podrá tomar decisiones basadas en datos. Con dichos datos, precisos y objetivos, las decisiones serán más efectivas y generarán mejoras en el producto.
- Estas pruebas permiten conocer las preferencias de los usuarios, por lo que nos dan la posibilidad de brindar un producto al público objetivo intentando cubrir de mejor manera sus necesidades y expectativas.

- Realizar pruebas A/B de manera continua y sostenida en el sistema, permitirá poder innovar en un área de constante evolución y cambios, de esta manera se puede estar a la vanguardia intentando cubrir las expectativas de los clientes.

¿En qué condiciones puedo aplicar esta técnica?

- Para llevar a cabo una prueba A/B efectiva, es importante tener suficiente tráfico en el sitio web o aplicación para poder dividirlo entre las diferentes variantes y que el volumen de interacción sea significativo y así obtener conclusiones confiables.

- Es importante tener un objetivo claro para la prueba y saber exactamente lo que se espera lograr.

- Antes de comenzar la prueba, es importante tener una hipótesis clara sobre qué variante del sitio web o aplicación funcionará mejor para lograr el objetivo deseado.

- Es importante ser capaz de medir los resultados de la prueba y determinar qué variante funcionó mejor para lograr el objetivo deseado.

Proceso de Desarrollo

1. **Definir el éxito y objetivo de la prueba:** Antes de comenzar una prueba A/B, es importante identificar claramente el objetivo de la prueba y lo que se espera lograr. Además de esto es muy importante que defina qué métrica o métricas va a tomar en cuenta para poder sacar conclusiones.

2. **Identificar cuellos de botella:** Una vez que haya determinado cuáles son las métricas de éxito cuantificables de su sitio, deberá centrar su atención en el análisis del mismo y descubrir dónde están los mayores cuellos de botella, los cuales suelen ser los lugares donde los usuarios están abandonando.

3. **Crear hipótesis:** Una vez que se ha identificado el objetivo, es importante crear una hipótesis sobre qué variante del sitio web o aplicación funcionará mejor para lograr ese objetivo.

4. **Crear variantes:** Después de crear una hipótesis, es hora de crear una variante del sitio web o aplicación para probar.

5. **Dividir el tráfico:** Para llevar a cabo una prueba A/B, es necesario dividir el tráfico entre las diferentes variantes del sitio web o aplicación. La división deberá ser equitativa en cantidad y totalmente al azar.

6. **Ejecutar la prueba:** Una vez que se ha dividido el tráfico, es hora de ejecutar la prueba y recopilar datos sobre cómo funcionan las diferentes variantes.

7. **Analizar los resultados:** Después de ejecutar la prueba, es importante analizar los resultados y determinar qué variante funcionó mejor en base a los objetivos previamente definidos.

8. **Implementar cambios:** Finalmente, después de analizar los resultados, se pueden implementar cambios en el sitio web o aplicación basados en lo que se aprendió durante la prueba A/B, en caso de que los resultados arrojen la necesidad de cambio.

Consideraciones para aplicar la técnica:

- No puede elegir un ganador hasta que decida cómo lleva el puntaje. Las pruebas A/B comienzan con la determinación de métricas de éxito cuantificables.

- Hay varias métricas posibles a utilizar, por ejemplo: tiempo en el sitio, páginas vistas, valor promedio de un pedido o compra, ingresos por visitante, etc. Tómese el tiempo para elegir el que sea adecuado para usted.

- El análisis del sitio junto con sus propios instintos le sugerirá cuellos de botella en los que puede centrar su atención.

- Comprender la intención del visitante con la ayuda de entrevistas y pruebas de usabilidad sugerirá hipótesis sobre qué cambiar y cómo.

- Priorice sus experimentos en función de su impacto.
- Comience a probar y continúe hasta que considere que el volumen de datos sea suficiente para sacar conclusiones.

Donde leer más

- R. Kohavi, D. Tang, and Y. Xu, *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge University Press, 2020.
- D. Siroker and P. Koomen, *A/B Testing: The Most Powerful Way to Turn Clicks into Customers*, USA, 2013.
- R. Koning, S. Hasan, and A. Chatterji, *Experimentation and startup performance: Evidence from A/B testing*. 2021.

ANEXO 9 - Guía de prototipos

Introducción

Las nuevas empresas de software exploran nuevas oportunidades comerciales y desarrollan productos innovadores en condiciones inciertas y de restricción de recursos. Si bien la disponibilidad de software de código abierto y los servicios de pago por uso brindan oportunidades comerciales basadas en software, las nuevas empresas luchan por encontrar el producto adecuado para el mercado y corren el riesgo de perder tiempo y recursos en el desarrollo de funciones que no tienen éxito. Un factor de éxito importante es probar la idea de negocio desde el principio para validar su viabilidad en el mercado. La creación de prototipos proporciona una forma rápida y relativamente económica de explorar y evaluar nuevas ideas y tecnologías mediante la producción de una muestra, un modelo o una versión temprana, que simula uno o varios aspectos del producto final y que se utiliza para explorar, comunicar y evaluar posibles soluciones.

Por lo tanto, un prototipo puede permitir pruebas rentables con usuarios reales y puede abarcar desde un simple boceto hasta una versión (incompleta) del software de producción, también conocido como producto mínimo viable.

¿Por qué debería incluirla en mi emprendimiento de software?

La creación de prototipos puede usarse con fines de validación, por ejemplo, demostrando maquetas a los clientes, como una forma rentable de obtener comentarios del mercado. De esta forma mejorando la comunicación entre el equipo de trabajo y los interesados.

Por otro lado, la aplicación de esta práctica ayuda al equipo de trabajo a identificar problemas de forma temprana, como pueden ser problemas de diseño o de usabilidad, entre otros. Lograr identificar estos problemas tempranamente tendrá como consecuencia un ahorro sustancial de tiempo y recursos, ya que tener que resolverlos en etapas más avanzadas del proyecto será una tarea que requerirá mayor esfuerzo.

¿En qué condiciones puedo aplicar esta técnica?

Si los requisitos de un proyecto no están del todo claros, no están bien definidos o suelen tener cambios habitualmente, generar un prototipo o varios y obtener información de cómo las partes interesadas interactúan con ellos, puede ayudar a definir de manera más acertada dichos requerimientos.

Otra situación en la que sería apropiado utilizarlos es cuando necesito probar algo, sea una funcionalidad, una interfaz de usuario, algún componente del sistema o parte de él, y se necesita obtener retroalimentación rápida, ya que de esta información dependen una serie de acciones posteriores.

Proceso de Desarrollo

Los pasos bases para llevar a cabo este proceso son:

1. **Definir el objetivo:** En esta etapa se debe definir cuál es el propósito del prototipo, que es lo que quiero probar o mostrar. ¿Validar una idea?, ¿Probar una funcionalidad nueva, o parte de ella?, ¿Atraer inversiones?

   Si logra definir con claridad cuál es el objetivo le será más fácil el diseño del mismo.

2. **Diseño:** En esta etapa, se crea un diseño inicial del sistema y se identifican las áreas que requieren más atención. También se pueden crear diagramas y modelos para ayudar a visualizar el sistema.

3. **Creación:** En este punto se debe crear el prototipo siguiendo el esquema previamente diseñado en el paso 2. Realiza pruebas de funcionalidad y usabilidad a medida que se construye el prototipo y posteriormente realiza los ajustes necesarios en el prototipo a medida que se vayan descubriendo posibles mejoras.

4. **Probar el prototipo:**

   En esta etapa, se evalúa el prototipo con los usuarios finales para obtener comentarios y sugerencias sobre cómo mejorar el sistema. Los cambios necesarios se realizan en el prototipo.

   Es importante registrar cómo se da la interacción y obtener toda la retroalimentación posible de la experiencia.

5. **Evaluar los resultados:**

   En esta instancia se analizará la información recolectada de la prueba del prototipo, con ella se podrá decidir si se vuelve a repetir el ciclo completo desde el paso 1, o con esta prueba ya se lograron sacar conclusiones definitivas y se avanza con el proyecto.

Consideraciones para aplicar la técnica:

- Es importante comprender las necesidades y expectativas de los usuarios finales antes de comenzar a desarrollar el prototipo.
- Es importante establecer objetivos claros para el prototipo y asegurarse de que sean alcanzables.
- Es importante utilizar herramientas adecuadas para el desarrollo del prototipo, como lenguajes de programación o materiales de construcción, teniendo en cuenta que es lo que se desea probar y qué conclusiones se espera recolectar.
- El diseño del prototipo debe ser lo más simple posible para facilitar su implementación y evaluación.
- Es importante obtener comentarios tempranos sobre el prototipo para poder realizar cambios en el diseño o la implementación según sea necesario.
- Se deben realizar pruebas exhaustivas en el prototipo para identificar errores o problemas antes de su implementación final. Sería aconsejable que se provea una guía de uso en caso de que la intención sea probar alguna sección en particular. Por el

contrario, si lo que desea es ver cómo interactúa un usuario nuevo sin conocer el sistema, déjelo que descubra por sí mismo.

- El proceso de refinamiento y mejora debe ser continuo durante todo el proceso de desarrollo del prototipo.

Donde leer más

- B. Clarysee and S. Kiefer, *The Smart Entrepreneur. How to Build for a Successful Business*. Elliott & Thompson, 2011.
- S. L. Pfleeger and J. M. Atlee, *Software Engineering. Theory and Practice*. Pearson, 2010.
- E. Bjarnason, "Prototyping Practices in Software Startups: Initial Case Study Results," in Proc. IEEE 29th Int. Requirements Eng. Conf. Workshops (REW), pp. 1-9, 2021.
- A. Nguyen-Duc, X. Wang, and P. Abrahamsson, "What Influences the Speed of Prototyping? An Empirical Investigation of Twenty Software Startups," in H. Baumeister et al. (Eds.): XP 2017, LNBIP 283, pp. 20–36, 2017.
- S. McConnell, *Rapid Development: Taming Wild Software Schedules*. Microsoft Press, 1996.

ANEXO 10 - Guía de pruebas de usabilidad

Introducción

Antes de adentrarnos en lo que respecta a esta guía, es importante saber que es específicamente la usabilidad. Podría decirse que algo es usable si no existe frustración a la hora de usarlo, y esto se da cuando el usuario puede hacer lo que él quiere de la forma que espera poder hacerlo, sin ningún tipo de trabas, dudas o preguntas.
Un producto puede considerarse usable si es útil, eficiente, eficaz, satisfactorio, fácil de aprender y accesible. Útil significa que el producto está diseñado para resolver un problema específico o satisfacer una necesidad concreta. Eficiente es cuando el producto permite a los usuarios realizar tareas rápida y fácilmente. Por eficaz entendemos que el producto ayuda a los usuarios a alcanzar sus objetivos. Satisfactorio indica que los usuarios disfrutan utilizando el producto y les resulta agradable. Fácil de aprender implica que los usuarios pueden aprender a utilizar el producto rápida y fácilmente. Accesible significa que el producto puede ser utilizado por personas con discapacidad o necesidades especiales.

Teniendo ahora esta breve definición de usabilidad, podemos adentrarnos en lo que refiere al contenido de esta guía, que es precisamente sobre las pruebas de usabilidad, que trata de un proceso que ayudará a conseguir un producto que cumpla con las seis propiedades anteriores, logrando que los usuarios tengan la mejor experiencia posible.

¿Por qué debería incluirla en mi emprendimiento de software?

- Las pruebas de usabilidad proporcionan información sobre cómo mejorar la experiencia del usuario. Esto conduce a un mayor compromiso, retención y satisfacción del usuario.

- Los emprendimientos de software tienen recursos limitados y necesitan optimizarlos. Este tipo de pruebas ayudan a detectar problemas de usabilidad en las primeras fases del proceso de desarrollo, lo que ahorra tiempo y dinero a largo plazo.

- Estos emprendimientos suelen tener suposiciones sobre su público objetivo y sus necesidades. Las pruebas de usabilidad ayudan a validar estas suposiciones y a garantizar que el producto está diseñado para satisfacer las necesidades del usuario.
- Un producto con una mejor experiencia de usuario puede dar a un emprendimiento de software una ventaja competitiva en el mercado. Las pruebas de usabilidad ayudan a identificar problemas de usabilidad que los competidores pueden estar pasando por alto, proporcionando una oportunidad de diferenciación.
- Una experiencia de usuario positiva puede aumentar la retención de usuarios, lo cual es importante para el crecimiento y el éxito de un emprendimiento de este tipo.
- Las pruebas de usabilidad pueden ayudar a aumentar las ventas al identificar y abordar los problemas que pueden estar impidiendo a los usuarios completar tareas o realizar compras en un sitio web o producto.

¿En qué condiciones puedo aplicar esta técnica?

Las pruebas de usabilidad son una herramienta que se puede aplicar en diversas situaciones, algunas de las cuales son:

- Se pueden aplicar durante el proceso de diseño de nuevos productos o servicios para garantizar que se creen soluciones fáciles de usar e intuitivas para los usuarios.
- También se pueden utilizar para mejorar productos y servicios existentes, lo que puede ayudar a identificar problemas en la usabilidad y realizar mejoras.
- Son especialmente útiles en el diseño de sitios web y aplicaciones móviles para garantizar que sean fáciles de usar y navegar.
- También se pueden utilizar para evaluar la accesibilidad de los productos y servicios para personas con discapacidades.

- Pueden ser utilizadas en la investigación de mercado para comprender mejor las necesidades y preferencias de los usuarios y diseñar soluciones más efectivas.

Proceso de desarrollo

Aquí se describen los pasos necesarios para implementar pruebas de usabilidad efectivas:

1. **Identificar los objetivos de la prueba:** Antes de realizar las pruebas de usabilidad, es importante tener objetivos claros. Los objetivos deben ser específicos y medibles para que los resultados de la prueba puedan ser evaluados con precisión.

2. **Seleccionar los usuarios**: Los usuarios deben ser seleccionados para representar el mercado objetivo del software. Los usuarios pueden ser reclutados a través de redes sociales, correos electrónicos, sitios web (ver sección de herramientas) o mejor aún entrevistas presenciales.

3. **Preparar el escenario de la prueba:** Se debe preparar un escenario que sea realista para el usuario y que incluya tareas comunes y críticas. Los evaluadores también deben preparar preguntas y notas para tomar durante la prueba.

4. **Realizar la prueba:** Los usuarios deben realizar las tareas mientras los evaluadores toman notas y hacen preguntas. La prueba debe ser grabada en video para que pueda ser revisada más tarde.

5. **Evaluar los resultados:** Después de la prueba, los evaluadores deben revisar las notas y los videos para identificar problemas de usabilidad. Los resultados deben ser documentados para que puedan ser evaluados más tarde.

6. **Realizar cambios:** Después de evaluar los resultados de la prueba, los cambios deben ser implementados en el software para mejorar la usabilidad.

Consideraciones para aplicar la técnica

- Los escenarios de prueba deben ser realistas y representar tareas que los usuarios podrían realizar normalmente en el software.

- La satisfacción del usuario es un factor importante en la usabilidad del software por lo que deben realizarse evaluaciones de satisfacción de usuario.
- Los usuarios reclutados para las pruebas deben representar el mercado objetivo del software. Es importante tener en cuenta la edad, género, nivel educativo y experiencia tecnológica de los usuarios para obtener resultados precisos.
- Los usuarios pueden tener necesidades y habilidades diferentes, por lo que es importante considerar la diversidad al preparar los escenarios de prueba.
- Grabar la pantalla y las interacciones del usuario puede ayudar a los evaluadores a revisar la prueba y a identificar problemas de usabilidad con mayor precisión.
- Después de identificar los problemas de usabilidad, es importante ser proactivo en la implementación de cambios. Los cambios deben ser evaluados para asegurarse de que se hayan abordado los problemas de usabilidad y no hayan creado nuevos problemas.
- Jakov Nielsen, considerado el padre de la usabilidad, dice que las pruebas de usabilidad muy elaboradas son un desperdicio de recursos. Los mejores resultados provienen de probar no más de 5 usuarios y ejecutar tantas pruebas pequeñas como se pueda. Por lo tanto, tenga esto en consideración cuando vea que se destinan muchos recursos en el contexto de su emprendimiento de software.

Herramientas Sugeridas

1. **UserTesting**: Es una plataforma en línea que te permite reclutar usuarios de todo el mundo para realizar pruebas de usabilidad. Los usuarios realizan tareas en el software mientras sus interacciones son grabadas y sus comentarios son recopilados para su análisis. https://www.usertesting.com/
2. **Hotjar**: Es una herramienta de análisis de comportamiento del usuario que te permite grabar la pantalla y las interacciones del usuario en tu sitio web o aplicación. Hotjar

también ofrece características de encuestas y análisis de retroalimentación de usuario. https://www.hotjar.com/

3. **UsabilityHub**: Es una plataforma en línea que ofrece pruebas de usabilidad rápidas y económicas, incluyendo pruebas de preferencia, pruebas de flujo de usuario y pruebas de navegación. UsabilityHub también ofrece herramientas de diseño de encuestas y retroalimentación de usuario. https://usabilityhub.com/

Es importante evaluar las necesidades de las pruebas de usabilidad antes de elegir una herramienta en particular, si corresponde. Cada herramienta tiene sus ventajas y desventajas, por lo que es importante investigar y evaluar varias opciones antes de tomar una decisión final.

Donde leer más

- J. Rubin and D. Chisnell, *Handbook of Usability Testing*, 2nd ed., Wiley.
- J. Nielsen, "Why You Only Need to Test with 5 Users," Nielsen Norman Group, 2000. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/
- S. Krug, *Don't Make Me Think, Revisited: A Common Sense Approach to Web and Mobile Usability*. New Riders, 2014.

ANEXO 11 - Guía de pruebas unitarias

Introducción

Las pruebas unitarias son una técnica utilizada en el desarrollo de software para verificar que una unidad de código, como una función o método, cumple su propósito y se comporta como se espera. Las técnicas de pruebas unitarias ayudan a mejorar la calidad del código, reducir los errores y acelerar el proceso de desarrollo.

Una buena prueba unitaria es una prueba automatizada que cumple las siguientes características:

- Verifica una pequeña pieza de código (también conocida como unidad).
- Las pruebas unitarias deberían ser rápidas, para de esa forma realizarlas de manera frecuente y de manera eficiente.
- Cada prueba unitaria debería ser independiente, siendo esto que no dependa del comportamiento de otras pruebas.
- Las unidades deben ser repetibles, significando esto que generen el mismo resultado cada vez que se ejecutan.
- Las pruebas deberían automáticamente determinar por sí mismas si han pasado o no las mismas.
- Deberían ser escritas de manera oportuna, preferentemente antes del código de producción para asegurar una adecuada cobertura y detección temprana de problemas.

El conjunto de estas propiedades conforma el acrónimo "FIRST", esto es conjunto de principios que este tipo de pruebas deberían seguir para garantizar que sean efectivas, confiables y mantengan un alto estándar de calidad.

¿Por qué debería incluirla en mi emprendimiento de software?

El objetivo principal es permitir el crecimiento sostenible del proyecto de software. El término sostenible es clave. Es bastante fácil hacer crecer un proyecto, especialmente cuando comienzas desde cero. Por el contrario, es muy difícil mantener este crecimiento en el tiempo.

Al comenzar un proyecto desde cero empiezas rápido porque no hay nada que te detenga. Es posible que todavía no se hayan tomado malas decisiones arquitectónicas, y no hay ningún código existente del que preocuparse.

Sin embargo, a medida que pasa el tiempo, debe dedicar más y más horas para lograr la misma cantidad de progreso que mostró al principio. Eventualmente, la velocidad de desarrollo se ralentiza significativamente, a veces incluso hasta el punto en que no puedes hacer ningún progreso en absoluto.

Este fenómeno de disminución rápida de la velocidad de desarrollo también se conoce como entropía del software. La entropía, que es la cantidad de desorden en un sistema, es un concepto matemático y científico que también se puede aplicar a los sistemas de software.

Las pruebas ayudan a revertir esta tendencia. Actúan como una red de seguridad, una herramienta que proporciona un seguro contra la gran mayoría de las regresiones.

Las pruebas ayudan a garantizar que la funcionalidad existente funcione, incluso después de introducir nuevas funciones o refactorizar el código.

¿En qué condiciones puedo aplicar esta técnica?

Las pruebas unitarias son aplicables en cualquier situación en la que se quiera asegurar la calidad del software y detectar errores temprano en el proceso de desarrollo. Es recomendable incluir pruebas unitarias como parte de la práctica general de desarrollo de software.

Proceso de Desarrollo

1. **Identifique la unidad:** Identifique la funcionalidad del código que se quiere probar.

2. **Defina el caso de prueba:** Defina el caso de prueba que desea cubrir en su prueba unitaria. Teniendo en cuenta entradas, salidas esperadas y casos bordes.

3. **Escriba la prueba:** Escriba la prueba que ejecutará los casos definidos en el punto anterior. La misma debe estar separada del código que está probando y debe utilizar un marco de pruebas que sea apropiado para su lenguaje de programación y entorno. Se sugiere utilizar el patrón AAA para escribir las mismas.

   El patrón AAA consiste en dividir cada prueba en tres secciones: organizar (*arrange*), actuar (*act*) y afirmar (*assert*).

   Este patrón proporciona una estructura simple y uniforme para todas las pruebas. Esta uniformidad es una de las mayores ventajas de este patrón: una vez que el usuario se familiariza lograra leer y comprender fácilmente cualquier prueba. Eso, a su vez, reduce los costos de mantenimiento para todo su conjunto de pruebas. La estructura del patrón es la siguiente:

   - En la sección de organizar, se organiza la prueba y sus dependencias a un estado deseado. Se inicializan todos los elementos que luego se utilizarán en la prueba.

   - En la sección actuar, es donde ocurre la interacción con las dependencias de forma controlada y se captura el valor de salida (si lo hay).

   - En la sección de afirmación, verifica el resultado. El resultado puede estar representado por el valor de retorno, el estado de una dependencia con la que se interactúa, entre otros.

4. **Ejecutar la prueba:** Ejecute la prueba y observe los resultados.

5. **Modifique el código:** Si la prueba falla, la prueba estuvo mal realizada o el sistema no se comporta de la manera esperada. En el último caso, debería modificar el sistema para que se comporte adecuadamente, para luego ejecutar las pruebas nuevamente.

6. **Iterar en el proceso:** Es posible iterar en los pasos anteriores mientras se refactoriza el código, hasta que se esté satisfecho con el resultado.

Consideraciones para aplicar la técnica:

- **Evite múltiples secciones de organizar, actuar y afirmar:** Ocasionalmente, puede encontrar una prueba con múltiples secciones de organizar, actuar o afirmar. Cuando vea múltiples secciones de estos actos, significa que la prueba verifica múltiples unidades de comportamiento. Y, como mencionamos anteriormente, tal prueba ya no es una prueba unitaria sino más bien una prueba de integración.
  Una sola acción garantiza que sus pruebas permanezcan dentro del ámbito de las pruebas unitarias, lo que significa que son simples, rápidas y fáciles de entender. Si ve una prueba en esta situación, será necesaria refactorizar. Extraer cada acción en una prueba propia.

- **Evitar declaraciones if en las pruebas:** De manera similar a las múltiples apariciones de las secciones de organizar, actuar y afirmar, algunas veces puede encontrar una prueba unitaria con una declaración if. Esto también es un anti-patrón. Una prueba unitaria, debe ser una secuencia simple de pasos sin bifurcaciones.

- **Desmontaje**: Podemos considerar el desmontaje como una cuarta sección, aunque no pertenezca a la aplicación del patrón anteriormente mencionado.
  Esta sección o actividad viene después de organizar, actuar y afirmar. Por ejemplo, puede usar esta sección para eliminar cualquier archivo creado por la prueba, cerrar una conexión a la base de datos, etc. El desmontaje generalmente se representa mediante un método separado que se reutiliza en todas las pruebas de la clase.

- **Reutilización de accesorios de prueba entre pruebas:** Es importante saber cómo y cuándo reutilizar el código entre pruebas, Por esto tiene sentido extraer algunas secciones en métodos o clases separados que luego se reutilizan entre pruebas es muy

útil. Un accesorio de prueba es un objeto contra el que se ejecuta la prueba. Este objeto puede ser una dependencia normal, también pueden ser datos en la base de datos o un archivo en el disco duro. Dicho objeto debe permanecer en un estado fijo conocido antes de cada ejecución de prueba, por lo que produce el mismo resultado ante la ejecución de la misma prueba.

- **Nombrar una prueba unitaria:** Siga las siguientes pautas para escribir nombres de prueba expresivos y fáciles de leer:
    - No siga una política de nombres rígida. No debe utilizar una descripción de alto nivel para un comportamiento complejo. Permitir la libertad de expresión.
    - Nombre la prueba como si estuviera describiendo el escenario a alguien que no es programador y que está familiarizado con el dominio del problema. Un experto en dominios o un analista de negocios es un buen ejemplo.
    - Separar palabras con guiones bajos. Hacerlo ayuda a mejorar la legibilidad, especialmente en nombres largos, en su defecto también podría usar la convención de nomenclatura *camelcase* o una combinación de ambos en nombres largos, procure ser coherente a través de las diferentes pruebas.

Donde leer más

V. Khorikov, *Unit Testing: Principles, Practices, and Patterns*. Manning Publications, 2017.

K. Beck, *Test-Driven Development: By Example*. Addison-Wesley Professional, 2002.

S. Freeman and N. Pryce, *Growing Object-Oriented Software, Guided by Tests*. Pearson Education, 2014.

ANEXO 12 - Guía de Wireframes

Introducción

El diseño de Wireframes es una parte fundamental del proceso de diseño de experiencia de usuario. Estos son planos básicos que ilustran la forma y función principales que se encuentran en una sola pantalla de su página web o aplicación. Son útiles porque permiten a los diseñadores y desarrolladores visualizar la estructura y el flujo del contenido antes de comenzar a trabajar en el diseño visual.

Al crear un wireframe, se puede determinar la ubicación y el tamaño aproximado de los elementos clave, como encabezados, menús de navegación y áreas de contenido. Esto ayuda a asegurarse de que el diseño final sea coherente y fácil de usar para los usuarios.

¿Por qué debería incluirla en mi emprendimiento de software?

Los Wireframes son una herramienta útil para visualizar la estructura y el flujo del contenido antes de comenzar a trabajar en el diseño visual. Esto puede ayudar a identificar problemas potenciales y mejorar la experiencia del usuario. Además, la utilización de wireframes puede ser una forma efectiva de comunicar su visión a otros miembros del equipo o a los clientes.

Proceso de Desarrollo

1. **Planificación y preparación:** Comprender los objetivos del proyecto y las necesidades de los usuarios es fundamental para crear Wireframes efectivos. Esto puede implicar realizar investigaciones de mercado, entrevistas con usuarios o análisis de datos.

   Hay muchas herramientas disponibles para crear Wireframes, desde lápiz y papel hasta software especializado. Es importante elegir una herramienta que se adapte a sus necesidades y habilidades.

Es importante recopilar toda la información relevante sobre el proyecto, como contenido, requisitos técnicos y limitaciones. Esto puede ayudarle a tomar decisiones al diseñar sus Wireframes.

2. **Crear un mapa de interacción y un diagrama de flujo:** Un mapa de interacción es una representación visual de la estructura general del sitio o aplicación. Puede ayudar a planificar cómo se relacionan las diferentes páginas o pantallas entre sí.

   Un diagrama de flujo muestra cómo los usuarios van a interactuar con el sitio o aplicación. Puede ayudar a identificar posibles problemas en la navegación o en la experiencia del usuario.

3. **Crear los Wireframes:** Los elementos individuales son los componentes básicos que componen un wireframe, como botones, formularios y menús. Es importante diseñar estos elementos con cuidado para asegurarse de que sean claros y fáciles de usar.

   Una vez que hayas diseñado los elementos individuales, es hora de organizarlos en una página para crear una estructura clara y fácil de usar. Es importante tener en cuenta la jerarquía visual y la legibilidad del contenido.

4. **Presentación y mejora:** En este punto es momento de presentar los Wireframes creados al equipo o al cliente para recibir comentarios. Con la intención de identificar problemas o áreas de mejora. Una vez que se reciban comentarios, es posible que se deba iterar y mejorar el diseño de sus Wireframes. Esto puede implicar hacer cambios en la estructura, los elementos individuales o la organización de los elementos en una página. Este proceso de iteración se debe llevar a cabo hasta que todos los involucrados estén satisfechos con el diseño final.

Consideraciones para aplicar la técnica:

- Comprender los objetivos del proyecto y las necesidades del usuario antes de comenzar a diseñar los Wireframes.

- Elegir una herramienta de diseño de Wireframes adecuada que se adapte a sus necesidades y habilidades del equipo.
- Recopilar toda la información relevante sobre el proyecto, como contenido, requisitos técnicos y limitaciones.
- Es muy importante antes de comenzar a crear un mapa de interacción y un diagrama de flujo para planificar cómo se relacionan las diferentes páginas o pantallas entre sí y cómo los usuarios interactúan con el sitio o aplicación.
- Diseñar elementos individuales que sean claros y fáciles de usar. Esto puede incluir elementos como botones, formularios, menús y otros elementos interactivos.

Donde leer más

- J. J. Garrett, *The Elements of User Experience: User-Centered Design for the Web and Beyond*. New Riders Publishing, 2002.
- M. J. Hamm, *Wireframing Essentials: A Beginner's Guide to Wireframing*. Packt Publishing, 2014. ISBN 978-1-78328-373-5.
- J. Beaird, *The Principles of Beautiful Web Design*. SitePoint Pty Ltd, 2010.

ANEXO 13 - Script contador de ocurrencias de técnicas de ingeniería de software

En el presente anexo, se proporciona un fragmento de código utilizado durante la fase de procesamiento y análisis de datos de la investigación. El propósito principal de este código es leer un archivo que contiene diversas técnicas de ingeniería de software, una por línea, y calcular la frecuencia de aparición de cada una de ellas. Posteriormente, se ordenan estas técnicas en función de su frecuencia y se imprimen los resultados.

```
from google.colab import drive
drive.mount('/content/drive')
# Inicializar un diccionario para guardar las ocurrencias de cada línea
line_counts = {}

# Abrir el archivo que contiene todas las técnicas, una por fila
with open('/content/drive/Mi unidad/Archivo.txt', 'r') as file:
    for line in file:
            # Eliminar espacios en blanco al inicio y fin de la línea y normalizar a lowercase
        line = line.strip().lower()

            # Si la línea no está en el diccionario, agregarla con el contador en 1
            # En caso contrario, incrementar el contador para esa línea
        if line not in line_counts:
            line_counts[line] = 1
        else:
            line_counts[line] += 1

# Ordenar el diccionario por valor en orden ascendente
sorted_dict = dict(sorted(line_counts.items(), key=lambda x: x[1], reverse=True))

# Imprimir los resultados
for line, count in sorted_dict.items():
    printf("{line}: {count} ocurrencia(s)")
```

ANEXO 14 - Formulario para validación de guías de apoyo por expertos del Grupo 1

Este formulario fue creado con la finalidad de obtener retroalimentación sobre las guías de apoyo que corresponden a las siguientes técnicas:

- **Pruebas AB**
- **Encuestas**
- **Mockups**
- **CI/CD**

La puntuación para cada pregunta del formulario se encuentra en el rango entre 1 y 5. En esta escala, el 1 se interpreta como "muy mal" y el 5 como "muy bien". Entre estos extremos, el 2 representa "mal", el 3 es considerado "neutral", y el 4 significa "bien".

Preguntas Generales

¿Cómo considera la estructura de las guías? Por estructura entendemos a las secciones de las mismas y su orden.

¿Cómo evalúa la presentación del proceso de desarrollo en las guías? Por proceso de desarrollo entendemos la sección donde se define la aplicación de la técnica en sí.

Preguntas sobre guía de Pruebas AB

¿Encuentra que el contenido de la guía de Pruebas AB es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de Pruebas AB es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de Pruebas AB?

Preguntas sobre guía de Encuestas

¿Encuentra que el contenido de la guía de Encuestas es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de Encuestas es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de Encuestas?

Preguntas sobre guía de Mockups

¿Encuentra que el contenido de la guía de Mockups es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de Mockups es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de Mockups?

Preguntas sobre guía de CI/CD

¿Encuentra que el contenido de la guía de CI/CD es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de CI/CD es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de CI/CD?

ANEXO 15 - Formulario para validación de guías de apoyo por expertos del Grupo 2

Este formulario fue creado con la finalidad de obtener retroalimentación sobre las guías de apoyo que corresponden a las siguientes técnicas:

- **Pruebas de Usabilidad**
- **Entrevistas**
- **Prototipos**
- **Observación**

La puntuación para cada pregunta del formulario se encuentra en el rango entre 1 y 5. En esta escala, el 1 se interpreta como "muy mal" y el 5 como "muy bien". Entre estos extremos, el 2 representa "mal", el 3 es considerado "neutral", y el 4 significa "bien".

Preguntas Generales

¿Cómo considera la estructura de las guías? Por estructura entendemos a las secciones de las mismas y su orden.

¿Cómo evalúa la presentación del proceso de desarrollo en las guías? Por proceso de desarrollo entendemos la sección donde se define la aplicación de la técnica en sí.

Preguntas sobre guía de Pruebas de Usabilidad

¿Encuentra que el contenido de la guía de Pruebas de Usabilidad es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de Pruebas de Usabilidad es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de Pruebas de Usabilidad?

Preguntas sobre guía de Entrevistas

¿Encuentra que el contenido de la guía de Entrevistas es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de entrevistas es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de entrevistas?

Preguntas sobre guía de Prototipos

¿Encuentra que el contenido de la guía de Prototipos es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de Prototipos es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de Prototipos?

Preguntas sobre guía de Observación

¿Encuentra que el contenido de la guía de observación es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de observación es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de observación?

ANEXO 16 - Formulario para validación de guías de apoyo por expertos del Grupo 3

Este formulario fue creado con la finalidad de obtener retroalimentación sobre las guías de apoyo que corresponden a las siguientes técnicas:

- **Estudio de casos**
- **Wireframes**
- **Pruebas unitarias**
- **Manejo de deuda técnica**

La puntuación para cada pregunta del formulario se encuentra en el rango entre 1 y 5. En esta escala, el 1 se interpreta como "muy mal" y el 5 como "muy bien". Entre estos extremos, el 2 representa "mal", el 3 es considerado "neutral", y el 4 significa "bien".

Preguntas Generales

¿Cómo considera la estructura de las guías? Por estructura entendemos a las secciones de las mismas y su orden.

¿Cómo evalúa la presentación del proceso de desarrollo en las guías? Por proceso de desarrollo entendemos la sección donde se define la aplicación de la técnica en sí.

Preguntas sobre guía de Estudio de casos

¿Encuentra que el contenido de la guía de estudio de casos es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de estudio de casos es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de estudio de casos?

Preguntas sobre guía de Wireframes

¿Encuentra que el contenido de la guía de Wireframes es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de Wireframes es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de Wireframes?

Preguntas sobre guía de Pruebas unitarias

¿Encuentra que el contenido de la guía de pruebas unitarias es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de pruebas unitarias es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de pruebas unitarias?

Preguntas sobre guía de Manejo de deuda técnica

¿Encuentra que el contenido de la guía de manejo de deuda técnica es relevante y aplicable para emprendimientos de software que buscan mejorar sus procesos de ingeniería de software?

¿Considera que la guía de manejo de deuda técnica es completa en términos de los aspectos que cubre?

¿Cómo evalúa la claridad de las explicaciones de la guía de manejo de deuda técnica?

ANEXO 17 - Transcripciones de entrevistas a expertos

Posterior a la realización de las encuestas presentes en los anexos Anexo 14, Anexo 15 y Anexo 16, realizamos entrevistas tomando como punto de partida a las mismas.

Cabe resaltar que estas transcripciones no son literales, sino que extrajimos las partes con más contenido. Sin más, estas son las siguientes:

Entrevista a Darío Machi

Entrevistador 1: *"Buenas noches, Darío, ¿Estás de acuerdo con que grabemos la llamada? Es para nosotros poder revisar luego, no tendrá exposición"*.

Entrevistado: *"Claro, no hay ningún problema"*.

Entrevistador 2: *"Buenas noches, Darío, gracias por atendernos. Para nosotros lo que nos puedes decir es muy valioso con el fin de mejorar las guías.*

Muchas gracias por llenar el formulario y para hoy preparamos cinco preguntas para hacerte brevemente. Quieres que te hagamos esas preguntas o quieres contarnos de manera general".

Entrevistado: "*Hagamos las preguntas si quieren y después comentamos cosas en general"*.

Entrevistador 1: "*Luego de revisar tus respuestas en el formulario que completaste vimos qué te pareció relativamente bien porque has respondido con una valoración de 4 y 5 todas las preguntas. Estas preguntas que te vamos a realizar no son específicas de cada una de esas guías, sino más bien en general"*.

Entrevistador 2: "*¿Pero, por ejemplo, la primera pregunta es qué te pareció la estructura de las guías? Es decir, las distintas secciones de estas"*.

Entrevistado: "*Bien, bien, te cuento. Me gustó lo que te decía recién que la estructura era como la misma en todas las guías que vi. Entonces creo que, si mantienes una misma estructura entre cada una de las guías, te permite fácilmente ir a qué parte te interesa. Después que conoces la forma, yo me imaginaba teniendo que usar estas guías con equipos*

míos de proyectos de fin de carrera, que tengan una startup y decirle la justificación, busquen en la sección por qué debería incluirla a mi emprendimiento de software. Y lo primero aprendan sea el por qué. Y entonces, cuando identifiquen un momento puntual en el cual están en esa misma situación, saben qué artefacto utilizar o qué técnica utilizar. Entonces es de mucha ayuda y está bueno que tengan esas partes bien, bien marcadas en la estructura.

Y como comentario general creo es lo único que me había marcado para decirles, es que estaría bueno en las referencias, si bien está bueno que ustedes no profundicen tanto en las guías, si me hubiese gustado ver referencias puntuales a cosas que no se explican en la misma guía. De esta manera en lugar de tener que googlear o preguntarle a chatGPT está duda, podía ir directamente a la referencia. Que ustedes digan, mira, acá esta es la referencia que te recomendamos nosotros".

Entrevistador 2: "*Después, otra pregunta que tenemos es: ¿Algo que te resultó confuso o poco claro en la lectura?".*

Entrevistado: "*Lo encontré muy sencillo de leer, muy rápido, de consumir, no encontré nada muy complejo o escrito de forma rebuscada".*

Entrevistador 2: "*Más allá de las cuatro técnicas que te enviamos. ¿Hay alguna otra cosa? ¿Algo que a vos te parece que es indispensable? Sabemos que no conoces las otras 8 guías".*

Entrevistado: "*Exacto. A ver, las que mencionamos están buenas. Sin duda entrevistas, prototipado y usabilidad. Creo que son todas súper útiles. Seguramente dentro de las que estén faltando, por ejemplo, deben tener el tema".*

Entrevistador 2: "*Te comento cuales son: A/B testing, CI/CD, Encuestas, Entrevistas, Estudio de casos, Manejo de deuda técnica, Mockups, Observación, Prototipo, Pruebas de Usabilidad, Pruebas Unitarias".*

Entrevistado: "*Se me ocurre análisis de competidores, o sea, no es de ingeniería, eso correo por el lado del emprendedurismo, es ver los competidores en el rubro en que estoy. Una vez*

que vos decidís que vas a construir esta bueno ver y revisar qué es lo que ya hay construido en el mercado".

Entrevistador 1: *"¿Eso que mencionas, no va por el lado del Estudio de casos?".*

Entrevistado: "*Si, puede ser, ahí te queda abierto, podría ser parte de esto que menciono. Es más a los grupos les recomiendo siempre, que lo comparen con lo producto más exitoso de su rubro, es más, para el caso que se trate de una app, lean las reviews de los stores, ven que piensa la gente de la aplicación del competidor. Se puede utilizar alguna técnica de teoría fundamentada para extraer estas opiniones".*

Entrevistador 2: "*Darío solo nos queda agradecerte por tu tiempo, es muy valioso lo que nos contas para mejorar los entregables".*

Entrevistado: "*Estoy disponible para lo que necesiten, si quieren mándenme el resto de las guías para poder verlas y ver si les puedo comentar algo".*

Entrevistador 2: "*Con gusto te las enviamos".*

Entrevista a Mario Camerota

Entrevistador 1: "*Respecto a la guía de encuestas, ¿Hay algún aspecto qué te parece que habría que mejorar respecto a la completitud? ¿Algo que faltó en esta guía?".*

Entrevistado: "*Las guías me resultaron muy familiares. Al tener en mi caso una carrera en licenciatura en sistemas, que está bien ligada a todas estas cosas, y también lo he aplicado a otros trabajos, por lo que me resultó muy familiar. No obstante, para mi gusto hay cosas que están muy detalladas, y nunca se me pasaron por la cabeza. Pero esto no quiere decir que sea incorrecto, sólo que yo no estaba en condiciones de implementarlo. Como guía me pareció bárbara".*

Entrevistador 1: "*En mockups y en CI/CD notamos como que de tu experiencia no te pareció algo tan importante para un emprendimiento que recién está surgiendo, ¿A qué se debe? ¿No se suele aplicar?".*

Entrevistado: "*Siempre hablaré conforme a mi experiencia, trataré de ser lo más sincero posible. Cuando yo inicie mi startup éramos dos, y éramos dos durante mucho tiempo. El costo humano, de horas trabajo, más el costo económico, tanto de herramientas para automatizar, y las horas hombre de recursos para automatizar procesos, se nos escapaban de las manos. Esa era nuestra realidad. Entendemos que está bueno, pero no teníamos recursos, necesitábamos hacer otras cosas, como validar el producto, conseguir ventas, ser sólidos en otros aspectos del negocio, como la transparencia, la seguridad, una cantidad de cosas. No digo que no sea importante, es algo que pensamos hacer ahora y nos permitirá escalar. En los 3 años de vida de la startup no pudimos centrarnos en la automatización. Tal vez si tienes suficiente inversión, ya de pique estás en otra situación y puede estar bueno".*

Entrevistador 1: "*¿Hay alguna información adicional que le agregarías a las guías para que sean más útiles?".*

Entrevistado: "*En base a mi experiencia, todo lo que leí fue muy rico. En su momento no sabía tanto de lo que leí. Me parece que es un material que está bueno tenerlo desde un inicio para ayudar. Así como te decía que no teníamos recursos ni dinero para automatizar cosas, si lo hubiésemos hecho antes nos habría ahorrado dinero. Me parece que las guías están super bien".*

Entrevistador 1: "*Mario, cuando hablabas de que utilizaban muchas encuestas, ¿Qué es lo que intentaban validar? ¿Intentaban validar la necesidad o la solución que tenían en la cabeza para esa necesidad que creían que existía?".*

Entrevistado: "*Sabemos que la necesidad existe. Lo que nosotros queríamos validar es si existía una herramienta que ayudará a la gestión de la organización de colectas y si esa persona estaría dispuesta a utilizarla. Si estuviera dispuesta a perder un porcentaje de comisión... más que perder... invertir en no tener que gestionarlo, en su beneficio, en no tener que manejar plata en efectivo, plata en su cuenta. Buscábamos para los organizadores saber si estaban dispuestos a usar una herramienta como la que ahora tenemos, y por otro lado buscábamos que los participantes si aparecería una herramienta que les facilitara participar en una colecta, si lo usarían. Sabíamos que la necesidad mayor estaba del lado del*

organizador, pero por otro lado hacer parte, o queríamos saber cómo hacer más parte al participante. Queríamos saber también cuánto participaban las personas de colectas durante el año".

Entrevistador 1: "*Más allá de las cuatro guías que te mandamos, ¿Hay alguna otra técnica que te parece que te parece que estaría bueno agregar?".*

Entrevistado: "*Lo nuestro fue todo medio a mano, medio a pulmón. No sé técnicamente el nombre de las guías, pero puedo asegurarles de que hemos intentado hacer un trabajo muy evolutivo. Si me hubiera gustado haber trabajado antes en el tema de experiencia de usuario (UX). Las personas se van o vienen dependiendo de su experiencia. Ahora contratamos una empresa que nos está dando una página diferente, con una navegabilidad diferente, mucho más fresca. Ayer llegamos a las 20000 transacciones realizadas en la plataforma. Pero tenemos una cantidad similar de transacciones canceladas. Si hubiéramos tenido una mejor experiencia de usuario tal vez no habríamos perdido tanto. Nosotros no tuvimos un inversor ángel o cualquier tipo de financiamiento, por lo que fuimos muy paso a paso. Tal vez de haberlo tenido habríamos podido encarar más la parte que ustedes están trabajando".*

Entrevistador 1: "*Claro... también creo que cuando queres hacer algo de una forma, aparecen muchas formas de hacerlo y al no conocer, te abruma".*

Entrevistado: "*Es tan importante eso, que si yo tuviera que decir que estaría bueno que estén en estas guías, tal vez estaría bueno que se les dé una clasificación. Por ejemplo, estas guías están buenas que están en esta etapa, estas otras en esta otra etapa, porque quizás necesitas tal o tal cosa. A mí me abruma todo eso, empiezo a buscar herramientas para gestionar y me aparecen mil, no sé cuál utilizar, por lo que etiquetar las guías de esa manera puede estar bueno".*

Entrevistador 1: "*Una última pregunta, ¿Te parece que este trabajo le puede servir a alguien?".*

Entrevistado: "*Cuando yo estuve en la incubadora del CIE nadie me dijo nada referido a estás técnicas de ingeniería de software, y creo que ayuda y optimiza. Muchos emprendedores están perdidos en este sentido. Así que seguro sirve".*

Entrevistador 1: "*Muchas gracias, Mario por todo*".

Entrevista a Pablo Benítez

Entrevistador: *"¿Qué te pareció la estructura de las guías?"*

Entrevistado: "*La estructura es consistente, está bueno que así lo sea. En la guía de deuda técnica sugieren herramientas, y en las otras no. No sé si es porque no aplica. ¿Por qué no pusieron las otras?"*

Entrevistador: "*Encontramos herramientas para casi todas las técnicas, el tema es que no sabíamos si recomendar herramientas que nosotros no conocíamos."*

Entrevistado: *"Eso fue lo único raro en cuanto a la estructura."*

Entrevistador: *"¿Te parece que hubiese estado bueno que hayamos recomendado más herramientas?"*

Entrevistado: *"Creo que eso puede allanar más el camino. Por ejemplo, estaría bueno que haya herramientas para la guía de Wireframes. Luego en la parte de estudio de casos me resultó interesante porque no es tanto del palo de la ingeniería... Esta guía no me resultó novedosa, pero nunca la vi aplicada a ingeniería de software. Siempre la vi en temas de negocio. ¿En Uruguay se utiliza esto?"*

Entrevistador: "*Eso salió en el relevamiento que utilizamos, de startups de otros países del mundo. En Uruguay no trabajamos con ningún paper. Esto va más por el lado de la elicitación de requerimientos."*

Entrevistado: *"Creo que todas las guías están muy claras, tienen una introducción, para que las pueden utilizar (a veces hay que salir al mercado y no se sabe que utilizar), y el proceso, que está clarito. El proceso de estudio de casos me pareció bastante complejo, no sé cuánto llevara aplicarlo en su totalidad. Pero más allá de eso me parece que todas están bien para mí."*

Entrevistador: *"¿Habría alguna información adicional que te parece útil que este?"*

Entrevistado: *"Lo único que le agregaría es el número de página."*

Entrevistado: *"Otro tema que veo es que las personas que apliquen estas guías deben ser técnicas, no son aptas para alguien no tan técnico."*

Entrevistador: "*¿Hay alguna técnica que te parezca que deba tener una guía?"*

Entrevistado: *"Me parece que algo relacionado a los sistemas de control de versiones (VCM), no solamente para código, sino también para documentación. Me parece importante, pero si ya tienen 12 guías, tal vez es suficiente."*

Entrevistador: "*La idea es que esto sea evolutivo, por eso nos interesa las opiniones que puedas darnos, en este caso respecto a nuevas guías."*

Entrevistado: *"Algo sobre proceso de desarrollo, como LEAN, estaría bueno que pueda agregarse."*

Entrevistador: *"Otra pregunta, ¿Qué te parece este proyecto que estamos realizando?"*

Entrevistado: *"Me parece muy bien"*

Entrevistador: *"Muchas gracias, Pablo"*

Entrevista Rodrigo Stratta

Entrevistador: *"Buenas noches, Rodrigo, gracias por atendernos. Para nosotros lo que nos puedes decir es muy valioso con el fin de mejorar las guías. Muchas gracias por llenar el formulario y para hoy preparamos cinco preguntas para hacerte brevemente. Quieres que te hagamos esas preguntas o quieres contarnos de manera general."*

Entrevistado: *"Coméntame las preguntas que prepararon."*

Entrevistador: *"¿Qué le pareció la estructura de las guías?"*

Entrevistado: *"Con respecto a la completitud me parece que están bien, me pareció muy bien explicado."*

Entrevistador: *"¿Hubo algún contenido que le resultara confuso o poco claro?"*

Entrevistado: *"No, la lectura me resultó fácil, estaba claro, se me hizo fácil leerlas."*

Entrevistador: *"¿Sugiere agregar algún otro tipo de información a la guía?"*

Entrevistado: *"A nosotros lo que nos resulta útil es automatizar las pruebas de QA, se podría entrar en eso"*

Entrevistador: *"¿Estas automatizaciones, en qué etapa de su proyecto pudieron incorporarlas?"*

Entrevistado: *"Claramente al principio no, no teníamos los recursos ni la gente para hacerlo, éramos cuatro personas, todos hacíamos todo y de manera manual. Pudimos incorporar estas automatizaciones con el sistema ya en producción y estable."*

Entrevistador: *"¿Sugiere alguna otra técnica de ingeniería de software que debería incluirse en estas guías? Nosotros te enviamos 4 de las 12 guías para que no resultara tan pesado el trabajo de leerlas."*

Entrevistado: *"Me parece útil que se brinde alguna herramienta financiera, nosotros cuando arrancamos tuvimos un cuello de botella ahí, ya que nuestro proyecto maneja dinero."*

Entrevistado: *"¿Cuáles son las otras técnicas que hicieron?"*

Entrevistador: *"Las 12 técnicas son: A/B testing, CI/CD, Encuestas, Entrevistas, Estudio de casos, Manejo de deuda técnica, Mockups, Observación, Prototipo, Pruebas de Usabilidad, Pruebas Unitarias."*

Entrevistado: *"Me parecen bien, por ejemplo, Wireframes no se bien que es y testing A/B nunca aplicamos, el resto me parecen muy bien y útiles."*

Entrevistador: *“Por último, ¿Qué opinión tiene sobre el proyecto que estamos llevando a cabo?”*

Entrevistado: *“Pienso que es muy útil, es un acercamiento a las opciones que hay y a veces uno no conoce cuando está arrancando. Si quieren me pueden mandar el resto de las guías, las miro y veo si les puedo comentar algo.”*

Entrevistador: *“Muchas gracias, Rodrigo por tu tiempo, Con gusto te las enviamos.”*

Entrevista a Andrés Bentos

Entrevistador 1: *“Buenas tardes, Andrés, gracias por tu tiempo, Tenemos algunas preguntas más, son breves.”*

Entrevistado: *“No hay problema, en lo que pueda ayudar”*

Entrevistador 2: *“También queríamos consultarte si estás de acuerdo con que grabemos la llamada, es para analizarla luego, no tendrá exposición.”*

Entrevistado: *“Claro, no hay problema”*

Entrevistador 1: *“Con respecto a la guía de pruebas A/b, a partir de tu votación pusiste que no era muy aplicable o relevante. ¿Era por algo en particular tu opinión sobre la técnica en sí?"*

Entrevistado: *“Creo que las pruebas, a ver si no me falla la memoria, son cuando vos introducís cambios en un sistema operando y medís la interacción del público objetivo para ver si determinado feature tiene el comportamiento deseado o no, o qué comportamiento tiene exactamente. Está bueno. A lo que me refería con ese comentario es que en la práctica está muy bueno y el resultado que te da es muy bueno. Pero creo que tiene que estar bastante avanzada la startup para tener un equipo o un producto. Porque tampoco puede poner una prueba muy grande, porque puede ser muy costoso hacer la prueba, a eso me refería. Es importante que no sea más costoso el hacer la prueba y el riesgo que asumís al colocar una feature que capaz que puede causar un error o no, y rechazo del público. Esto es algo que me*

parece que está bueno. Resumiendo, la técnica es muy útil, hay que tener cuidado de en qué situaciones la aplico."

Entrevistador 1: *"Buenísimo, genial. Gracias Andrés."*

Entrevistador 1: *"¿Hay alguna información adicional que te hubiera gustado ver en las guías o qué te parece que estaría bueno que esté?"*

Entrevistado: *"Me parece que están bien las guías, pero creo que, respecto a eso, información adicional, glosario o algo de terminología que ustedes en algún relevamiento o conversando con gente que de repente no es tecnológica, entiendan que puede necesitar, capaz que no es muy grande esta lista y tal vez es eso es que vos tengas en las guías tenga todo lo que necesitas para entenderla. No tenés que escribir un manual, pero asegúrate de eso."*

Entrevistador 1: *"Genial, genial. ¿Después de lo que recordás, de lo que leíste, hubo algo que te resultara que estaba muy enredado o confuso o poco claro? "*

Entrevistado: *"No, no, estaban bastante claras. Concretas. Va paso a paso. Creo que no Puede ser que haya algo que se entendí que capaces de simplificar se requería, pero me parece que es eso. Fue fácil de leer, entendible, así que te diría que no, por lo menos lo que recuerdo ahora está genial."*

Entrevistador 1: *"Después teníamos otra pregunta, ¿se te ocurre alguna otra técnica más allá de que no las conoces todas?"*

Entrevistado: *"Mira, Contame cuales son y las conversamos."*

Entrevistador 1: "*Son A/B testing, CI/CD, Encuestas, Entrevistas, Estudio de casos, Manejo de deuda técnica, Mockups, Observación, Prototipo, Pruebas de Usabilidad, Pruebas Unitarias."*

Entrevistado: *"Me parece que esas técnicas están bien."*

Entrevistado: *"Lo primero que veo es que son todas complementarias. No da mucho sentido hacer una sin la otra, como que no te podrías enfocar. Yo pienso que eso se puede hacer, pero las pruebas unitarias las hacen para lograr una característica en el sistema que sea que permita que sea más modificable o que por lo menos tenga menos riesgos al meter errores cuando haces un cambio después."*

Entrevistado: *"¿Para quienes están enfocadas estas guías?"*

Entrevistador 1: *"En realidad está enfocado principalmente a equipos de tesis, también a startups. Y una cosa que me comentaba y que la conversamos es que para hacer las guías partimos de la base de que tenés idea de lo que estamos hablando. Las guías asumen que el lector tiene bases de ingeniería de software."*

Entrevistado: "*Yo lo preguntaba porque recordaba que una vez que hice un trabajo similar de investigación relacionado con emprendimientos. Creo que era el 50 o el 60% de los emprendimientos. No tenían un socio tecnológico. Entonces vos hablas con gente que sabe mucho de negocios, pero no tiene las capacidades para implementarlas, pero sabiendo que son personas con poco conocimiento técnico, me parece genial"*

Entrevistador 1: ***"****Ahora creo que teníamos una última pregunta y era tu opinión. ¿Panzas que este proyecto va a servir? ¿Aporta o, así como está? ¿Te parece que no aporta, que esta información es moneda corriente? ¿Cuál es tu opinión?"*

Entrevistado: "*Bueno, sinceramente, mi opinión es que siempre cualquier cosa que vos hagas aporta, sea un resumen o cualquier cosa, porque la gente siempre interpreta distinto. Entonces yo capaz que hoy lo veo, digo si esto es así, pero capaz que mañana alguien que de repente con otro background técnico y que hace un emprendimiento lo bicha y dice bueno, voy a seguir esta guía, yo creo que va a servir y en la medida en que esto se pueda probar y se pueda arreglar, lo que es seguro, si ustedes continúan conectados con estas guías, lo que es seguro que en dos años las ideas van a ser totalmente distintas. Pero bueno, alguien tiene que hacer la primera guía para empezar a evolucionar, la de ahí. Entonces para mí va a ser útil el consolidar información práctica, concisa, escrita, clara, con un toque adecuado de academia, es decir, de una buena redacción y con un buen contenido que tenga sentido y que no se vaya por las ramas. Siempre suma, sobre todo cuando hay que organizar un equipo o*

tomar decisiones del tipo cómo construir tu producto o tu servicio. Lo que siempre creo que hay que tener presente es los costos asociados a esas decisiones que muchas veces que por querer hacer una metodología o querer encarar el trabajo de una forma, no los termines ahogando en algo que no sea hacer tu producto. Porque siempre acuérdense que los emprendimientos tecnológicos o por lo menos lo que yo siempre pienso es que lo primero es siempre encontrar lo que quiere el cliente, lo que le agrega valor, lo que el cliente estaría dispuesto a pagar, lo que al cliente más le duele. Entonces creo que teniendo siempre la seriedad de lo que nosotros agreguemos, nosotros debemos servir como soporte a aquel que va a hacer algo. No se complejiza más ese proceso, porque el proceso es más complicado."

Entrevistador 1: **"***Muchas gracias por todo Andrés."*

ANEXO 18 - Transcripción de entrevista a equipo de proyecto en conjunto con industria

Entrevistador: "*¿Qué técnicas pudieron implementar?".*

Entrevistado: "*Utilizamos las técnicas de Mockups y la de Prototipos".*

Entrevistador: *"¿Cómo les fue en el proceso de implementación? ¿Fue sencillo seguir las guías?".*

Entrevistado: "*Nosotros teníamos experiencia previa con ambas técnicas, y consideramos que son tremendas herramientas. Leyendo lo que ustedes fueron escribiendo pudimos confirmar que hicimos lo que ustedes plantearon. Nos juntamos con Diego e hicimos bocetos, cada uno por la suya, y luego vimos que nos gustaba de ambas partes. Luego nos juntamos con el cliente, y volvimos a hacer lo mismo entre los tres. En un pizarrón empezamos a hacer los flujos, y luego de discutirlo, empezaron a surgir problemas, donde a base de dibujos, pudimos verlos y ver que queríamos. Ahí terminó esa parte para esas pantallas. Luego utilizando Adobe XD empezamos a plasmar esas ideas. Tuvimos un feedback constante con el cliente y así obtener retroalimentación. Estamos en una etapa media en el proyecto, ya que el objetivo final es tener todo prototipado con el usuario final, y ver como interactúa y reacciona el usuario final, haciendo los cambios necesarios para que la aplicación se adapte a eso".*

Entrevistador: "*¿En qué etapa del proyecto están?".*

Entrevistado: "*Estamos en la etapa de desarrollo. Ya pasamos la etapa del entendimiento del proyecto. Para llegar a este punto tuvimos que entender muchas cosas. Definimos un objetivo en el que teniendo un buen entendimiento, empezamos a definir, y luego empezamos a hacer mockups".*

Entrevistador: "*El proceso de ustedes entonces fue bastante parecido a las guías".*

Entrevistado: "*Nosotros ya conocíamos mockups, pero fue una ayuda de memoria, para refrescar nuestros conocimientos."*

Entrevistador: "*Creen que, si ustedes no tuvieran conocimientos de Mockups, ¿Les hubiera servido? ¿Es clara la guía? ¿Se entiende?".*

Entrevistado: "*Yo creo que maneja un lenguaje muy entendible, y puede ser aún más enriquecedora para alguien que recién está arrancando".*

Entrevistador: "*¿Hubo algún obstáculo o desafío durante la implementación en alguna de las guías utilizadas?".*

Entrevistado:" Al *principio costó empezar a definir los mockups, la parte de definir los componentes no la hicimos y empezamos medio al vuelo. Para tener mejores ideas lo que hicimos fue ver aplicaciones similares. En base a eso, pudimos ver que funcionaba, que estaba bueno, y en base a eso lo plasmamos en los bocetos finales. Por ejemplo, de esta aplicación me gusta esto y esto, para resolver esta funcionalidad me gusta esto y aquello, ese enfoque nos gustó mucho".*

Entrevistador: "*Tenemos una guía para esa necesidad, se llama estudios de caso la técnica".*

Entrevistador: *"¿Sintieron que la guía es completa, o consideran que le faltaba información?".*

Entrevistado: "*En la de Mockups cubre los puntos que debe cubrir, ya que, si entrara mucho en detalle, deja de ser útil. Si es más genérica la guía, se puede adaptar a muchos escenarios. Una de las cosas que hicimos cuando empezamos a hacer los mockups fue usar crazy eight, pero, si en la guía me obligan a usar algo así, tal vez no la usaría. Prefiero la flexibilidad.*

Puedo asegurarte de que las guías funcionan muy bien. Siguiendo las guías, siguiendo su camino, vas a lograr que un abanico de ideas se convierta en el problema que quieres resolver, solucionándolo. Yo considero que es el camino".

Entrevistador: "*¿Cuáles fueron todas las técnicas que les dio Darío?"*

Entrevistado: "*Entrevistas, wireframe, mockups y prototipos".*

Entrevistador: ¿Por qué decidieron no implementar Entrevistas y Wireframes?

Entrevistado: *Podríamos ver la guía de apoyo de entrevista, pero no hemos llegado aún a esa parte. No llegamos al punto en que vamos a preguntar al usuario para entender su contexto, y en base a eso tener referencias de por dónde avanzar. Lo que hicimos en la primera parte fue entender lo que quería el cliente y a donde iba apuntado, pero no llegamos a entrevistar".*

Entrevistador: "*¿Hay algunas técnicas que consideren que pudieran implementar en el futuro cercano? No necesariamente debe ser alguna de las que les ha dado Darío, técnica que vean que quizás deban ver cómo se implementan".*

Entrevistado: "*Algo que hemos hecho en la tesis son técnicas de gestión de riesgos".*

Entrevistador: "*¿Recomendarían estas guías a otras startups o personas?".*

Entrevistado: "*Si no tuvieran conocimiento alguno, si, obvio. Ahora, si tienen conocimiento, qué conocimiento, y en caso de que, si lo tengan, apuntaría a algo mucho más complejo, como una técnica específica. Por ejemplo, tengo este problema, tienes esta solución, que es esta técnica. Pero creo que cada caso es diferente. Esta guía tiene su paso a paso, es más general, y puede que en nuestro caso podemos decir que la guía es válida, pero si alguien se encuentra en un caso puntual, tal vez la guía no lo resuelve tanto".*

Entrevistador: *"Tal vez podría haber una sección en la guía donde mostramos la generalidad, y luego casos más específicos."*

Entrevistado: *"Agregar una sección donde haya distintas técnicas específicas. Por ejemplo, prototipos estaba conectada con su guía de apoyo de Wireframes."*

Entrevistador: *"¿Tienen algún comentario o sugerencia para mejorar las guías?"*

Entrevistado: *"No se nos ocurre ahora"*

Entrevistador: *"¿Ustedes no usaron tantos prototipos?"*

Entrevistado: *"No la usamos tanto porque aún no pudimos llegar a validar los prototipos con el cliente, pero nos resultó muy útil."*

ANEXO 19 - Agrupamiento de técnicas de ingeniería de software

Aquí se detallan las decisiones que fueron necesarias tomar en cuanto a agrupar técnicas que implican la misma o similar acción pero que fueron recabadas de los documentos de manera fiel con distintos nombres, pudiendo ser variaciones tanto en inglés como en español, de esta forma lo que se busca es unificar un criterio y de este modo hacer el recuento de forma más afinada.

Entrevistas agrupa a las siguientes técnicas:

- Semi-structured interviews
- Interviews
- Structured interviews
- Face-to-face interviews
- Customer interviews
- Personal interviews
- Entrevistas en profundidad
- Entrevistas con expertos
- Entrevista como técnica de recolección de datos

Prototipos agrupa a las siguientes técnicas:

- Paper prototypes
- Low-fidelity prototyping
- Paper prototyping

- Piggyback prototyping
- Revisiones/prototipos internos
- Enfoque de prototipado evolutivo
- High-fidelity prototyping
- High-fidelity prototype
- Prototyping: proof of concept
- Extensive prototyping
- Digital prototypes

Ingeniería de requerimientos agrupa a las siguientes técnicas:

- Enfoque ad-hoc en ingeniería de requisitos
- Clasificación de requisitos
- Selección de requisitos
- Análisis de requerimientos
- Validación de requerimientos
- Priorización de requisitos con mayor impacto en el cliente
- Managing requirements through constant planning
- Métodos ad-hoc para la elicitación de requerimientos

Análisis de datos agrupa a las siguientes técnicas:

- Análisis de datos operacionales

- Análisis de datos de uso
- Análisis de datos en equipo y revisión por pares
- Análisis de datos y síntesis
- Análisis temático como técnica de análisis de datos
- Big data
- Online data collection
- Usage data analysis

Retroalimentación con el cliente agrupa a las siguientes técnicas:

- Customer feedback technique (CFT)
- Customer feedback (validated learning)
- Recopilación de comentarios

Gestión de riesgos agrupa a las siguientes técnicas:

- Mitigación de riesgos
- Quantitative risk calculations
- Risk identification grid

Experimentación agrupa a las siguientes técnicas:

- Business experiments navigator (BEN)
- Continuous experimentation
- Experimentación en ingeniería de software

- Experimentación y aprendizaje continuos
- Experimentación de modelos de negocio
- Experimentos controlados
- Experiment-driven software requirements
- Experimentation machine
- Experiment driven development (EDD)

Experiencia de usuario agrupa a las siguientes técnicas:

- User experience (UX)
- Diseño UX
- Experiencia de usuario mínima viable (MVUX)
- Diseño de la experiencia del usuario
- Minimum viable user experience (MVUX)

Observación agrupa a las siguientes técnicas:

- Observación no participante
- Observación de comportamiento
- User observation

Métricas agrupa a las siguientes técnicas:

- Métrica funcionalidad
- Métrica reusabilidad
- Métrica usabilidad

- Métrica mantenibilidad
- Métrica portabilidad
- Definición de métricas
- Goal question metric (GQM)
- Three engines framework
- Pirate metrics framework
- Metric analysis

Manejo de deuda técnica agrupa a las siguientes técnicas:

- Technical debt management
- Prevención del endeudamiento técnico
- Identificación del endeudamiento técnico
- Pago del endeudamiento técnico

Estudio de casos agrupa a las siguientes técnicas:

- Case study
- Case studies
- Desarrollo de descripciones de casos
- Análisis transversal de casos
- Método de encuesta de casos
- *Purposive sampling*

- Análisis cruzado de casos
- Metodología de estudio de caso

Pruebas de validación agrupa a las siguientes técnicas:

- Validación de que su producto se vende
- Validación y prueba de nuevas ideas
- Validación de la viabilidad del mercado
- Validación en el entorno de uso previsto
- Validación de soluciones
- Validación por parte de los usuarios finales
- Identificación y validación de problemas

Ingeniería de hipótesis agrupa a las siguientes técnicas:

- Elicitación de hipótesis
- Priorización de hipótesis
- Especificación de hipótesis
- Análisis de hipótesis
- Gestión de hipótesis
- Lista de hipótesis
- Uso de hipótesis falsificables
- Métodos para confirmar las hipótesis de la efectividad del modelo de negocio
- Hipótesis de valor

ANEXO 20 – Subconjunto de Artículos Relevantes

Los artículos académicos que figuran a continuación son aquellos de los cuales pudimos extraer técnicas de ingeniería de software, que luego fueron utilizadas a lo largo del proyecto. Las mismas no pretenden ser una cita bibliográfica, sino una simple mención a dichos artículos.

Archivo	Año	Autor	Publicador/Instituto
00 The role of process in software start-up	2000	Stanley M. Sutton, Jr.,	IEE Software
08 An investigation into software development process formation in software start-ups	2008	Gerry Coleman1 and Rory V	School of Computing, Dublin City University,
12 Applying Lean Startup. An Experience Report	2012	Beverly May	2012 Agile Conference - New York, NY
12 Lean Software Development. A Tutorial	2012	Mary Poppendieck,	Massachusetts Institute of Technology
12 Software Value Map	2012	Mahvish Khurum, Tony Gorschek	Published online in Wiley Online Library (wileyonlinelibrary.com).
13 Creating Minimum Viable Products in Industry-Academia	2013	Jurgen Munch, Fabian Fagerholm	Department of Computer Science, University of Helsinki
13 Lean Product Development in Early Stage Startups	2013	Jens Björk, Jens Ljungblad	University of Technology Göteborg, Sweden
13 The Early Stage Software Startup Development Model	2013	Jens Björk, Helena Holmström	University of Technology Göteborg, Sweden
14 Why Early-Stage Software Startups Fail. A Behavioral Framework	2014	Carmine Giardino, Xiaofeng Wang,and Pekka Abrahamsson	University of Bolzano
15 A Conceptual Framework of Lean Startup Enabled Internal	2015	Henry Edison	University of Bolzano
15 Design Thinking in Engineering Education and its adoption in technology-driven startups	2015	Daniel Rother	Springer-Verlag Berlin Heidelberg
15 Early Product Design in Startups. Towards a UX Strategy	2015	Laura Hokkanen, Kati Kuusinen, and Kaisa Väänänen	Tampere University of Technology, Korkeakoulunkatu
15 Early-Stage Software Startup Patterns	2015	Daniel Cukier	University of São Paulo
15 Hunter-gatherer cycle. A conceptual model of the evolution of software startups	2015	Anh Nguyen, Pekka Abrahamsson	Department of Computer and Information Science,Trondheim, Norway
15 Key Challenges in Early-Stage Software Startups	2015	Carmine Giardino, Sohaib Shahid Bajwa	University of Bolzano
15 Lean Software Startup. An Experience Report	2015	Antero Järvi, Ville Taajamaa, and Sami Hyrynsalmi	Department of Information Technology, University of Turku, Turku, Finland
15 Seven Patterns for Software Startups	2015	Jorge Melegati, Alfredo Golman	University of São Paulo

15 UX Work in Startups. Current Practices and Future Needs	2015	Laura Hokkanen and Kaisa Väänänen-Vainio-Mattila	Department of Pervasive Computing, Tampere University of Technology, Tampere, Finland
16 Eight Paths of Innovations in a Lean Startup manner	2016	Mikko Raatikainen, Marko Komssi, Harri Kiljander, Laura Hokkanen, Jukka Marijarvi, and Omar Mohout	Department of Pervasive Computing, Tampere University of Technology, Tampere, Finland
16 Guide for Minimal Viable Product (MVP) Testing in lean develipment cycle	2016	Lin Chou Cheng	School of Informatics & IT, Singapore.
16 InnoStartup. A Toolbox for Innovation in software development process	2016	A. W. T. Borba, G. H. C. Batista and R. A. C. Souza	IEEE LATIN AMERICA TRANSACTIONS, VOL. 14, NO. 8, AUG. 2016
16 Minimum Viable Product or Multiple Facet Product. The Role of MVP in Software Startups	2016	Anh Nguyen Duc and Pekka Abrahamsson	Department of Computer and Information Science,Trondheim, Norway
16 Minimum Viable User EXperience	2016	Laura Hokkanen, Kati Kuusinen, and Kaisa Väänänen	Department of Pervasive Computing, Tampere University of Technology, Tampere, Finland
16 MVP Explained	2016	Valentina Lenarduzzi, Davide Taibi	Faculty of Computer Science, University of Bolzano/Bozen, Bolzano/Bozen - Italy
16 Requirements Engineering in Software Startups. A Grounded Theory approach	2016	Jorge Melegati and Alfredo Goldman	Department of Computer Science University of São Paulo São Paulo, Brazil
16 Software Development in Startup Companies. The Greenfield Startup Model	2016	Carmine Giardino, Nicolo Paternoster, Michael Unterkalmsteiner,	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 42, NO. 6, JUNE 2016
16 Software Product Innovation through Startup Experimentation in large companies	2016	Henry Edison	University of Bozen-Bolzano, Bolzano, Italy
16 Software Startups. A Research Agenda	2016	Michael Unterkalmsteiner,Pekka Abrahamsson,XiaoFeng Wang	Software Engineering Journal, Volume 10, 2016
16 The Developers Dilemma. Perfect Product Development or Fast Business validation	2016	Henri Terho, Sampo Suonsyrja, and Kari Systa	Tampere University of Technology, Tampere, Finland
16 Towards an early stage software startup evolution model	2016	Anh Nguyen Duc and Pekka Abrahamsson	Department of Computer and Information Science,Trondheim, Norway
17 An empirical study on software engineering and software startups	2017	Leandro Pompermaier, Afonso Sales, Rafael Chanin	Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, RS, Brazil
17 Applying Customer Development for Software requirements in a startup	2017	Rafael Chanin, Leandro Pompermaier, Kellen Fraga, Afonso Sales and Rafael Prikladnicki	Computer Science Department, PUCRS, Porto Alegre, RS, Brazil
17 Are Software Startups Applying Agile Practices	2017	Jevgenija Pantiuchina, Marco Mondini, Dron Khanna, Xiaofeng Wang and Pekka Abrahamsson	Faculty of Computer Science, University of Bolzano/Bozen, Bolzano/Bozen - Italy
17 Estudio descriptivo de las prácticas de diseño y arquitectura	2017	Guillermo Pizarro	Revista Ciencia UNEMI
17 Evolution of Software Product Development in startup companies	2017	Mercy Njima, Serge Demeyer	Department of Mathematics and Computer Science,University of Antwerp, Antwerp, Belgium
17 Exploring the Applicability of Software Startup Patterns in the Ugandan Context	2017	Grace Kamulegeya, Raymond Mugwanya	Makerere University, Kampala, Uganda

17 Exploring the outsourcing relationship in software startups	2017	Anh Nguyen Duc and Pekka Abrahamsson	Department of Computer and Information Science,Trondheim, Norway
17 How are Product Ideas Validated	2017	Pertti Seppänen, Nirnaya Tripathi, Markku Oivo and Kari Liukkunen	University of Oulu, Oulu, Finland
17 How do entrepreneurs think they create value	2017	Dennis Lyth Frederiksen & Alexander Brem	Springer Science+Business Media New York
17 How do Software Startups Approach Experimentation	2017	Matthias Gutbrod, Jürgen Münch, and Matthias Tichy	Institute of Software Engineering and Programming Languages, Ulm University, Ulm, Germany
17 Miniumum Viable Product Creation through adaptive project management	2017	Ala Nuseibah, Christian Reimann, Maria Zadnepryanets	IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems:
17 Requirements elicitation techniques applied in software startups	2017	Usman Rafiq, Sohaib Shahid Bajwa, Xiaofeng Wang	Faculty of Computer Science, University of Bolzano/Bozen, Bolzano/Bozen - Italy
17 Risk management in startups	2017	S. Tolun tayali, S. Gulcesen	Meltepe University, Istambul, Turkey
17 Software Engineering in Startups	2017	Renata Souza, Karla Malta, Eduardo Santana de Almeida	Federal University of Bahia Department of Computer Science Salvador, Brazil
17 Startup Framework based on Scrum Framework	2017	Sirin Thongsukh, Smitti Darakorn Na Ayuthaya, Supaporn kiattisin	Faculty of Engineering Mahidol University Nakhon Pathom, Thailand
17 Startups and technical debt	2017	Marcos Chicote	2017 IEEE/ACM 1st International Workshop on Software Engineering for Startups (SoftStart)
17 The RIGHT model for Continuous Experimentation	2017	Fabian Fagerholma,Alejandro Sanchez Guinea, Hanna Mäenpää, Jürgen Müncha	The Journal of Systems and Software
17 Towards Understanding Startup Product Development as Effectual	2017	Anh Nguyen Duc and Pekka Abrahamsson	Department of Computer and Information Science,Trondheim, Norway
17 What Influences the Speed of Prototyping	2017	Anh Nguyen Duc and Pekka Abrahamsson	Department of Computer and Information Science,Trondheim, Norway
17 “Failures” to be celebrated. An analysis of major pivots of software startups	2017	Sohaib Shahid Bajwa and Xiaofeng Wang	Springer Science+Business Media New York 2016
18 100+ Metrics for Software Startups	2017	Kai-Kristian Kemell,Xiaofeng Wang, Anh Nguyen-Duc	University of Southeast Norway, Norway
18 A Collaborative Approach to Teaching Software Startups	2018	Rafael Chanin, Afonso Sales, Alan Santos, Leandro Pompermaier, Rafael Prikladnicki	Pontifical Catholic University of Rio Grande do Sul, School of Technology
18 Agile Business Model Innovation in Digital Entrepreneurship. Lean Startup approaches	2018	Antonio Ghezzi, Angelo Cavallo	Journal of Business Research
18 An anatomy of requirements engineering in software startups	2018	Nirnaya Tripathi, Eriks Klotins, Rafael Prikladnicki, Markku Oivoa	The Journal of Systems and Software
18 Antecedents and consequences of technology pivots in software startups	2018	Nicolai Bohn, Dennis Kundisch	Hasso-Plattner-Institute, School of Entrepreneurship, Potsdam, Germany

18 Application of the Lean Startup Methodology in Project Management at Launching	2018	Nataliia Veretennikova, Roman Vaskiv	Information Systems and Networks Department, Lviv Polytechnic National University,Lviv, Ukraine
18 Challenge Based Startup Learning. A Framework to Teach Software Startup	2018	Rafael Chanin, Leandro Pompermaier, Kellen Fraga, Afonso Sales and Rafael Prikladnicki	Pontifical Catholic University of Rio Grande do Sul, School of Technology, Porto Alegre, Brazil
18 Comparison of release engineering practices in a large mature company and a startup	2018	Eero Laukkanen, Maria Paasivaara, Juha Itkonen,Casper Lassenius	Department of Computer Science, Aalto, Finland
18 Continuous Innovation through experimentation	2018	Cherif Amirat, Richard Reeps	2018 IEEE Technology and Engineering Management Conference (TEMSCON)
18 Decision-making in software product management	2018	Andrey Saltan, Slinger Jansen and Kari Smolander	Lappeenranta University of Technology, Lappeenranta, Finland
18 Digital startups and the adoption and implementation of Lean Startup approaches	2018	Antonio Ghezzi	Technological Forecasting & Social Change
18 Exploration of Technical Debt in Start-ups	2018	Eriks Klotins, Michael Unterkalmsteiner, Panagiota Chatzipetrou	Pontifical Catholic University of Rio Grande do Sul Porto Alegre, Brazil
18 From MVPs to Pivots. A Hypothesis-Driven Journey of Two Software startups	2018	Dron Khanna, Anh Nguyen-Duc, and Xiaofeng Wang	University of Bozen-Bolzano, Bolzano, Italy
18 La metodología Lean Startup	2018	Francisco Javier Llamas Fernández, Juan Carlos Fernández Rodríguez	Universidad Antonio de Nebrija
18 Lean Internal Startups for Software Product Innovation in Large Companies	2018	Henry Edisona, Nina M. Smørsgård, Xiaofeng Wanga, Pekka Abrahamssonc	The Journal of Systems and Software
18 Measurements in the Early Stage Software Start-ups	2018	Grace Kamulegeya, Raymond Mugwanya, Regina Hebig	Chalmers Gothenburg University, Gothenburg, Sweden,
18 MVP Explained. A Systematic Mapping Study on the Definitions of Minimal Viable Product	2018	Valentina Lenarduzzi, Davide Taibi	Faculty of Computer Science Free University of Bolzano/Bozen, Bolzano/Bozen - Italy
18 Poster. Startup Software Development Education	2018	Rafael Chanin, Afonso Sales, Leandro Pompermaier, Rafael Prikladnicki	Pontifical Catholic University of Rio Grande do Sul, School of Technology
18 Software start-ups through an empirical lens	2018	Eriks Klotins, Michael Unterkalmsteiner, Panagiota Chatzipetrou	Blekinge Institute of Technology, Karlskrona, Sweden
18 Software Startup Education around the World	2018	Rafael Chanin, Dron Khanna	PUCRS, Porto Alegre, Brazil
18 Software startup engineering. A systematic mapping study	2018	Vebjørn Berga,Jørgen Birkeland	The Journal of Systems & Software
18 Software-Intensive Product Engineering in Start-Ups (Bis)	2018	Eriks Klotins, Michael Unterkalmsteiner, and Tony Gorschek,	Blekinge Institute of Technology, Karlskrona, Sweden
18 Survey. How much the academic startups know and use Agile	2018	Simona Ibba, Gavina Baralla, Roberto Tonelli	Department of Electric and Electronic Engineering, University of Cagliari, Cagliari, Italy
18 Teaching Lean Startup Principles	2018	Matthias Gutbrod and Jürgen Münch	Faculty of Informatics, Reutlingen University, Reutlingen, Germany
18 Tech Startup Learning Activities. A Formative Evaluation	2018	Kevin Buffardi	Dept. of Computer Science California State University, Chico
18 Technical Aspects of software development in startups	2018	Bruno Henrique Cavalcante, Gislaine Camila Lapasini Leal, Ivaldir de Farias Junior	Ciência da Computação, Universidade Estadual de Maringá, Maringá, Paraná

18 The Evolution of Requirements Practices in Software Startups	2018	Catarina Gralha, Daniela Damian	Universidade NOVA de Lisboa
18 The Role of Data Analytics in Startup companies	2018	Vebjørn Berg, Jørgen Birkeland , Ilias O. Pappas ,and Letizia Jaccheri	Department of Computer Science, Norwegian University of Science and Technology, rondheim, Norway
18 Towards Prioritizing Software Business Requirements in Startups	2018	Saleh Albuga, Yousra Odeh	2018 8th International Conference on Computer Science and Information Technology (CSIT)
18 What influences software startups to use Lean Startup	2018	Jorge Melegati	University of Bozen-Bolzano, Bolzano, Italy
19 A Case Study about Startups' Software Development Practices	2019	Renata Souza, Karla Malta, Eduardo Santana de Almeida	Federal University of Bahia (UFBA), Salvador, Bahia, Brazil
19 A model of requirements engineering in software startups	2019	Jorge Melegati, Alfredo Goldmanb, Fabio Konb, Xiaofeng Wanga	Information and Software Technology
19 A progression model of software engineering goals, challenges, and practices in start-ups	2019	Eriks Klotins, Michael Unterkalmsteiner	JOURNAL OF LATEX CLASS FILES, VOL. 13, NO. 99
19 Boosting Agile by Using User-Centered Design and Lean Startup	2019	Ingrid Signoretti, Sabrina Marczak, Larissa Salerno, Augusto de Lara	School of Technology, Porto Alegre, Brazil
19 Characterizing industry-academia collaborations in software engineering	2019	Vahid Garousi, Dietmar Pfahl, João M. Fernandes, Michael Felderer	Empirical Software Engineering
19 Collaborative Practices for Software Requirements Gathering in Software Startups	2019	Rafael Chanin, Leandro Pompermaier, Afonso Sales and Rafael Prikladnicki	2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)
19 Comparing effectuation to discovery-driven planning, prescriptive entrepreneurship	2019	Yashar Mansoori & Martin Lackéus	Small Bus Econ
19 Customer Feedback Prioritization Technique	2019	Syeda Sumbul Hossain, S. A. M. Jubayer, Shadikur Rahman	Daffodil International University, Dhaka, Bangladesh
19 DEEP. The Product Roadmap Maturity Model	2019	Jürgen Münch, Stefan Trieflinger, Dominic Lang	Department of Computer Science, Reutlingen University, Reutlingen, Germany
19 Digital startups and the adoption and implementation of Lean Startup Approaches	2019	Antonio Ghezzi	Technological Forecasting & Social Change
19 Enablers and Inhibitors of experimentation in early-stage software startups	2019	Jorge Melegati, Rafael Chanin, Xiaofeng Wang, Afonso Sales,and Rafael Prikladnicki	Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
19 Evolving with Patterns. A 31-Month Startup Experience Report	2019	Miguel Ehécatl Morales-Trujillo, Gabriel Alberto García-Mireles	Computer Science and Software Engineering Department, University of Canterbury, Christchurch, New Zealand
19 Improving a Startup Learning Framework Through an Expert Panel	2019	Rafael Chanin, Afonso Soles, Leandro Pompermaier, and Rafael Prikladnicki	School of Technology, PUCRS, Porto Alegre, Brazil
19 Insights into startup ecosystems through exploration of multi-vocal literature	2019	Nirnaya Tripathi, Pertti Seppänen, Ganesh Boominathan, Markku Oivo, Kari Liukkunen	Group, University of Oulu, Oulu, Finland
19 Investigating Agile Practices in Software Startups	2019	Renata Souza, Karla Malta, Eduardo Santana de Almeida	Federal Institute of Bahia (IFBA) Santo Antônio de Jesus, BA, Brazil

19 Lean startup and the business model. Experimentation revisited	2019	Teppo Felina, Alfonso Gambardellab, Scott Sternc,Todd Zengerd	Bocconi University, Department of Management & Technology and ICRIOS, Milan, Italy
19 MVP Development Process for Software Startups	2019	Leandro Pompermaier, Rafael Chanin, Afonso Sales, and Rafael Prikladnicki	School of Technology, PUCRS, Porto Alegre, Brazil
19 Perceived Benefits and Challenges of Learning Startup methodologies	2019	Jorge Melegati, Rafael Chanin, Xiaofeng Wang, Afonso Sales,and Rafael Prikladnicki	University of Bozen-Bolzano, Bolzano, Italy
19 Requirements for Measurement Dashboards and their Benefits. A Study of Start-ups	2019	Grace Kamulegeya, Raymond Mugwanya, Regina Hebig	2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)
19 Startup ecosystem effect on minimum viable product development in software startups	2019	Nirnaya Tripathi, Markku Oivo, Kari Liukkunen, Jouni Markkula	University of Oulu, Oulu 90014, Finland
19 Technical Debt Trade-Off. Experiences from Software Startups	2019	Orges Cico	Norwegian University of Science and Technology, Trondheim, Norway
19 The drivers of success in new-product development	2019	Robert G. Coopera	University's Smeal College of Business Administration, USA
20 Combining User-Centered Design and lean startups	2020	Ingrid Signoretti, Larissa Salerno, Sabrina Marczak, and Ricardo Bastos	Group, School of Technology, PUCRS, Porto Alegre, RS, Brazil
20 Continuous Information Monitoring in software startups	2020	Usman Rafiq and Xiaofeng Wang	Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
20 InnoDev Workshop. A One Day Introduction to Combining Design Thinking, Lean Startup	2020	Franziska Dobrigkeit, Danielly de Paula, Noel Carroll	32nd IEEE Intl. Conference on Software Engineering Education & Training
20 Lean Startup, Agile Methodologies and Customer Development	2020	Diego Souza Silva, Antonio Ghezzi, Rafael Barbosa de Aguiar, Marcelo Nogueira, Cortimiglia, Carla Schwengber ten Caten	Industrial Engineering Department, Universidade Federal do Rio Grande do Sul. Porto, Alegre, Brazil
20 Learning Outcomes Development for Informatics Engineering Students by Using Lean-Startup Model	2020	M. A. Gumilang, B. Etikasari, E. Antika, T. D. Puspitasari, A. H. Utomo	Department of Electrical Engineering, Universitas Negeri Malang, Malang, Indonesia
20 MVP and experimentation in software startups. A qualitative survey	2020	Jorge Melegati, Rafael Chanin, Xiaofeng Wang, Afonso Sales,and Rafael Prikladnicki	2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)
20 On the Pragmatics of Requirements Engineering Practices in a Startup Ecosystem	2020	Carina Alves, João Cunha, João Araújo	2020 IEEE 28th International Requirements Engineering Conference (RE)
20 Practices and Tools for Software Start-Ups	2020	Gislaine Camila Lapasini Leal, Rafael Prikladnicki, Christof Ebert, Renato Balancieri, and Leandro Bento Pompermaier	PUBLISHED BY THE IEEE COMPUTER SOCIETY
20 Reaching the Unreachable. A Method for Early Stage Software Startups to Reach Inaccessible Stakeholders	2020	Soo Ling Lim, Peter J. Bentley, Fuyuki Ishikawa	2020 IEEE 28th International Requirements Engineering Conference (RE)
20 Requirements Engineering in Software Startups. A systematic mapping	2020	Varun Gupta, Jose Maria Fernandez-Crehuet, Thomas Hanne and Rainer Telesko	Universidad Politécnica de Madrid, 28006 Madrid, Spain; josemaria.fernandez-crehuet@upm.es
20 Software Startup Practices (Essence Theory of Software Engineering)	2020	Kai-Kristian Kemell, Ville Ravaska, Anh Nguyen-Duc, and Pekka Abrahamsson	University of Southeast Norway, Notodden, Norway

20 Software Startups in growth phase SE practices adopted to SEE	2020	Orges Cico	2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)
20 Startups transitioning from early to growth phase	2020	Orges Cico, Renata Maria de Souza, Letizia Jaccheri	Norwegian University of Science and Technology, Trondheim, Norway
20 Taming and Unveiling Software Reuse opportunities	2020	Renata Souza, Karla Malta, Eduardo Santana de Almeida	Federal University of Bahia (UFBA), Salvador, Brazil
20 The Character of Software Startup Hubs in an Emerging Ecosystem	2020	Grace Kamulegeya, Raymond Mugwanya, Regina Hebig	2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)
20 The effect of firm complexity and founding team size on agile internal	2020	Tomás F. González-Cruz, Dolores Botella-Carrubi, C. M. Martínez-Fuentes	International Entrepreneurship and Management Journal (2020)
20 The Perception and Management of technical debt in software startups	2020	Cecilia Apa, Helvio Jeronimo, Luciana M. Nascimento, Diego Vallespir, and Guilherme Horta Travassos	Springer Nature Switzerland AG 2020
20 Towards Specific Software Engineering practices for Early-Stage Startups	2020	Jorge Melegati, Rafael Chanin, Afonso Sales,and Rafael Prikladnicki	Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
20 Towards Transferring Lean Software Startup Practices in software engineering education	2020	Orges Cico	Norwegian University of Science and Technology, Trondheim, Norway
20 Understanding Hypotheses Engineering in Software Startups through a Gray Literature Review	2020	Jorge Melegati , Eduardo Guerra , Xiaofeng Wang	University of Bozen-Bolzano, Bolzano, Italy
20 XPro. A Model to Explain the Limited Adoption and implementation	2020	Jorge Melegati, Henry Edison, and Xiaofeng Wang	University of Bozen-Bolzano, Bolzano, Italy
21 A Progression Model of Software Engineering Goals, Challenges, and Practices in Start-ups	2021	Eriks Klotins, Michael Unterkalmsteiner, Panagiota Chatzipetrou	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 47, NO. 3, MARCH 2021
21 A Systematic Mapping Study on the Use of... to develop MVPs	2021	Silvio Alonso, Marcos Kalinowski, Marx Viana, Bruna Ferreira, Simone D.J. Barbosa	Pontifical Catholic University of Rio de Janeiro (PUC-Rio) Rio de Janeiro, Brazil
21 A Trend Analysis of Software Business Research	2021	Sami Hyrynsalmi and Arho Suominen	Department of Software Engineering, LUT University, Lahti, Finland
21 Agile Development Practices Applied to Software Startups	2021	Paulo Cesar Abrantes, Ana Paula Furtado	2021 16th Iberian Conference on Information Systems and Technologies (CISTI)
21 Analytics Mistakes that Derail Software Startups	2021	Usman Rafiq and Xiaofeng Wang	Faculty of Computer Science, University of Bozen-Bolzano, Bolzano, Italy
21 Business Model Flexibility and Software-intensive companies	2021	Magnus Wilson, Krzysztof Wnuk, Lars Bengtsson	e-Informatica Software Engineering Journal, Volume 15, 2021,
21 Dual-Track Agile in Early-Stage Startups	2021	Elena Lape	ICSEA 2021 : The Sixteenth International Conference on Software Engineering Advances
21 HyMap. Eliciting hypotheses in early-stage software startups using cognitive maps	2021	Jorge Melegati, Eduardo Guerra, Xiaofeng Wang	University of Bozen-Bolzano, Bolzano, Italy

21 Innovation Through Startup Collaboration	2021	Varun Gupta, Luis Rubalcaba	IEEE ENGINEERING MANAGEMENT REVIEW, VOL. 49, NO. 3, THIRD QUARTER, SEPTEMBER 2021
21 Making Internal Software Startups Work	2021	Anastasiia Tkalich, Nils Brede Moe, and Rasmus Ulfsnes	SINTEF Digital, 7034 Trondheim, Norway
21 Minimum Viable Product Creation and Validation in Software Startups	2021	Gerardo Matturro, Gonzalo Nieto, Alfonso González, Martín Solari	Facultad de Ingeniería, Universidad ORT Uruguay
21 Online Feedback Management Tools for Early-Stage Startups	2021	Varun Gupta, Luis Rubalcaba	Universidad Politecnica de Madrid
21 Prototyping Practices in Software Startups	2021	Elizabeth Bjarnason	2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)
21 Quality Assurance Guidelines for Successful Startups	2021	Shatadru Shikta, H. M. Mahir Shahriyar, Sowvik Kanti Das	Software Engineering, Independent University ,Dhaka, Bangladesh
21 Risk Exposure and Management in software startups	2021	Gholamhossein Kazemi, Orges Cico, Quang-Trung Nguyen	Springer Nature Switzerland AG 2021
21 Startups Transitioning from Early to Growth Phase	2021	Orges Cico, Renata Souza, Letizia Jaccheri, Anh Nguyen Duc, and Ivan Machado2	Norwegian University of Science and Technology, Trondheim, Norway
21 The entrepreneurial logic of startup software development	2021	Anh Nguyen-Duc, Kai-Kristian Kemell, Pekka Abrahamsson2	Empirical Software Engineering (2021) 26: 91
21 The lean startup framework. Closing the academic-practitioner divide	2021	Dean A. Shepherd and Marc Gruber	University of Notre Dame, IN, USA
21 Toward a Technical Debt Relationship with the Pivoting of Growth	2021	Orges Cico, Terese Besker	Norwegian University of Science and Technology, Trondheim, Norway
21 Understanding Barriers to Internal Startups in Large Organizations	2021	Tor Sporsem, Anastasiia Tkalich, Nils Brede Moe, Marius Mikalsen	2021 IEEE/ACM Joint 15th International Conference on Software and System processes
21 Understanding Hypotheses Engineering in Software Startups	2021	Jorge Melegati , Eduardo Guerra , Xiaofeng Wang	University of Bozen-Bolzano, Piazza Domenicani 3, Bolzano, Italy
21 Use of Agile Practices in Start-up Companies	2021	Eriks Klotins, Michael Unterkalmsteiner, Panagiota Chatzipetrou∗∗,	e-Informatica Software Engineering Journal, Volume 15,, 2021,
21 User Experience Practices in Early-Stage Software Startups	2021	Guilherme Corredato Guerino, Nayra Suellen Borges Cruz Dias,	State University of Maring´a, Maring´a, Brazil
21 UX work in software startups. A Thematic Analysis of the Literature	2021	Jullia Saad, Suéllen Martinell,Leticia S. Machado	University of São Carlos, Rdv. João L. Santos, Sorocaba, SP, Brazil
22 Development frameworks for software startups	2022	Narendranath Shanbhag and Eric Pardede	La Trobe University, Australia
22 Assessing the Role of Minimum Viable Products in Digital Startups	2022	J. Umbreen, M.Z. Mirza,Y. Ahmad, A. Naseem	2022 IEEE International Conference on (IEEM)
22 Hacking or Engineering. Towards an Extended Entrepreneurial software engineering model	2022	Marco Kuhrmann, Jürgen Münch, Jil Klünder	Reutlingen University, Germany
22 Improving Agile Software Development using User-	2022	Maximilian Zorzetti, Ingrid Signoretti, Larissa Salerno, Sabrina Marczak, Ricardo Bastos	School of Technology, Porto Alegre, RS, Brazil

Centered Design and lean startup			
22 Influences of UX factors in the Agile UX context of software startups	2022	Joelma Choma, Eduardo M. Guerra, Alexandre Alvaro, Roberto Pereira, Luciana Zaina	Federal University of Parana, Curitiba, Brasil
22 Long-Term UX framework. Supporting software startups in UX research	2022	Suéllen Martinelli, Luciana Zaina	Federal University of São Carlos (UFSCar), Brazil
22 The Viability of Continuous Experimentation in Early-Stage Software Startups	2022	Vihtori Mäntylä, Bettina Lehtelä and Fabian Fagerholm	Aalto University, Espoo, Finland
22 Towards understanding analytics in software startups	2022	Usman Rafiq	2022 IEEE/ACM International Workshop on Software-Intensive Business (IWSiB)
22 Towards Understanding How Software startups deal with UX	2022	Joelma Choma, Eduardo M. Guerra, Alexandre Alvaro, Roberto Pereira, Luciana Zaina	Federal University of Parana, Curitiba, Brasil
22 User Experience Practices in Software Startups	2022	Guilherme Corredato Guerino, Marcos de Assumpção,Thiago Jose da Silva	Departament of informatics, state University of maringa, Brasil
22 Using Lean Personas to the Description of UX-relatted requirements	2022	Gabriel V. Teixeira and Luciana A. M. Zaina	24th International Conference on Enterprise Information Systems (ICEIS 2022)
22 XPro. A Model to Explain the Limited Adoption. .. of Experimentation in Software Startups	2022	Jorge Melegati , Henry Edison , and Xiaofeng Wang	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 48, NO. 6, JUNE 2022
22 A Systematic Mapping Study and Practitioner Insights on the	2022	Silvio Alonso, Marcos Kalinowski, Marx Viana, Bruna Ferreira, Simone D.J. Barbosa	Department of Informatics - Pontifical Catholic University of Rio de Janeiro Rio de Janeiro, Brazil
23 Does maturity level influence the use of Agile UX methods by digital startups	2023	Varun Gupta	Multidisciplinary Research Centre for Innovations in SMEs, GISMA University of Applied Sciences, 14469, Potsdam, Germany