UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERIA

Proceso Ágil para un curso de grado

Desarrollo, documentación y estudio de caso

Angie Stephanie Lecot Emaldi

Tesis de Maestría presentada al Programa de Posgrado en Ingeniería de software, Facultad de ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magíster en Ingeniería de software.

Director:
Diego Vallespir

Tribunal:
Dra. Laura González (revisora)
Dra. Juliana Herbert
Dra. Adriana Marotta

Montevideo – Uruguay
Noviembre de 2022

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Proceso Ágil para un curso de grado

Desarrollo, documentación y estudio de caso

Angie Stephanie Lecot Emaldi

Programa de Posgrado en Ingeniería de software
Facultad de ingeniería
Universidad de la República

Montevideo – Uruguay
Noviembre de 2022

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Proceso Ágil para un curso de grado

Desarrollo, documentación y estudio de caso

Angie Stephanie Lecot Emaldi

Tesis de Maestría presentada al Programa de Posgrado en Ingeniería de software, Facultad de ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magíster en Ingeniería de software.

Director:
Diego Vallespir

Director académico:
Diego Vallespir

Montevideo – Uruguay
Noviembre de 2022

Lecot Emaldi, Angie Stephanie

Proceso Ágil para un curso de grado / Angie Stephanie Lecot Emaldi. - Montevideo: Universidad de la República, Facultad de ingeniería, 2022.

XIII, 141 p. 29, 7cm.

Director:

Diego Vallespir

Director académico:

Diego Vallespir

Tesis de Maestría – Universidad de la República, Programa en Ingeniería de software, 2022.

Referencias bibliográficas: p. 128 – 131.

1. curso capstone basado en proyectos, 2. proyecto real, 3. ingeniería de software, 4. procesos de desarrollo, 5. ágil. I. Vallespir, Diego, . II. Universidad de la República, Programa de Posgrado en Ingeniería de software. III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Dra. Laura González (revisora)

Dra. Juliana Herbert

Dra. Adriana Marotta

Montevideo – Uruguay
Noviembre de 2022

Para vos, abuelo; mi familia y
amigos...

Agradecimientos

Cuando alguien escribe la tesis, planifica escribir los agradecimientos al final, piensa «lo dejo para lo último» porque ya tengo todo el resto y es lo más simple de hacer... Sin embargo, mi propia experiencia dice que llegado el momento de escribirla no es tan fácil. Pasan por la mente muchas cosas: ¿Los escribo formalmente o a «mi estilo»? ¿Qué pasa si me olvido de alguien importante?, ¿o no agradezco lo suficiente?

Antes de continuar, cualquier olvido u omisión fue involuntario y espero que se vean reflejados todos aquellos que colaboraron, ya sea por sus aportes en la construcción de forma directa o siendo parte del apoyo intangible de esta tesis.

Si bien se tiende a pensar que una tesis es una construcción individual —o de un pequeño colectivo—, en esta participaron muchos, aportando grandes granitos de arena para poder lograrla y, así, contar con un resultado del que me siento sumamente satisfecha y orgullosa, tanto por haberlo logrado como de su contenido, luego de tanto esfuerzo.

A Diego, que, sea cual sea el significado de la palabra tutor, él fue más allá y ha sido un acompañante fundamental, moviendo los hilos durante la tesis para lograr este resultado, exigiendo lo que él sabe que puedo dar y aflojando en los momentos que fue necesario. Todo esto sin dejar de lado la construcción conjunta a la par de esta tesis y el logro de nuestros objetivos.

A todos los que colaboraron con sus aportes o escuchando en diferentes instancias de construcción, revisión y validación del proceso, desde sus distintas visiones o en intercambios totalmente informales: Cholo, Cecilia, Vane, Leti, Vale, Arianne, Santi, Cristina, Jorge y Maile.

A Vane y Vale, por confiar en la propuesta cuando aún se estaba construyendo y enriquecerla a través de su experiencia. A los estudiantes y clientes que colaboraron para que la aplicación y evaluación del proceso fueran posibles.

A Sil y Ceci, por animarse a aplicarlo sin tanto apoyo.

A Seba, por sus aportes para inspirar en los momentos justos. Saldamos la deuda.

Al Grupo de Ingeniería de Software (Gris) de la Facultad de Ingeniería, por su acompañamiento y colaboración de una u otra forma en este camino.

Al revisor y al resto de tribunal, por haber aceptado ser parte de este trabajo y sus aportes.

Al equipo de tecnología de Agesic por el aguante, a los que tomaron la posta y especialmente a Elena por darme los espacios cuando los necesité.

A mi mamá, la de los mensajes diarios de Mickey, por ser la madre que más entiende todos los temas tecnológicos.

A mi papá, por sus escuetos mensajes para no molestar y por haberme dado el mejor mensaje al ingresar al liceo —y que caló hondo—: «Si no estudiás, te... ¡Bip!».

A la enana, por siempre estar orgullosa y entender que los bichos raros somos así y seguimos estudiando infinitamente.

A mi abuela, que aunque no entiende qué es lo que estudio y odia los celulares, siempre se preocupa por la cantidad de horas de estudio y trabajo.

A mi abuelo julio y mi madrina, que estén donde estén siempre me han acompañado.

A los gurises, por estar siempre al firme desde hace más de la mitad de una vida.

A mis sobrinos y ahijados, todos ellos, aunque algunos no lo van a entender ahora, gracias por ser comprensivos y perdón por las ausencias.

A Cecikia, que, aunque sus correcciones de Whatsapp desesperan, me regaló uno de los mejores libros y su apoyo es esencial para exprimir el idioma español.

Al resto de la familia (Lecot, Emaldi, Dudok, Parrilla, etc) y amigos, que han sido pacientes en aceptar varios «no» por respuesta a encuentros y cumpleaños.

A Santi, no sabría ni cómo escribir en un libro entero todo lo que tengo para agradecerte. Entonces, gracias por TODO, ¡se viene Gloomhaven!

One principle problem of educating software engineers is that they will not use a new method until they believe it works and, more importantly, that they will not believe the method will work until they see it for themselves.

Watts Humphrey

RESUMEN

El software cada vez está más inmerso en la sociedad. Su uso intensivo hace que no concibamos la vida sin él. Muchas responsabilidades y toma de decisiones a nivel individual y como sociedad involucran algún tipo de software desde uno de propósito general (e.g., editor de texto) a más críticos y complejos como los diseñados para tomar decisiones de forma autónoma (e.g., autos autónomos). Las necesidades cambiantes, la intensificación del uso de las tecnologías, y su criticidad, presentan dos grandes desafíos: (i) incorporar procesos de desarrollo ágiles que permitan crear software eficiente, oportuno y de calidad; y (ii) desarrollar en los profesionales de software las habilidades y capacidades requeridas para el desarrollo de software contemporáneo.

Las universidades abordan el desafío de formar profesionales que cuenten con las competencias necesarias para la construcción de software. Los cursos *capstone* basados en proyectos, donde grupos de estudiantes siguiendo y adaptando un proceso de desarrollo de software llevan adelante un proyecto de construcción de un producto, buscan que los estudiantes apliquen su conocimiento previo en un problema real que sea lo más similar a la industria para lograr que desarrollen distintas habilidades necesarias para la producción de software.

El "Proyecto de Ingeniería de Software" (PIS) es un curso *capstone* basado en proyectos de desarrollo de software del que participan estudiantes de cuarto año de la carrera Ingeniería en Computación de la Facultad de Ingeniería de la Universidad de la República de Uruguay. El proceso que se utiliza actualmente en el PIS es iterativo e incremental y dirigido (fuertemente) por planes; y pasaron diez años desde su última actualización.

Esta tesis propone para el PIS un proceso de desarrollo ágil, llamado PISAgile, que guía el aprendizaje de los estudiantes y la construcción de software de calidad. Promueve la colaboración, experimentación de forma temprana y la reflexión para agregar valor, evolucionar y adaptarse a las necesidades cambiantes. PISAgile consta de tres fases: fase inicial, fase de construcción y fase de transición. En la fase inicial los estudiantes tienen un primer acercamiento al proyecto y proceso, se busca que entiendan las necesidades del cliente y lo-

gren conceptualizar el producto a construir. La fase de construcción tiene por objetivo la construcción y aceptación del producto de software. En la fase de transición se hace la transferencia final de los productos al cliente.

PISAgile se aplicó a tres grupos de estudiantes en 2020 y para su evaluación se diseñaron, ejecutaron y analizaron encuestas a los distintos actores: estudiantes, docentes y clientes. En particular en esta tesis, por aspectos vinculados al alcance del trabajo, se presenta únicamente el análisis de la percepción de los estudiantes sobre distintos aspectos a evaluar. Principalmente se evaluaron las habilidades blandas.

Los resultados indican que contar con un proceso ágil disciplinado y un curso como el PIS logra preparar a los estudiantes para su práctica profesional desarrollando tanto habilidades técnicas como blandas. Además, PISAgile sirvió para que los estudiantes pudieran construir software de calidad. Los estudiantes perciben que las fortalezas de PISAgile son contar con fases, iteraciones y roles e incorporar la reflexión y evaluación para la mejora continua. También entienden que el aspecto a mejorar es la forma en la que actualmente está documentado PISAgile.

Entendemos que es un importante aporte para el curso PIS contar con un proceso ágil especificado y actualizado que pueda evolucionar por cambios en el curso. Para la academia entendemos que es un aporte contar con un proceso concebido y evaluado en un contexto académico, que se puedan aplicar, y generar mayor conocimiento en la enseñanza de procesos y el aprendizaje de los estudiantes.

Las principales líneas del trabajo futuro se orientan a realizar los ajustes que surgen de la evaluación, particularmente mejorar la documentación de PISAgile, implantarlo para todos los grupos de estudiantes del PIS y complementar la evaluación realizada con la percepción de docentes y clientes.

Palabras claves:
curso capstone basado en proyectos, proyecto real, ingeniería de software, procesos de desarrollo, ágil.

Tabla de contenidos

Capítulo 1

Introducción

La masificación en el uso de software ha hecho que esta industria sea una de las de mayor desarrollo en los últimos años, requiriendo así contar con profesionales preparados y especializados en la construcción y evolución de software. A su vez, la fuerte competencia y las exigencias de los clientes determinan la importancia de contar con software de calidad que cumpla con las expectativas de estos y que el tiempo de comercialización (del inglés, *time to market*) sea el más corto posible.

Este es un gran desafío para las universidades como responsables de la formación de profesionales con conocimientos y habilidades que le permitan ingresar a la industria de software y desarrollarse para cumplir con los requisitos y estándares actuales y futuros (Garousi et al. 2020; Tuzun et al. 2018). Este desafío se puede abordar mediante la enseñanza y entrenamiento en ingeniería de software.

La Ingeniería de software (IS) es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, mantenimiento y operación del software (IEEE, 1990). El desarrollo de métodos, técnicas, principios y prácticas de esta disciplina son importantes para el desarrollo de sistemas grandes, complejos y confiables (Joint Task Force on Computing Curricula 2005, 2005).

Un proceso de software es un conjunto de actividades y tareas que se encuentran interrelacionadas, que transforman los productos de trabajo de entrada en productos de trabajo de salida para desarrollar software (Bourque y Fairley, 2014). Los modelos de procesos de software son una representación simplificada y abstracta de un proceso, son una descripción de un proceso de software desde una perspectiva en particular (Sommerville, 2016). Se utilizan

como descripciones de la forma en la que debe realizarse el desarrollo de software, por lo tanto son importantes para comunicarlo a todos los interesados (del inglés, *stakeholders*). Fundamentalmente, sirven para la coordinación de las tareas del equipo de desarrollo de software, el entendimiento del rol de cada integrante y la mejora del propio proceso (Cha et al. 2019).

En la carrera Ingeniería en Computación de la Facultad de Ingeniería de la Universidad de la República de Uruguay, se dicta "Proyecto de Ingeniería de Software" (PIS), un curso *capstone* basado en proyectos, destinado a los estudiantes de grado de la carrera. Como es tomado durante el segundo semestre del cuarto año de estudios, los estudiantes ya tienen aprobados cursos en diferentes áreas temáticas de la computación: programación, arquitectura de computadores, sistemas operativos, redes, sistemas de información, ingeniería de software, entre otras.

El PIS consiste en realizar un proyecto de software, que es llevado a cabo por los estudiantes aplicando un proceso de desarrollo de software, en grupos de entre diez a quince integrantes. A cada grupo se le asigna un proyecto presentado por un cliente real (i.e. cliente que presenta un proyecto análogo a los de la industria de desarrollo de software, independientemente del ámbito al que pertenezca —industria, academia o estado—). El objetivo del curso es proporcionar a los estudiantes una experiencia similar a la que pueden experimentar en la industria de software. Se busca que incorporen, afirmen y profundicen los conocimientos de ingeniería de software mediante su aplicación práctica (InCo, 2017).

Desde el año 2002, el PIS utiliza modelos de proceso iterativos e incrementales, y dirigidos (fuertemente) por planes. Existe un modelo base, el "Modularizado Unificado y Medible" (MUM) (Pedrana y Bellini, 2005), y dos extensiones: una enfocada a lenguajes orientados a objetos y otra al ambiente de desarrollo GeneXus[1]. El MUM está documentado con la herramienta Eclipse Process Framework (EPF) desde el año 2009. Además, está disponible la memoria organizacional de los proyectos ejecutados en el marco de la asignatura para que los estudiantes cuenten con evidencia que les ayude a aplicar y mejorar el proceso.

Los procesos ágiles son procesos de desarrollo iterativos que se focalizan en

[1]GeneXus es un ambiente de desarrollo de sistemas de información que utiliza un lenguaje de cuarta generación; más conocido hoy como lenguajes de poco código *(low code)*. https://www.genexus.com/

reducir la sobrecarga de los procesos, en disminuir la documentación a generar y en realizar una entrega de software incremental (Sommerville, 2016). Dos de los procesos ágiles más utilizados en la industria son Scrum (Schwaber y Sutherland, 2020) y eXtreme Programming (Beck y Andres, 2004).

Al no contar en el PIS con una descripción de un proceso ágil y dada la necesidad planteada por los clientes de aplicar uno, en algunos casos los mismos clientes del curso han brindado la descripción del proceso a seguir y en otros los estudiantes, con la guía del tutor (docente del curso), lo van definiendo a medida que avanza el proyecto. Al no existir una definición uniforme entre los tutores, el proceso termina dependiendo directamente del grupo y su tutor, perdiendo los beneficios que proporcionan los procesos cuando están especificados.

Un proyecto de desarrollo de software final (del inglés, *Capstone project*), realizado por un grupo de estudiantes, les permite sintetizar el conocimiento adquirido en varios cursos y aplicarlo a un problema real. Al aplicarlos a la enseñanza de la ingeniería de software, mejora las habilidades de comunicación y técnica de los estudiantes (Tafliovich et al. 2019). Para lograr que los proyectos sean lo más realistas posible, es importante la colaboración con clientes de la industria y que estos tengan una colaboración activa y suficiente tiempo para dedicar al equipo del proyecto (Paasivaara et al. 2019). De esta forma, se involucra a la industria generando sinergia entre esta y la academia en la formación de los estudiantes para el ámbito laboral.

En este trabajo presentamos el proceso de desarrollo de software PISAgile, creado en el marco de esta tesis, para ser aplicado en el contexto del curso de grado PIS.

PISAgile tiene por objetivo guiar en la construcción de productos de software de calidad acorde a las necesidades del cliente y en los plazos definidos, y ser un facilitador para que los estudiantes —por medio de su aplicación— puedan adquirir y desarrollar ciertos conocimientos y habilidades. Es un proceso para el desarrollo de software que realiza un abordaje que va desde la identificación de la necesidad del cliente hasta la transferencia de los productos construidos y los resultados obtenidos. Contempla las principales disciplinas de la ingeniería de software (Bourque y Fairley, 2014) y el desarrollo de las habilidades técnicas y blandas necesarias para llevar a cabo un proyecto de desarrollo de software. Para las habilidades blandas se tuvieron en cuenta las propuestas en otro curso de tipo *capstone project*, que evalúa el efecto que tiene este diseño de curso

en la adquisición de las habilidades blandas necesarias para trabajar en la industria del software por parte de estudiantes en un proyecto real (Khakurel y Porras, 2020).

También, en el marco de esta tesis, realizamos una evaluación del proceso PISAgile, aplicándolo directamente en los proyectos de tres grupos de estudiantes en el curso 2020. Para evaluarlo realizamos encuestas individuales y anónimas, que fueron respondidas por los distintos actores que intervinieron en el curso, y a su vez, analizamos los resultados de algunas de las encuestas.

1.1. Motivación

Desde hace algunos años, la industria de software a nivel mundial y, particularmente la nacional, ha tenido cambios importantes en su crecimiento, cambios en sus métodos y forma de trabajo. Además, desde nuestra visión, en Uruguay en los últimos años ha habido un auge tanto en el surgimiento de *startups* como del uso de los métodos ágiles.

Estos cambios no han resultado ajenos al curso PIS. Se han presentado al curso cada vez más clientes cuyas organizaciones trabajan de forma exclusiva con métodos ágiles y/o solicitan que durante el proyecto los estudiantes los apliquen. Este desafío lo enfrentaron diversas universidades adoptando métodos ágiles para sus cursos (Hsu et al. 2019). En este contexto, emprendimos también este desafío y buscamos enseñar métodos ágiles en el PIS.

Hasta la fecha, el curso PIS tiene disponible, de carácter formal y estable, un único modelo de proceso de desarrollo de software, el "Modularizado Unificado y Medible" (MUM). Dicho modelo, dirigido por planes, iterativo e incremental, se creó en el año 2005 y fue actualizado por última vez en 2012. Desde que se creó, era requisito para la presentación de proyectos por parte de clientes, que aceptaran que los estudiantes iban a estar aplicándolo. Desde hace unos años este requisito se tornó más flexible y los clientes pueden proponer qué modelo de proceso o método a seguir durante el proyecto. En muchas ocasiones proponen un método ágil con el que no se cuenta actualmente.

Dado que uno de los objetivos del curso es que los estudiantes aprendan a seguir un modelo de proceso de desarrollo de software y sean capaces de ajustarlo a sus necesidades, es importante que tengan uno descrito. Por lo tanto, como ya mencionamos, se han aplicado distintas alternativas para cubrir este vacío. Esto tiene como ventaja que el propio equipo puede definir el proceso

que desea seguir, pero a su vez la desventaja de que los estudiantes se están formando y no tienen mucha experiencia en la definición de modelos de procesos de desarrollo y los objetivos de aprendizaje que se buscan con su aplicación. En el caso de que los estudiantes sigan un modelo de proceso descrito por el cliente está el inconveniente de que no siempre es trasladable de forma directa al contexto del curso. Otro problema que impacta directamente en los objetivos de aprendizaje es la percepción de los estudiantes acerca de los procesos y de los métodos ágiles. Al no tener el modelo de proceso descrito, hemos observado que asumen que no deben seguir un proceso o que los métodos ágiles no tienen un proceso a seguir, lo que deriva en un conocimiento erróneo.

El curso tiene dentro de sus objetivos de aprendizaje preparar a los estudiantes para su práctica profesional, teniendo como experiencia un proyecto en un ámbito que sea lo más similar posible a la industria del software, pero controlado en un ambiente académico. Actualmente, varias organizaciones aplican métodos ágiles o los prefieren ante los dirigidos por planes, y era un aspecto no abordado formalmente en el curso. Esto, sumado al tiempo que ha pasado desde la última revisión del MUM —teniendo en cuenta que los procesos deben actualizarse para mejorarlos—, es importante incorporar un nuevo proceso basado en métodos ágiles que permita cumplir con los objetivos de aprendizaje del curso y a su vez construir software de calidad.

En el programa del curso, cuya última actualización corresponde al 2017, se tiene en cuenta fomentar el trabajo en equipo, la responsabilidad, el compromiso individual y colectivo, la evaluación de pares y la autoevaluación individual y grupal (InCo, 2017). Sin embargo, no es explícito en la necesidad del desarrollo de determinadas habilidades blandas por parte de los estudiantes que son necesarias para la industria del software actual y específicamente para trabajar en equipos que promuevan un trabajo con métodos ágiles. Investigaciones en la industria de software indican que las organizaciones valoran mucho las habilidades blandas cuando reclutan ingenieros de software (ver, por ejemplo, (Khakurel y Porras, 2020)). A su vez, varios reportes indican que los profesionales no están bien preparados para la industria de software (ver, por ejemplo, (Bastarrica et al. 2017)).

1.2. Objetivos

El objetivo principal de nuestro trabajo es *crear un proceso de desarrollo de software ágil para ser aplicado en el PIS que haga foco en el aprendizaje de los estudiantes y que permita construir software de calidad acorde a las expectativas de los clientes.* Por medio de su aplicación, los estudiantes deben poder incorporar y poner en práctica sus conocimientos en ingeniería de software, y aplicar y desarrollar las habilidades (técnicas y blandas) necesarias para trabajar como profesionales en la industria del software (InCo, 2017).

Como segundo objetivo nos planteamos *evaluar el proceso creado* desde la mirada de los tres actores que participan en el proceso (estudiantes, docentes y clientes). Considerando su percepción sobre distintos aspectos a evaluar, principalmente en las habilidades blandas. En caso de ser necesario, se harán ajustes que surjan de su aplicación y evaluación.

Buscamos que el proceso creado se pueda implantar en el PIS para que los docentes y estudiantes comiencen a utilizarlo. Esto es para contar con un proceso ágil descrito que pueda evolucionar por cambios en el curso, en necesidades formativas de los estudiantes y mejoras en el proceso. Un estudio similar al nuestro fue llevado a cabo en un curso *capstone* para estudiantes de ciencias de la computación de la Universidad de Chile (Bastarrica et al. 2017).

A partir de estos dos objetivos de alto nivel, nuestros objetivos detallados son los siguientes:

1. Conocer características y experiencias en un contexto académico de cursos basados en proyectos (incluyendo el PIS) aplicando un proceso de desarrollo de software en equipos, donde los estudiantes aprendan, aplicando conocimiento y desarrollando habilidades de forma práctica, en una experiencia similar a la que se vive en la industria del software —también referido como «mundo real»—. A partir de esto seleccionar buenas prácticas a ser aplicadas en el PIS.
2. Entender los elementos y mejores prácticas que conforman diversos procesos y modelos de desarrollo existentes para contar con una perspectiva a nivel general que permita evaluar su uso en nuestra propuesta.
3. Proponer, crear y documentar un proceso de desarrollo de software ágil para que pueda ser aplicado por los estudiantes en el PIS, que colabore con su aprendizaje y que sirva para desarrollar software de calidad.

4. Evaluar el uso de PISAgile en una experiencia empírica durante un curso. Es decir, aplicar el proceso en el contexto para el cual fue creado buscando conocer si se cumplen los objetivos del curso (InCo, 2017): afianzar (e integrar) los conocimientos de ingeniería de software y otros cursos, y aplicar y desarrollar principalmente las habilidades blandas sin dejar por fuera las técnicas. Por aspectos vinculados al alcance de la tesis, el análisis se realiza desde la mirada parcial de uno de los actores: los estudiantes.
5. Conocer si el proceso sirve para que grupos de estudiantes del PIS logren construir productos de software de calidad en los plazos definidos.

1.3. Trabajo realizado y principales aportes

Realizamos una revisión sistemática de la literatura para abordar el primer objetivo detallado. La dirigimos particularmente a dos conferencias: la sección *Software Engineering Education and Training* de la *IEEE/ACM International Conference on Software Engineering* y la *IEEE Conference on Software Engineering Education and Training*. También se consideraron otros artículos sugeridos por expertos y una revisión de antecedentes para conocer si existen estudios con una revisión similar. Elegimos estructurar una búsqueda manual porque el objeto de búsqueda es muy amplio e inicialmente no contábamos con información para acotarlo. Seleccionamos estas conferencias porque son reconocidas en el ámbito de la enseñanza de Ingeniería de software. Como antecedente, encontramos un mapeo sistemático que caracteriza la educación de procesos de desarrollo de software (Heredia et al. 2015).

En su trabajo, Heredia et al. (2015) realizan una síntesis de los cursos encontrados por nivel académico de los participantes, métodos educacionales, métodos de entrega, procesos aplicados indicando la distribución porcentual entre los mismos, y presentan lecciones aprendidas de los cursos. En nuestra revisión, de un total de cincuenta artículos tenidos en cuenta, luego de aplicar nuestro protocolo de selección nos quedamos con siete que cumplieron nuestro criterio y agregamos los cinco sugeridos por expertos que también los cumplían.

Estudiamos y evaluamos las características generales, prácticas y experiencias de procesos ya definidos para cumplir con el segundo objetivo. Inicialmente, nos basamos en el resumen de procesos y *frameworks* presentado en el Handbook of Software Engineering (Cha et al. 2019). Luego agregamos el conocimiento adquirido por la autora de esta tesis en cursos realizados en el

marco de la maestría en ingeniería de software[1], charlas con especialistas y una revisión de evidencia —memoria organizacional del curso, experiencias de docentes y resultados de revisiones— con la que cuenta el PIS y la especificación del proceso disciplinado vigente en el curso. Todo ello lo profundizamos con literatura de referencia para cada uno de los procesos que surgieron del estudio.

Para proponer y crear el proceso realizamos una sistematización de las características del curso y los proceso actuales del PIS, seleccionamos aspectos de distintos procesos y marcos de trabajo, y complementamos con decisiones propias. Previo a documentarlo, realizamos una breve revisión de estándares y herramientas para especificar procesos. Decidimos construir una serie de documentos que describan el proceso con cierta navegabilidad mediante hipervínculos y usar como base el estándar SPEM V2.0 (*Software Process Engineering Metamodel*) (Object Management Group (OMG), 2008) con algunas modificaciones.

La creación y documentación del proceso la hicimos de forma paulatina e incremental conforme avanzaba el curso PIS en su edición 2020. A medida que se liberaban documentos, estos pasaban por un proceso de revisión y validación del cual participaban distintos perfiles de personas (docentes del curso, tanto actuales como de años anteriores, especialistas de la industria y la academia en procesos de desarrollo de software, ex-estudiantes del PIS y personas no vinculadas al desarrollo de software que ayudaron a evaluar la legibilidad del documento). La validación final fue realizada por uno de los docentes responsables del curso.

Para la evaluación del proceso utilizamos tres grupos de estudiantes del curso para estudiar su aplicación. Para ello seleccionamos los docentes y los grupos que participarían, definimos la forma de trabajo, realizamos una inducción de los docentes al proceso (brindando además apoyo durante toda la experiencia por parte de la autora de la tesis), y entregamos la especificación del proceso a todos los actores. La descripción detallada de la experiencia se puede ver en la Sección 4.3: *Experiencia 2020*. Luego se diseñó e implementó el método que permitió obtener la información necesaria para nuestra evaluación y análisis. El resultado de la evaluación se puede ver en el Capítulo 6: *Evaluación de PISAgile*.

[1]Maestría en ingeniería de software de la Facultad de ingeniería, Universidad de la República, Uruguay. Maestría en la cual se enmarca este tesis.

Con nuestro trabajo (1) retomamos la línea de investigación asociada al curso Proyecto de Ingeniería de Software, que desde hace nueve años estaba inactiva, abordando de esta manera una necesidad identificada por primera vez en el 2013 de contar con un proceso ágil para el PIS (incluyendo también una revisión de los procesos utilizados hasta el momento); (2) estudiamos cuáles eran las necesidades de un proceso ágil en el curso; (3) diseñamos y construimos un proceso ágil para el PIS; (4) evaluamos su adecuación a los fines para los que se creó aplicándolo en el curso; (5) estudiamos el proceso y los resultados de su aplicación con respecto a otros estudios vinculados a la enseñanza de procesos de desarrollo de software.

Los resultados de nuestro trabajo muestran, desde la percepción de los estudiantes, la importancia de contar con cursos **capstone** basados en proyectos como el nuestro para prepararlos para su desempeño profesional. También que aplicando en nuestro contexto procesos de desarrollo disciplinados en proyectos «reales», se puede cumplir el objetivo para el cual se creó el proceso: el aprendizaje de los estudiantes y construir software de calidad. Vinculado al aprendizaje, los estudiantes perciben mejoras en varias de las habilidades blandas y la necesidad de aplicarlas en un proyecto de desarrollo de software. Los aspectos positivos encontrados en esta primera evaluación nos llevaron a replicar la experiencia en 2021 y para este año (2022) se prevé que sea PISAgile el único proceso a usarse en el curso PIS.

1.4. Organización del documento

El informe de esta tesis cuenta con seis capítulos además del actual. En el Capítulo 2: *Procesos de desarrollo de software* se brinda una breve descripción teórica y referencias a los principales procesos mencionados en este trabajo. Luego en el Capítulo 3: *Enseñanza en ingeniería de software* se presenta una introducción a la enseñanza en ingeniería de software, los resultados de la revisión realizada de cursos académicos similares al PIS y sus principales características y experiencias.

En el Capítulo 4: *Curso Proyecto de Ingeniería de Software* se brinda una breve historia de nuestro curso y se describen sus principales características. Luego se comentan de forma detallada las particularidades de la experiencia vivida en la edición 2020 del curso, finalizando con los principales desafíos que se presentaron en dicha edición.

La especificación del proceso se encuentra en el Capítulo 5: *Descripción del proceso PISAgile*. Allí se brinda una descripción general del mismo y su arquitectura, y una especificación más detallada de los elementos del proceso. Es importante aclarar que el objetivo es brindar una visión general del proceso, resaltando sus principales particularidades. La especificación completa de PISAgile se encuentra publicada con acceso libre en el repositorio Zenodo[1].

En el Capítulo 6: *Evaluación de PISAgile* se presentan los resultados, análisis y discusión de la evaluación de PISAgile desde la percepción de los estudiantes, seguido por las limitaciones de la evaluación realizada y finalmente resaltando algunos comentarios que nos parecen importantes.

Finalmente, en el Capítulo 7: *Conclusiones y trabajo futuro* se presentan las conclusiones del trabajo, los principales aportes logrados, las limitaciones encontradas y posibles líneas en las cuales seguir avanzando.

[1] https://doi.org/10.5281/zenodo.6730363

Capítulo 2

Procesos de desarrollo de software

En este capítulo se realiza una breve descripción de los principales procesos de desarrollo de software y marcos de trabajo que son mencionados en los estudios de la revisión realizada o utilizados en la construcción de PISAgile. No pretendemos ser exhaustivos en la especificación de cada uno de ellos sino brindar una visión general de los mismos focalizando en los aspectos principales y utilizados junto con referencias que permitan su profundización. Estos son: Scrum, Proceso Modularizado Unificado y Medible (MUM), eXtreme Programing (XP), el Team Software Process (TSP), Agile unified process (AUP), Kanban y Design Thinking.

Scrum es un marco de trabajo para gestionar proyectos en el cual se definen los elementos esenciales que habilitan la gestión, no brindando instrucciones detalladas acerca de cómo llevar a cabo su aplicación (por mayor detalle ver, (Schwaber y Sutherland, 2020)). Tiene un enfoque iterativo e incremental, se encuentra fuertemente basado en el aprendizaje y toma de decisiones en base a lo observado a través de la experimentación.

Se basa en tres pilares (transparencia, inspección y adaptación), y promueve cinco valores para guiar el trabajo y comportamiento de las personas en el proyecto (foco, coraje, respeto, compromiso y apertura). Está organizado en tres componentes: el equipo scrum, un conjunto de artefactos, y un evento contenedor (*sprint*) en el cual se combinan cuatro eventos (o ceremonias) para inspección y adaptación.

El equipo scrum es un equipo multifuncional, autogestionado, de tamaño

pequeño, que tiene todas las habilidades necesarias para cumplir con los objetivos y generar valor durante el *sprint*. Consta de tres roles: un *scrum master*, un dueño de producto y desarrolladores. Los desarrolladores deben crear cualquier aspecto de un incremento utilizable en cada *sprint*. El dueño de producto es responsable por maximizar el valor del producto y de la gestión del backlog del producto. El *scrum master* es responsable por guiar al equipo de scrum en la aplicación de scrum, en que el equipo sea autogesionado y multifuncional, y asegurarse que los eventos se ejecuten acorde a las mejores prácticas.

El *sprint* es un evento que contiene los otros eventos necesarios para el logro de su objetivo. Durante el *sprint* se ejecutan los eventos: planificación del *sprint*; *daily* scrum donde el equipo evalúa el progreso y sincroniza el trabajo; revisión del *sprint* donde el equipo muestra el resultado del trabajo al dueño de producto y se chequea el cumplimiento del objetivo; y retrospectiva del *sprint* donde el equipo analiza lo ocurrido (vinculado a personas y relaciones, proceso y herramientas) y elabora un plan de acción de mejora.

Los artefactos de scrum representan el trabajo a realizar. Los principales son: el backlog del producto (lista ordenada de lo que se necesita para evolucionar el producto); el backlog del *sprint* (porción del backlog del producto que representa el trabajo a realizar durante el *sprint*); el incremento (es un peldaño concreto hacia el logro del producto, es software funcionando e incluye lo hecho en *sprint* anteriores); y la Definición de hecho (del inglés, Definition of done (DoD)) (acuerdo de calidad que se debe cumplir para determinar cuando un ítem está terminado).

El "Modularizado Unificado y Medible (MUM)" aplicado durante estos años en el PIS, es un modelo dirigido por planes, iterativo e incremental. Consta de dos dimensiones: la dimensión dinámica que lo divide en fases, iteraciones e hitos; y la dimensión del modelo que agrupa el proceso y disciplinas. Se compone de un modelo base que tiene dos extensiones, una enfocada a lenguajes orientados a objetos y otra a Genexus (por mayor detalle ver, (Pedrana y Bellini, 2005)).

eXtreme Programming (XP) es un método de desarrollo ágil iterativo compuesto por ciclos cortos, en pequeños y medianos equipos de desarrollo de software para ser aplicado principalmente frente a requisitos que estén especificados vagamente o que cambien rápidamente (por mayor detalle ver, (Beck y Andres, 2004; Wells, 2013)). Para su aplicación define seis roles (programador, cliente, encargado de pruebas, encargado de seguimiento, entrenador y gestor).

Está focalizado en el desarrollo aplicando Test Driven Development (TDD)[1] donde reconoce como único producto de valor el código ejecutable. Define iteraciones con actividades donde se planifica, diseña, codifica y se prueba para generar un incremento a partir de historias usuario[2]. XP propone una serie de valores y principios, para guiar el desarrollo y la aplicación de un conjunto de prácticas.

El Team Software Process (TSP) es un marco de trabajo para guiar a los equipos de desarrollo en los desafíos generales de gestión y de la ingeniería de software (poder tener previsibilidad en los costos y el cronograma, y mejorar en la productividad y la calidad del producto). Tiene embebido un proceso de mejora basado en evidencias. Algunos elementos que lo componen son una serie de fases, medidas, roles, guías del trabajo, mejores prácticas y herramientas de soporte al proceso para la recolección y análisis de datos. Los equipos de TSP son autogesionados (participan en la planificación, gestión y seguimiento de su propio trabajo). Hay un líder, un *coach*, y los integrantes desempeñan roles tradicionales (como ser desarrollador o tester) y ocho roles de gestión vinculados a distintos aspectos técnicos y de soporte (por ejemplo gestión de la calidad y gestión de la interfaz de cliente). Cada integrante debe asumir al menos un rol de gestión. Las responsabilidades de los roles de gestión son para asegurarse que se cubran las áreas claves guiando al equipo en ello, generar sentido de pertenencia y que el líder pueda enfocarse en sus responsabilidades (por un mayor detalle ver, (W. Humphrey, 2006; W. S. Humphrey, 2005)).

El Agile unified process (AUP) está basado en el Rational Unified Process (RUP). Es un proceso iterativo compuesto de cuatro fases (comienzo (del inglés, *inception*), elaboración, construcción y transición) y siete disciplinas (modelado, implementación, pruebas, desarrollo, gestión de la configuración, gestión del proyecto, entorno). Propone la aplicación de técnicas ágiles como Test Driven Development (TDD) y la refactorización, entre otras (por mayor información ver, (*Agile unified process (AUP)*, s.f.)).

Kanban es un método Lean[3] para gestionar el trabajo, permite visualizar

[1] El desarrollo guiado por pruebas es una práctica donde se escriben las pruebas antes que el código del software y luego se va modificando el código guiado por el resultado obtenido en las pruebas

[2] Especificación corta de funcionalidades que debe tener el software desde la perspectiva de un usuario

[3] forma de pensar y una práctica sobre la creación del valor necesario con menos recursos y menos desperdicio, consiste en la experimentación continua para lograr un valor perfecto con cero desperdicio.

el trabajo y cómo este se mueve a través de un flujo de trabajo definido (por mayor detalle ver, (*Kanban*, s.f.)).

Design thinking es una metodología compuesta por un conjunto de pensamientos y creencias que determinan el modo en que se aplica un proceso y herramientas para generar un mejor diseño. Busca dar respuestas (en nuestro caso) a las necesidades reales de un cliente o interesado en obtener el producto o resultados (por mayor detalle ver, (dinngo crowdfunding, s.f.)). Hay diversas implementaciones de la metodología, por ejemplo la propuesta por el Laboratorio de Innovación Social en Gobierno Digital[1].

[1]https://www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/lab

Capítulo 3

Enseñanza en IS

La enseñanza de las áreas de conocimiento de la ingeniería de software (IS), la implementación de cursos donde la base sea el trabajo de un equipo en un proyecto, y sobre el final de la carrera, proyectos donde implique considerar problemas del «mundo real» (i.e. la industria de desarrollo de software) haciendo énfasis en la práctica profesional, son algunas de las recomendaciones de la currícula de Ingeniería en computación (Joint Task Force on Computing Curricula, 2013, 2015).

En este capítulo se presenta el protocolo aplicado en la revisión sistemática, luego se realiza una introducción a la enseñanza en IS, particularmente en la basada en proyectos de desarrollo de software. Se presenta el resultado de la revisión sistemática de la literatura que diseñamos y condujimos, se caracterizan los cursos encontrados y los distintos aspectos que conforman su diseño: grado de cercanía al «mundo real» —según características de los clientes y proyectos—, características de los proyectos —tamaño, dedicación—, la conformación de los grupos y los roles involucrados, métodos de enseñanza, procesos de desarrollo de software, y objetivos de aprendizaje. Luego se presentan los resultados obtenidos en las experiencias de los estudios seleccionados. Finalmente se presenta una síntesis del aporte de la revisión para nuestro trabajo.

3.1. Protocolo de la revisión sistemática

La revisión sistemática de la literatura la dirigimos a dos conferencias de 2019: la sección *Software Engineering Education and Training* de la *IEEE/ACM International Conference on Software Engineering (ICSE-SEET*

2019) y la *IEEE Conference on Software Engineering Education and Training (CSEE&T 2019)*. Seleccionamos estas conferencias porque son reconocidas en el ámbito de la enseñanza de IS. También consideramos otros artículos sugeridos por expertos y una revisión de antecedentes para conocer si existen estudios con una revisión similar. Elegimos una búsqueda manual (no utilizando una cadena en un motor de búsqueda) porque el objeto de búsqueda es muy amplio e inicialmente no contábamos con información para acotarlo.

Como antecedente, encontramos un mapeo sistemático que caracteriza la educación de procesos de desarrollo de software (Heredia et al. 2015). Al no contar con un mapeo más actualizado, decidimos realizar la revisión sistemática abarcando el año 2019.

Para nuestra revisión sistemática de la literatura nos planteamos el siguiente objetivo: Queremos conocer cursos en donde se aborde la ingeniería de software desde una aplicación integral y práctica en la cual se aplique uno o varios procesos de desarrollo de software durante la ejecución de un proyecto de software que es llevado adelante por un grupo de estudiantes.

Nuestro criterio de selección de artículos se conformó con las siguientes condiciones:

- deben estar enfocados en la enseñanza basada en proyectos de desarrollo de software.
- los proyectos presentados deben ser ejecutados por un equipo de estudiantes trabajando en grupo de forma coordinada.
- la experiencia presentada debe buscar ser similar a la que se vive en la industria.
- parte de los objetivos de aprendizaje del curso presentado en el artículo debe ser aprender un proceso de desarrollo de software.
- preferentemente se busca que los artículos presenten objetivos de aprendizaje vinculados al desarrollo de conocimientos y habilidades de la ingeniería de software de forma integral.
- los cursos presentados deben estar enfocados en la enseñanza de grado (aunque puede haber participación de estudiantes de posgrado).
- los cursos no deben ser de carácter introductorio a la ingeniería de software. Esto es, que no sea el primer acercamiento del estudiante a la temática.

Definimos y ejecutamos un proceso de selección, aplicado por una única

persona, para filtrar los trabajos de las conferencias según los criterios de selección definidos. Aplicamos un primer filtrado (preselección) a partir de la lectura del título del artículo. Luego se continuó filtrando por el resumen, y en caso de no ser suficiente para tomar una decisión, se avanzó en la lectura del trabajo (introducción, conclusiones, llegando en algunos casos a la lectura completa) para poder ser aceptado o descartado.

Luego de aplicar nuestro criterio de selección, de un total de cincuenta artículos, donde treinta surgieron de *CSEE&T 2019* y veinte de la preselección de los presentados en la *ICSE-SEET 2019*, nos quedamos con siete (ver tabla 3.1) que lo cumplieron y sumamos los cinco sugeridos por expertos por fuera de la revisión que también los cumplían.

La extracción se enfocó en conocer los siguientes aspectos de los estudios: (i) aspectos que conforman el diseño de los cursos basados en proyectos: grado de cercanía al «mundo real» —según características de los clientes y proyectos—, características de los proyectos —tamaño, dedicación—, la conformación de los grupos y los roles involucrados, métodos de enseñanza, procesos de desarrollo de software, objetivos de aprendizaje; y (ii) los resultados que obtuvieron.

3.2. Tipos de cursos y método de enseñanza

Existen diversos tipos de cursos utilizados para la enseñanza en ingeniería de software. Entendemos que estos, en su diseño, tienen en cuenta aspectos tales como los conocimientos y habilidades a adquirir por los estudiantes, grado de avance en su currícula, la profundidad con la que se quieren abordar las distintas áreas de conocimiento de la ingeniería de software, entre otros.

Para la educación, aprendizaje y entrenamiento en procesos de desarrollo de software se aplican distintos métodos de enseñanza. Heredia et al. (2015) presentan un mapeo sistemático de estos. El método más tradicional incluye clases, y ejercicios o pequeños proyectos para poner en práctica los conocimientos. Uno de los métodos más utilizado son las experiencias de trabajo en proyectos que sean lo más cercanas posibles al «mundo real». Otros métodos se componen de simulaciones combinadas con tareas, tutoriales y sesiones de trabajo, aunque dicha combinación es más orientada a profesionales y graduados. Por último, en un solo caso se presenta la ludificación como método de enseñanza.

Los cursos avanzados en ingeniería de software no solo deben estar cen-

Tabla 3.1: Artículos seleccionados

Título	Autores	Publicado en	Año	Universidad	País
Collaborating with Industrial Customers in a Capstone Project Course: The Customers' Perspective	M. Paasivaara, J. Vanhanen, C. Lassenius	ICSE-SEET 2019	2019	Universidad Aalto	Finlandia
Facilitating Entrepreneurial Experiences through a Software Engineering Project Course	H. Burden, J. P. Steghöfer, O. Hagvall Svensson	ICSE-SEET 2019	2019	Universidad Tecnológica Chalmers	Suecia
A Grading Schema for Reinforcing Teamwork Quality in a Capstone Course	M. C. Bastarrica, D. Perovich, F. J. Gutierrez, M. Marques	ICSE-SEET 2019	2019	Universidad de Chile	Chile
Attitudes, Beliefs, and Development Data Concerning Agile Software Development Practices	C. Matthies and J. Huegle and T. Dürschmid and R. Teusner	ICSE-SEET 2019	2019	Universidad de Potsdam	Alemania
How Much Authenticity can be Achieved in Software Engineering Project Based Courses?	Z. S. H. Abad and M. Bano and D. Zowghi	ICSE-SEET 2019	2019	Universidad de Calgary	Canadá
Teaching Software Engineering with Free Open Source Software Development: An Experience Report	A. Tafliovich and F. Estrada and T. Caswell	CSEE&T 2019	2019	Universidad de EEUU no identificada por los autores	Estados Unidos
Practicing Scrum in Institute Course	H-J. Hsu and E. Lin and K. Chang and E. Hsiao	CSEE&T 2019	2019	Feng Chia University	Taiwán
The Effect of Real-World Capstone Project in an Acquisition of Soft Skills among Software Engineering Students	J. Khakurel and J. Porras	CSEE&T 2020	2020	Universidad Politécnica de Lappeenranta	Finlandia
What Can Students Get from a Software Engineering Capstone Course?	M. C. Bastarrica and D. Perovich and M. M. Samary	ICSE-SEET 2017	2017	Universidad de Chile	Chile
Enhancing the Student Learning Experience in Software Engineering Project Courses	M. Marques and S. F. Ochoa and M. C. Bastarrica and F. J. Gutierrez	IEEE Transactions on Education	2018	Universidad de Chile	Chile
Learning by Doing: Goals and Experience of Two Software Engineering Project Courses	M. Moore and C. Potts	Software Engineering Education, 7th SEI CSEE Conference, Proceedings	1994	Instituto de Tecnología de Georgia	Estados Unidos
Process activities in a project based course in software engineering	T. Germain and P. N. Robillard and M. Dulipovici	32nd Annual Conference on Frontiers in Education, IEEE	2002	Escuela Politécnica de Montreal	Canadá

ICSE-SEET 2019: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)
CSEE&T 2019: 2019 IEEE 31st Conference on Software Engineering Education and Training (CSEE&T)
CSEE&T 2020: 2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T)
ICSE-SEET 2017: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)

trados en la adquisición de conocimientos sino que deben preparar a los estudiantes para conocer los desafíos y dificultades de trabajar en proyectos de desarrollo de software, y desarrollar las habilidades necesarias para afrontarlos. Por eso es importante que en los cursos se busque el aprendizaje a través de la reflexión acerca del desempeño en un proyecto (Abad et al. 2019).

En la revisión que realizamos encontramos que las universidades en general presentan dos tipos de cursos de este estilo, en donde se combinan determinados métodos de enseñanza: cursos basados en proyectos y cursos *capstone*. El primero consiste en cursos donde el aprendizaje se da a través de la participación del estudiante en un proyecto. Estos cursos pueden estar (o no) enfocados en un aspecto en particular del desarrollo de software. Por otro lado, los cursos *capstone* son generalmente impartidos al final de la carrera, donde se busca que el estudiante aplique de forma integral el conocimiento y las habilidades adquiridas a lo largo de la misma. En general es el último curso tomado por los estudiantes y una característica común de estos cursos es que los estudiantes deben enfrentarse y resolver problemas similares a los que se encuentran en la «mundo real» y a su vez contar con alguien que los oriente (Bastarrica et al. 2017). En la mayoría de los casos ambos tipos son combinados, obteniendo cursos *capstone* basados en proyectos.

Moore y Potts (1994) presentan uno de los primeros artículos acerca de cursos basados en proyectos para el área de ingeniería de software. El curso presentado ("*Real World Lab*") se creó bajo el argumento de que el desarrollo de las habilidades de los estudiantes en ingeniería de software se da en parte por la experiencia en proyectos reales (i.e. de la industria de desarrollo de software). Concepto que compartimos plenamente y que es, sin dudas, el principal motivo para mantener actualizados los procesos de desarrollo que utilizamos en el curso Proyecto de Ingeniería de Software de nuestra facultad.

3.3. Diseño de cursos basados en proyectos

En esta sección se presentan aspectos extraídos de la revisión que conforman el diseño de los cursos basados en proyectos independientemente de que sean *capstone* o no. Como se indica en nuestros criterios de selección no nos enfocamos exclusivamente en los cursos *capstone* (pero no perdiendo de vista su importancia, ya que el PIS es uno de ellos). Esto es porque nuestro foco no estaba solo en el curso en si mismo, sino en el proceso y que lo princi-

pal fuera un curso basado en proyectos aplicando un proceso de desarrollo de software por un grupo de estudiantes. Entonces entendímos que de esta forma contemplámos un mayor número de experiencias. También detectamos en varios casos que si bien el curso no es reconocido como tal, tiene características similares a uno *capstone*, ya que depende de la implementación de la currícula.

El diseño de cursos basados en proyectos en ingeniería de software es variable, no hay un estándar definido y presentan varios desafíos. Hilburn y Humphrey (2002) describen aspectos generales a considerar para estos cursos: identificar los objetivos del curso; si hay restricciones de tiempos de dedicación o es la primera experiencia en proyectos en equipo, se deben utilizar problemas bien definidos en lugar de experiencias basadas en el «mundo real»; usar un proceso de desarrollo de software definido; asegurarse que los estudiantes sigan el proceso (disciplina de proceso) y que los docentes pasen del típico rol de impartir clases a ser un *coach*.

En el resto de la tesis, cuando nos referimos a cursos basados en proyectos, estos son cursos donde un grupo de estudiantes desarrolla software siguiendo algún proceso de desarrollo —aunque sea ad hoc— y en donde se brinda una experiencia similar a la que se vive en la industria de software —también referido como «mundo real»—. Esto mismo aplica para proyectos y clientes reales.

3.3.1. Grado de realidad en la experiencia

Un aspecto importante de los cursos basados en proyectos es el grado de acercamiento a lo que sucede en la industria. Inciden diversos factores para lograr esta cercanía, por ejemplo, quién formula el trabajo (o proyecto) de los estudiantes. El principal desafío es mantener el balance entre los beneficios de una experiencia lo más cercana al «mundo real» y un entorno formativo protegido para que los estudiantes avancen en su aprendizaje. Por ello, en algunos primeros cursos de ingeniería de software se aplican entornos simulados (Matthies et al. 2019), donde por ejemplo los clientes son los estudiantes (Abad et al. 2019). Para aumentar el grado de realismo, en otros cursos se trabaja con clientes de la industria que presentan proyectos (Bastarrica et al. 2017; Burden et al. 2019; Paasivaara et al. 2019), clientes del entorno académico (Moore y Potts, 1994), o mediante la participación directa en proyectos que se encuentran en desarrollo en la industria, particularmente se presenta un caso

en el cual se suman como contribuyentes a una comunidad de software libre de código abierto (Tafliovich et al. 2019).

Una forma de aumentar el grado de realidad en el vínculo con el cliente, es mediante la aplicación de procesos de selección similares a los que ocurren en la industria; los equipos de estudiantes envían sus currículums vitae a los clientes y de esta forma se postulan al proyecto que les interese, y luego se realizan las asignaciones de común acuerdo entre las partes (Paasivaara et al. 2019). También la formalidad del vínculo de los clientes con la universidad es variable, por ejemplo, en algún caso se establece una firma de contrato (Paasivaara et al. 2019).

Otro factor importante es la locación de trabajo de los estudiantes. En algunos casos se les recomienda a los estudiantes compartir la mayor parte del tiempo en una misma ubicación (ver, por ejemplo, (Paasivaara et al. 2019)) pero no se les proporciona un espacio físico determinado como sí sucede en otros cursos (Hsu et al. 2019). Si bien hay una tendencia a que en los cursos más avanzados el grado de realidad es mayor, esto no es determinante. Bastarrica et al. (2017) reportan que aparte de trabajar con un cliente de la industria, los estudiantes trabajan en las instalaciones del cliente cumpliendo con la dedicación establecida. Incluso, en este caso en particular, el cliente realiza pagos por el trabajo de los tutores del curso.

3.3.2. Proyectos y dedicación

Las características de los proyectos varían en los distintos cursos. Hay cursos donde no hay requisitos particulares sobre los proyectos, no existiendo limitantes respecto a las tecnologías ni el dominio del mismo (Paasivaara et al. 2019). En otros casos la limitación se define por el alcance, como es el caso de la construcción de un prototipo funcional (Burden et al. 2019). Hay cursos en los cuales hay exigencias respecto a las tecnologías (o plataformas) (Abad et al. 2019; Tafliovich et al. 2019). El tamaño de los proyectos también es variable y esto se refleja en su duración, hay de corta duración —hasta diez semanas— (Burden et al. 2019; Hsu et al. 2019), otros de una duración media —doce a dieciséis semanas— (Abad et al. 2019; Bastarrica et al. 2017; Bastarrica et al. 2019; Germain et al. 2002; Marques et al. 2018; Matthies et al. 2019; Tafliovich et al. 2019), y de duración media a larga —seis meses en adelante— (Paasivaara et al. 2019). En general la dedicación de los estudiantes ronda unas 16

hs. semanales (Bastarrica et al. 2017; Bastarrica et al. 2019) aunque algunos cursos tienen una dedicación inferior (10 hs.) (Germain et al. 2002). Si bien en comparación con proyectos de desarrollo reales no son tantas horas, en contextos educativos son normalmente cursos de alta carga en comparación con otros cursos de las carreras.

3.3.3. Conformación de grupos y roles

La conformación de los grupos y los roles también es variable. Los roles en algunos casos son asociados a las actividades generales del propio curso y en otros acorde al proceso de desarrollo que aplican. En la mayoría de los casos los grupos están conformados por estudiantes que tienen similar nivel académico y de experiencia (ver, por ejemplo, (Abad et al. 2019)), en otros hay combinaciones de estudiantes con distinto nivel académico dentro de una misma carrera (ver, por ejemplo, (Moore y Potts, 1994)) o mediante la combinación de estudiantes de grado con estudiantes avanzados, por ejemplo de maestría o doctorado (Hsu et al. 2019; Paasivaara et al. 2019).

El tamaño de los grupos también varía, algunos tienen tamaño pequeño —hasta seis integrantes— (Burden et al. 2019; Germain et al. 2002; Tafliovich et al. 2019), otros tienen un tamaño medio —siete a nueve integrantes— (Matthies et al. 2019; Paasivaara et al. 2019), y otros tienen un tamaño grande —más de nueve integrantes— como es el caso de nuestro curso. En los casos presentados por Bastarrica et al. (2017) y Marques et al. (2018) se admiten principalmente grupos pequeños, aunque puede existir alguno de tamaño medio. En varios de los artículos que surgen de la revisión, no se encuentran definidos los roles que toman los actores (estudiantes, docentes, clientes, etc.) en el proceso de desarrollo y de enseñanza. Sí hay algún caso particular donde los grupos de estudiantes tienen un doble rol, hacen de cliente de otro grupo y a su vez ellos llevan a cabo un proyecto asignado cuyo cliente es otro grupo (Abad et al. 2019).

3.3.4. Métodos de enseñanza

En general en los cursos basados en proyectos, se realizan diversas combinaciones de métodos de enseñanza con el fin de enriquecer la experiencia del estudiante y mantener un entorno de trabajo controlado cuyo foco principal debe ser el aprendizaje en lugar de la construcción de un producto de softwa-

re per se. En estos cursos, el mayor esfuerzo de los estudiantes está centrado en su participación en el proyecto, a excepción de algunos casos donde son dedicadas varias semanas a la enseñanza de teoría y práctica para que luego lo apliquen en el proyecto (ver, por ejemplo, (Hsu et al. 2019)). Dos métodos que la mayoría utiliza para complementar el trabajo de los estudiantes durante el proyecto son: clases esporádicas y sesiones de trabajo (Burden et al. 2019; Matthies et al. 2019; Paasivaara et al. 2019), y el aprendizaje colaborativo entre los estudiantes (ver, por ejemplo, (Abad et al. 2019)). De la revisión realizada surge que en general el autoestudio es ampliamente utilizado para el aprendizaje de lenguajes de programación, herramientas y tecnologías, y en algún caso tienen el apoyo del cliente (Paasivaara et al. 2019). También aplican métodos adicionales como ser la ludificación y la obtención de certificaciones (Paasivaara et al. 2019) .

Se implementan diversas alternativas para acompañar y guiar a los estudiantes en los proyectos proporcionando diferentes grados de acompañamiento. Dentro de las que brindan acompañamientos más cercano lo hacen por medio de un *coach* (Bastarrica et al. 2019; Paasivaara et al. 2019), un tutor (Bastarrica et al. 2017; Matthies et al. 2019), un monitor que aplica un método de monitoreo formativo —monitoreo semanal reflexivo (del inglés, reflexive weekly monitoring (RWM))— que utiliza prácticas que llevan a la autoreflexión de los estudiantes y el aprendizaje colaborativo (Marques et al. 2018), o un asistente que aparte de brindarles guía evalúa el progreso del equipo mediante un seguimiento semanal (Tafliovich et al. 2019). En otros casos se direcciona a los grupos por medio de asignaciones de tareas concretas que acompañan la evolución del proyecto y hacen visibles las dificultades con las que se pueden encontrar aplicando un enfoque de evaluaciones reales (del inglés, *authentic assessment*) (Abad et al. 2019), o aplicando el método de enseñanza justo a tiempo (del inglés, *Just-in-time*) incorporando los temas en las clases (o laboratorios) justo antes que el estudiante deba aplicarlo en el proyecto (Tafliovich et al. 2019).

En algunos casos, el vínculo con la industria va más allá de su participación como cliente, sino que existe una responsabilidad en la formación; por ejemplo, los profesionales comparten experiencias con los grupos en sesiones (Paasivaara et al. 2019) o facilitan algunas de las ceremonias de Scrum (Hsu et al. 2019).

Si se consideran los cursos analizados, no todos se ubican en el mismo grado de avance de la carrera. Hay algunos que se ubican en los primeros años

(Abad et al. 2019), otros en los últimos luego de que los estudiantes ya transitaron por otros cursos vinculados a la ingeniería de software y tienen cierta experiencia (Bastarrica et al. 2017; Bastarrica et al. 2019; Germain et al. 2002; Matthies et al. 2019; Tafliovich et al. 2019), y Paasivaara et al. (2019) presenta un caso particular que tiene estudiantes con distinto avance curricular y pertenecen a carreras distintas. Las diversas combinaciones de métodos de enseñanza dependen del contexto y la ubicación del curso dentro de la currícula de los estudiantes. En varios casos los cursos basados en proyectos se ubican luego de que el estudiante ya ha transitado por otros cursos que le dieron cierta formación general, y se implementan cursos de proyectos *capstone* (Matthies et al. 2019; Paasivaara et al. 2019; Tafliovich et al. 2019) para que el estudiante sintetice los conocimientos ya adquiridos y avance en su desarrollo profesional. En otros casos, principalmente en los cursos iniciales, se aplica el aprendizaje basado en proyectos pero con un alcance menor; por ejemplo en Abad et al. (2019) aplican simulación de roles acompañado de tareas y problemas reales definidos por los docentes para simular el contexto. En algunos casos no depende solo de la currícula sino que incide también el desarrollo de la persona y la forma en que es promovido. Este es el caso presentado por Moore y Potts (1994) donde coexisten tres niveles de estudiantes, y el método de enseñanza es adaptado en cada caso. En el primer nivel el estudiante es responsable por tareas pequeñas y se les da instrucciones precisas de trabajo, en el segundo se les brinda menor dirección y pueden liderar subproyectos, y en el tercer nivel pueden ser líderes de equipo.

3.3.5. Procesos de desarrollo de software

Respecto a los procesos de software, no hay un consenso de cuál debe utilizarse ni cuáles son los más adecuados para la enseñanza, sino que depende de cada caso según el contexto y la experiencia. Sin embargo, Hilburn y Humphrey (2002) recomiendan que sea un proceso bien definido donde se cuente con una descripción detallada, especificación de roles y responsabilidades, datos a recolectar para seguimiento y calidad, y un enfoque de desarrollo cíclico.

En algunos casos se toman procesos conocidos y en otros se define o se adapta un proceso específicamente para el curso. La mayoría reporta el uso de metodologías ágiles (al igual que la revisión de Heredia et al. (2015)). Algunos aplican Scrum (Burden et al. 2019; Hsu et al. 2019) tal como se especifica en

su guía oficial (Schwaber y Sutherland, 2020), en otro no se indica un proceso de forma específica (Khakurel y Porras, 2020) o un único proceso a utilizar (Matthies et al. 2019), y en otros casos se le da libertad al estudiante para que elija el proceso a seguir brindándole ciertas opciones pero con ciertas restricciones, como exigir que se aplique test driven development (TDD) (Abad et al. 2019). En otros casos, como el presentando por Tafliovich et al. (2019), se compone de un proceso iterativo guiado por fases/entregables que se encuentran asociados a actividades a realizar en un proyecto de desarrollo de software. Germain et al. (2002) reportan la aplicación del proceso de desarrollo UPEDU (*Unified Process for EDUcation*), el cual es una adaptación de RUP (*Rational Unified Process*). Marques et al. (2018) aplican una adaptación del proceso SSP (*Simple Software Process* (Ochoa et al. 2006)) y lo definen como un proceso disciplinado con un trabajo poco acoplado por parte de los participantes, es decir, realizan actividades de forma individual con coordinaciones esporádicas del equipo.

Los cursos que proponen el uso de Scrum definen diversas implementaciones, en algunos casos se realizan cambios sustanciales al proceso y en otros solo se establecen parámetros de implementación. Generalmente cuando hay un cliente real (i.e. cliente, que no es docente del curso, que presenta un proyecto análogo a los de la industria de desarrollo de software, independientemente del ámbito al que pertenezca —industria, academia o Estado—) lo hace en el rol de dueño del producto y debe participar activamente en algunas ceremonias (Paasivaara et al. 2019). Paasivaara et al. (2019) proponen una variación respecto a Scrum ya que durante el primer sprint se realiza un configuración inicial del proyecto —se conoce el equipo y el cliente, se crea una visión del producto, se crea el backlog inicial, se seleccionan y estudian las tecnologías, se definen herramientas, etc—. En algunos casos se establecen seis sprints de tres semanas de duración cada uno (ver, por ejemplo, (Paasivaara et al. 2019)), en otros casos como el presentado por Burden et al. (2019) los sprints tienen una duración de una semana, y en otro se presenta la implementación de Scrum más corta con dos sprints de dos semanas cada uno (Hsu et al. 2019). A su vez, Hsu et al. (2019) presentan algunas particularidades respecto a la forma en que se cubren determinados roles; el rol de *scrum master* es cubierto en las reuniones diarias por medio del profesor, y durante la planificación, revisión y retrospectiva lo ejerce un *scrum master* certificado perteneciente a la industria, luego el dueño del producto es elegido por el propio equipo pero no aclara

quién debe ser.

Matthies et al. (2019) proponen una combinación de procesos, al principio los estudiantes aplican Scrum —con algunas modificaciones a causa del contexto— durante cuatro sprints con una duración de dos o tres semanas, luego de ello hacia el final del proyecto aplican Kanban. En este caso los roles de dueño del producto y *scrum master* son miembros del grupo de estudiantes mientras que el resto son los desarrolladores, y un aspecto a resaltar es que el tutor participa de las principales ceremonias.

Un caso particular de combinación de procesos es reportado por Moore y Potts (1994) donde en un mismo proyecto cada estudiante puede estar aplicando un proceso distinto dependiendo de su nivel de desarrollo y visibilidad del proyecto. En el primer nivel el estudiante es responsable por tareas pequeñas y se les da un modelo de proceso a seguir (símil cascada) y estándares. En el segundo y tercer nivel tienen autonomía en los métodos y técnicas a utilizar.

3.3.6. Objetivos de aprendizaje

Un ingeniero de software debe poder desarrollar productos de calidad cumpliendo con restricciones de tiempo y los compromisos adquiridos. Para ello debe aprender a manejar los desafíos que surjan y a aplicar los mejores métodos (W. Humphrey, 2005). Para producir software de calidad cada individuo que participa en su construcción debe hacer trabajo de calidad (W. Humphrey, 2000). Parte de ello es trabajar de forma integrada y armoniosa para lograr los mejores resultados, esto se logra aplicando un proceso disciplinado y estandarizado, y estándares en lugar de que cada uno aplique su propio proceso, que es lo que generalmente se tiende a hacer (W. Humphrey, 2005). Esto hace que el área de procesos sea un área de conocimiento de particular importancia para la formación de profesionales vinculados al desarrollo de software.

Uno de los principales objetivos de aprendizaje en los cursos basados en proyectos, es que los estudiantes entiendan los desafíos presentes en proyectos de desarrollo de software (Paasivaara et al. 2019) y que aprendan a aplicar un proceso de desarrollo de software (Burden et al. 2019). Otro caso tiene como foco principal enseñar las mejores prácticas de colaboración y desarrollo ágil en un entorno de múltiples equipos autoorganizados en un proyecto del «mundo real» (Matthies et al. 2019). En otros casos, como los presentados por Paasivaara et al. (2019) y Hsu et al. (2019), hacen referencia al aprendizaje de

un proceso en específico como objetivo central del curso. En la mayoría de los casos, sumado al aprendizaje del proceso, se busca también el aprendizaje de la práctica de los métodos asociados a las distintas áreas de conocimiento y a la práctica profesional (ver, por ejemplo, (Tafliovich et al. 2019)).

La adquisición de habilidades blandas (también conocidas como "habilidades no técnicas", "habilidades sociales", "habilidades personales") ha ganado relevancia dentro de la ingeniería de software, tanto para desarrolladores de software como para investigadores (Matturro et al. 2015). Esto se debe principalmente a que la ingeniería de software es una disciplina con un fuerte componente humano y social. Eso es consecuencia de la alta interacción humana para construir un software, la relevancia de los sistemas interactivos, y la importancia de construir software de calidad acorde a las necesidades (y restricciones) de los interesados. Esto hace que muchas veces los principales desafíos a enfrentar en un proyecto de desarrollo de software estén vinculados a las habilidades blandas antes que a las técnicas.

Paasivaara et al. (2019) plantean como objetivo de aprendizaje que los estudiantes puedan abordar desafíos que se encuentran más vinculados a las habilidades blandas que a las técnicas, entre ellos se encuentran: entender los requisitos de software, colaboración con el cliente y trabajo en equipo. En la mayoría de los estudios analizados se presentan las habilidades blandas como objetivo de aprendizaje de los cursos (Abad et al. 2019; Bastarrica et al. 2017; Bastarrica et al. 2019; Khakurel y Porras, 2020; Marques et al. 2018; Tafliovich et al. 2019). Si bien pueden diferir en cuáles son aquellas habilidades que deben adquirir los estudiantes, algunas siempre están presentes: comunicación, negociación, colaboración, organización y trabajo en equipo. Para evaluar la colaboración con sus compañeros, Tafliovich et al. (2019) proponen una evaluación de pares.

Si bien el aprendizaje de aspectos tecnológicos se encuentra en general en un segundo plano, algunos cursos tienen dentro sus objetivos que los estudiantes aprendan nuevas tecnologías (Marques et al. 2018). Normalmente estas deben ser aprendidas por cada estudiante (Paasivaara et al. 2019) y en ocasiones de forma colaborativa entre ellos (Burden et al. 2019).

Algunos casos agregaron objetivos que no están presentes en el resto: Burden et al. (2019) presentan que su objetivo es que los estudiantes incorporen competencias empresariales, y que aprendan a reflexionar sobre su propia estrategia de aprendizaje y la de su equipo. En otros casos como el prestando

por Tafliovich et al. (2019) más allá de los objetivos de aprendizaje se busca motivar a los estudiantes.

3.3.7. Resultados obtenidos en estudios seleccionados

En esta sección se presentan los resultados obtenidos en las experiencias de los distintos estudios que comprenden la revisión de la literatura. Dado que los distintos artículos hacen foco en distintas características del curso o proceso, se presentan agrupando, en primer instancia, los que observan aspectos particulares del curso o proceso, y posteriormente los que estudian la adquisición de conocimientos y el desarrollo de habilidades durante el curso. Para estos últimos se mencionan los conocimientos y habilidades a adquirir, y luego los resultados (informando qué hicieron y sus conclusiones). En general, salvo que existan similitudes, se presentan comentando artículo por artículo.

Los distintos estudios se centran en observar ciertos aspectos particulares del curso. Paasivaara et al. (2019) buscan conocer la perspectiva del cliente con respecto a su participación en los proyectos. Les interesa conocer por qué participan del curso, qué características consideran apropiadas para los proyectos, la colaboración de los clientes con los estudiantes y qué consejos dar a futuros clientes. Los principales motivos por los cuales participan los clientes son para reclutar personal, obtener resultados del proyecto de software e investigar nuevas tecnologías. Solo dos empresas reportaron que el principal motivo es desarrollar «software real» (i.e. software para ser realmente utilizado, independientemente del ámbito al que pertenezca, industria, academia o Estado). Los clientes consideran que la visión general del proyecto debe: ser fácil de entender, ser importante para la empresa para ser motivante —pero no crítico—, tener un equilibrio entre que el alcance sea adecuado para el curso y presente un desafío para los estudiantes, contar con una temática motivadora, y reflejar el trabajo en la empresa para cumplir con el objetivo de reclutamiento. Los clientes resaltan la importancia de invertir tiempo en la colaboración y comunicación, principalmente al principio del proyecto. Además, se identificaron trece consejos para los nuevos clientes.

Matthies et al. (2019) estudiaron el comportamiento de los equipos de estudiantes y la aplicación de prácticas ágiles. Utilizaron encuestas para obtener la percepción de los estudiantes sobre prácticas ágiles. Los estudiantes percibieron las siguientes prácticas como las que se vinculan más con los valores ágiles:

la propiedad colectiva del código, uso de un sistema de control de versiones y no trabajar hasta último momento.

Hsu et al. (2019) presentan las lecciones aprendidas en el curso. Resaltan la influencia positiva de la retrospectiva en el proyecto, y mencionan como lecciones aprendidas: mantener un horario fijo para las reuniones diarias de Scrum (que como adaptación al curso se realizaron dos por semana), que cada equipo cuente con un espacio de trabajo común y el uso de tableros electrónicos como una buena opción para equipos distribuidos. Un aspecto particularmente interesante es que dentro de las lecciones aprendidas plantean que tener a un profesor como *scrum master* no es escalable, por lo que sugieren que los estudiantes sean entrenados para cumplir con el rol y que sea rotativo dentro del equipo.

En varios de los artículos se presentan los resultados de aprendizaje vinculados a la adquisición de conocimientos y el desarrollo de habilidades por parte de los estudiantes. Estos temas están estrechamente vinculados con nuestro estudio. A continuación se brinda un detalle de los mismos.

Los diferentes estudios contemplaron distintos conjuntos de habilidades y conocimientos a adquirir. Abad et al. (2019) definieron dos conjuntos de habilidades: habilidades para resolver problemas (comprensión del problema, planificación, comprensión de la calidad y adaptabilidad), y habilidades sociales (manejo de personas, negociación, organización). En el artículo presentado por Khakurel y Porras (2020) se plantean objetivos de aprendizajes asociados a: cómo realizar actividades vinculadas a los proyectos —por ejemplo planificación y estimación—, conocimientos técnicos —por ejemplo, aprender a utilizar herramientas de desarrollo—, y otros más globales —por ejemplo, implantar un sistema de software—. Bastarrica et al. (2017) plantean como objetivo de aprendizaje principal que los estudiantes se interioricen en cuán relevante es desarrollar habilidades blandas críticas para el éxito de los proyectos. En el curso presentado por Burden et al. (2019) se busca que los estudiantes aprendan a enfocarse en la entrega de valor a los interesados en lugar del producto. Definen objetivos específicos de aprendizaje —en lugar de habilidades— y se los vincula con un conjunto de competencias necesarias para el emprendurismo basadas en el marco de competencias para el emprendurismo (Bacigalupo et al. 2016), que consideran importantes para un ingeniero de software. Como objetivos específicos definen: aprender a especificar, implementar y evaluar sistemas basados en lo que los interesados perciben como valor; aplicar un

proceso de desarrollo de software estructurado; reflexionar sobre las estrategias de aprendizaje propias y del equipo; entre otros. Las competencias están organizadas en tres bloques: ideas y oportunidades —competencias para identificar oportunidades para crear soluciones a desafíos, adherirse a una visión definida y evaluar el valor de las ideas—, recursos —competencias que permitan identificar las necesidades de recursos de manera eficiente—, y en acción —competencias vinculadas a iniciar procesos, enfrentar desafíos y definir objetivos. También adaptarse y tomar decisiones—. Estas competencias son muy similares a las necesarias para el desarrollo de software.

Abad et al. (2019) luego de realizar un análisis cuantitativo y cualitativo teniendo en cuenta trabajos reflexivos entregados por los estudiantes, concluyen que el diseño del curso —basado en proyectos combinado con evaluaciones reales (del inglés, *authentic assessment*)— ayuda a los estudiantes a mejorar las habilidades para resolver problemas y las habilidades sociales. Respecto a las habilidades para resolver problemas, la que presentó una mayor mejora significativa fue la comprensión del problema, seguida por la planificación. Las que resultaron más desafiantes durante las primeras iteraciones fueron la comprensión de la calidad y adaptabilidad. Los equipos le quitaron prioridad a la comprensión de la calidad para no afectar los plazos, un aspecto que mejoraron luego de considerar la retroalimentación recibida en la segunda iteración. Respecto a las habilidades sociales, los estudiantes encontraron más desafiantes las actividades que involucraban el manejo de personas. Al ser consultados acerca de su percepción respecto a si habían aprendido durante el curso, en una escala del 1 al 10 se obtuvo una media de 6.5.

Khakurel y Porras (2020) realizan un análisis temático sobre reportes solicitados a los estudiantes donde estos reflexionan sobre el proyecto, para identificar conceptos clave vinculados al desarrollo de habilidades blandas. Identifican un total de quince habilidades blandas: comunicación interna, comunicación externa, habilidades organizativas, pensamiento crítico y resolución de problemas, gestión de conflictos, trabajo en equipo y colaboración, habilidades interpersonales, habilidad para trabajar bajo presión, orientación al cliente, apertura a los cambios y adaptabilidad, liderazgo, responsabilidad, motivación, afán de aprender, y habilidad para trabajar independientemente. La comunicación externa fue la habilidad más identificada seguida por las habilidades organizativas, mientras que solo un estudiante identificó trabajar bajo presión. La frecuencia con las que aparecen las habilidades los conduce a concluir que

el diseño del curso aporta a mejorar la mayoría de las habilidades blandas necesarias en la industria del software. Como resultado adicional presentan que el contar con un cliente externo real hace que los equipos tengan un mayor compromiso.

Como conclusión Bastarrica et al. (2017) reportan que a medida que avanza el proyecto, los estudiantes ven que el valor de las habilidades blandas aumenta, mientras que el de las habilidades técnicas disminuye. Sucede algo similar con la dificultad percibida de lograr ambos tipos de habilidades, a excepción de la negociación con el cliente donde la percepción de la dificultad disminuye. Estos resultados los obtienen a partir de la ejecución de una encuesta a los grupos al comienzo y al final del proyecto, donde evalúan el valor relativo y la dificultad que perciben al abordar distintas dimensiones involucradas en el éxito de los proyectos.

Marques et al. (2018) muestran que un curso diseñado con la inclusión de un método de monitoreo formativo, ayuda a que los estudiantes sean más efectivos —capacidad de centrarse en los requisitos obligatorios del proyecto—, coordinados —capacidad de trabajar juntos para lograr los objetivos del proyecto—, y experimenten un mayor sentido de pertenencia en el equipo —percepción de los miembros del equipo de compartir un objetivo en común—. Por otro lado, no encontraron evidencia de una mejora en la productividad. Sin embargo, hubo casos en que cuando los estudiantes tuvieron el proyecto bajo control, se detectó alguna baja en el esfuerzo.

En algunos artículos se analiza la opinión de los clientes para evaluar aspectos de aprendizaje. En un estudio los clientes mencionan que uno de los aspectos más positivos es el aprendizaje de los estudiantes y cuánto mejoraron los equipos durante el proyecto (Paasivaara et al. 2019). Esta mejora se detectó en la comunicación y en la proactividad.

Burden et al. (2019) presentan el análisis comparativo de cómo se relaciona cada momento de enseñanza del curso con las áreas de competencias para el emprendurismo, y reportan las percepciones y experiencias de los estudiantes teniendo en cuenta las mismas áreas. Concluyen que el curso colabora con el desarrollo de habilidades relacionadas con la gestión y adquisición de recursos, y las habilidades necesarias para pasar de la ideación (donde se genera la idea del producto, servicio, o lo que se vaya a desarrollar) a poner en acción lo que se ideó. El desarrollo de competencias del bloque ideas y oportunidades se ve limitado debido a que cuentan con un dueño de producto externo el cual define

el alcance y también porque los estudiantes tienden a ser más conservadores. Sin embargo, les proporciona a los estudiantes un marco para que puedan trabajar en la ideación ya que Scrum no lo tiene. Reportan que la mayoría de los estudiantes desarrollaron habilidades de trabajo en equipo, tuvieron problemas con el balance entre los desafíos técnicos y sus ideas respecto a lo que el cliente quiere, se sintieron motivados por crear valor para otros —principalmente trabajar ante necesidades reales de un actor externo— y entienden (consultaron a los estudiantes mediante encuestas, informes de reflexión de los equipos y formularios de evaluación individuales) que aplicaron un proceso que les funcionó.

Tafliovich et al. (2019) reportan como objetivos de aprendizaje varias de las áreas de conocimiento típicas de la ingeniería de software, la escritura técnica, trabajo en equipo y habilidades blandas, y práctica profesional. Los autores creen que la experiencia de contar con un curso que cumple con las características necesarias para ser *capstone* colaborando en un proyecto de software libre de código abierto real, ayuda a que los estudiantes desarrollen habilidades de comunicación, y fomenta en estos un gran sentido de pertenencia y responsabilidad por el producto de trabajo. Sin embargo, en el artículo no se presenta evidencia ni resultados que apoyen lo expresado.

A diferencia de otros, en el curso experimental presentado por Hsu et al. (2019) el objetivo de aprendizaje está centrado exclusivamente en el proceso; se espera que los estudiantes aprendan y experimenten con metodologías ágiles, y que comprendan las diferencias entre estos y el desarrollo tradicional. Si bien los autores no presentan resultados que permitan inferir que se cumple con dicho objetivo de aprendizaje, opinan que todos los equipos cambian su forma de trabajo en los proyectos después de la reunión de retrospectiva y obtienen un mejor resultado en el segundo *sprint*.

Germain et al. (2002) entienden (si bien no se presentan resultados que soporten esta conclusión) que los estudiantes luego de realizar el curso, desarrollan habilidades que pueden ser aplicadas en proyectos de la vida profesional, adquieren nuevos conocimientos, y son bien preparados para el aprendizaje independiente a lo largo de toda la vida.

Bastarrica et al. (2019) exploran el impacto de cambios a un esquema de evaluación que considera diversas perspectivas —gestión de proyectos, calidad del producto, presentación, valor de la solución y evaluación de pares— por el cual se brinda una calificación a cada estudiante. Esta calificación se da al fina-

lizar cada iteración, es realizada por el docente y los estudiantes. El cambio se realizó para darle mayor peso a la evaluación de pares. Los autores concluyen que este cambio hace que los estudiantes sean más responsables de las calificaciones de sus compañeros de equipo cuando todavía hay tiempo para mejorar. Comparando con las evaluaciones previas al cambio, se encuentra que en los semestres anteriores no hubo casi cambios en las calificaciones durante las tres iteraciones, siendo siempre calificaciones altas y no presentándose variaciones entre los estudiantes. Por lo cual, concluyen que también tuvo un impacto positivo en el trabajo en equipo respecto al modelo de evaluación anterior. Como conclusión final, se plantea que el hecho de empoderar al estudiante al momento de calificar a otros, hace que lo tome como una oportunidad para dar y recibir retroalimentación para mejorar el trabajo en equipo.

En la revisión que realizamos notamos que los autores han recurrido a diversos métodos de evaluación para estudiar el impacto del curso en el aprendizaje de los alumnos como ser evaluación de pares, entrevistas con clientes o los estudiantes, datos de los proyectos, y en varios de los casos solo presentan sus opiniones sin la presentación de un análisis. En algunos estudios se presenta la recolección de los datos pero, en nuestra opinión, no es suficiente para concluir las afirmaciones que realizan. Entendemos que la evaluación del aprendizaje en cursos de estas características puede resultar muy compleja debido al diseño del curso, ya que no sigue un modelo tradicional donde el estudiante de forma individual debe rendir exclusivamente una prueba específica. En la revisión realizada por Dutson et al. (1997) se reporta que este es uno de los principales desafíos de los cursos *capstone* en cualquier área, al igual que otros casos donde se plantea como uno de los aspectos a resolver en el curso desde la óptica de asignar una nota de forma individual o de forma grupal (ver, por ejemplo, (Moore y Potts, 1994)). Entendemos que este desafío sigue aún totalmente vigente.

3.4. Conclusiones de la revisión

En la revisión de la literatura no encontramos procesos con un detalle de especifición suficiente: estaban descriptos de forma genérica o aplicaban marcos y procesos conocidos sin detallar acerca de su adaptación para el curso, y hacían foco en algún aspecto en particular del proceso. Esta dificultad es planteada también en el mapeo sistemático revisado en nuestro trabajo (Heredia et al.

2015). Tampoco presentaban consideraciones para la creación de un proceso de desarrollo para el curso.

De todas formas, para nuestro trabajo se tomaron algunas ideas de los procesos y de los cursos, y los resultados obtenidos en los estudios. Estas ideas las aplicamos en la creación de PISAgile y en su evaluación.

Los principales resultados aplicados para el diseño de PISAgile fueron algunos vinculados a los roles de los actores: la importancia de contar con un *coach*, que el cliente real cumpla el rol de dueño de producto (con una participación activa), y que el rol de *scrum master* no sea desempeñado por el docente debido a que no es escalable durante le ejecución del proyecto por la alta participación que debe tener. También incluimos consideraciones vinculadas al comportamiento a promover en el proceso: la importancia de entregar valor a los interesados sobre la construcción de un producto y la colaboración entre los involucrados. Incluímos detalles respecto a las etapas del proceso, como ser la configuración inicial del proyecto y la ideación, que Scrum no tiene. Otro resultado que nos resultó de utilidad, fue la identificación de las habilidades que fueron más desafiantes para los estudiantes y así dotar a PISAgile de elementos que permitan apuntalarlas, como ser aspectos vinculados al manejo de la calidad y la adaptabilidad. Si bien algunos de estos aspectos los teníamos considerados previamente, la revisión sirvió para ajustar su aplicación.

Otra idea que aplicamos para acotar el alcance de la tesis, fue enfocar la investigación hacia un aspecto particular del aprendizaje como lo hacen los estudios revisados: las habilidades blandas. Para confeccionar nuestro conjunto de habilidades blandas a desarrollar incluimos las contempladas en los estudios, y a su vez comparamos los resultados obtenidos en ellos con nuestros resultados.

Sumado a lo anterior, encontramos que varias propuestas de los estudios ya las teníamos en cuenta por haber sido parte de algunos procesos ad hoc del PIS (por ejemplo, tener retrospectivas), y varias de las conclusiones a las que llegan tienen coincidencias con el diseño del PIS (por ejemplo, tener espacios de monitoreo). A su vez, hubieron algunos aspectos que si bien evaluamos su inclusión, no fueron incluidos por requerir cambios mayores y no era el foco de este trabajo. Estos fueron: la posibilidad de tener estudiantes de grado junto con estudiantes de posgrado en distintos roles, contar con estudiantes de grado con distintos niveles de avance con métodos de enseñanza adaptados a cada caso, e incluir en el curso evaluación de pares.

Capítulo 4

Curso Proyecto de Ingeniería de Software

En este capítulo se presenta la evolución de la enseñanza de la IS en nuestra universidad y se describe nuestro curso —Proyecto de Ingeniería de Software—. Finalmente, se caracteriza la experiencia realizada en el año 2020, primer año en que se experimentó con PISAgile.

Dentro de la currícula sugerida en el plan de estudios actual de la carrera Ingeniería en Computación de la Facultad de Ingeniería de la Universidad de la República de Uruguay[1], existen dos cursos obligatorios que abordan la ingeniería de software desde una enfoque integral: Introducción a la Ingeniería de Software (IIS) y Proyecto de Ingeniería de Software (PIS). Ambos cursos son tomados por los estudiantes en cuarto año —penúltimo año de la carrera—, en el séptimo y octavo semestre respectivamente.

El objetivo del curso IIS es brindar un panorama general de la ingeniería de software profundizando en algunos aspectos de la disciplina, e introducir algunas técnicas y herramientas necesarias para que el estudiante pueda participar en distintos roles en un proyecto de ingeniería de software. El PIS, que es un curso *captstone* basado en proyectos, tiene por objetivo afirmar y profundizar los conocimientos en ingeniería de software.

[1] https://www.fing.edu.uy/carrera/grado/ingeniería-en-computación

4.1. Breve Historia del PIS

La primera vez que se utilizó un curso *capstone* basado en proyectos en nuestra facultad fue en 1999. En dicho año se les planteó a los estudiantes trabajar en proyectos con un plan de trabajo iterativo e incremental en grupos de doce estudiantes. Los proyectos tenían una duración fija, roles y entregas preestablecidas. Esta fue una versión experimental de lo que luego sería el PIS.

Al año siguiente, se crearon y aplicaron dos modelos de procesos orientados al desarrollo en Java específicamente para el curso. En el año 2002 se tuvo un modelo de procesos nuevo para desarrollo con GeneXus. Paralelamente, los docentes evolucionaron los procesos creados en el año 2000. En el curso se fueron aplicando distintos modelos de procesos, en forma paralela evolucionando los ya creados y creando nuevos.

Fueron varios años de evolución de procesos para contar con uno establecido que permanece desde el 2005 hasta la fecha, el cual es un modelo de proceso dirigido por planes: iterativo e incremental, el "Modularizado Unificado y Medible (MUM)". Es un modelo base que tiene dos extensiones, una enfocada a lenguajes orientados a objetos y otra a Genexus.

Para facilitar el mantenimiento del proceso, en el año 2009 el MUM se documentó y publicó a través de la herramienta Eclipse Process Framework (EPF). La última actualización fue en el año 2012 donde se generó la versión 2.0 del MUM. Versión que se sigue usando actualmente en algunos grupos del PIS.

4.2. Caracterización del PIS

El PIS es un curso de pregrado, *capstone* y basado en proyectos reales (i.e. la industria de desarrollo de software independientemente del ámbito al que pertenezca —industria, academia o estado—).

Antes de cursar el PIS, los estudiantes tienen otros cursos de computación con una amplia cobertura en diferentes áreas temáticas; programación, arquitectura de sistemas, sistemas operativos y redes, ingeniería de software, entre otras. Si bien son estudiantes avanzados que pasaron por cursos técnicos en los que aprenden métodos y técnicas asociados al desarrollo de software, puede que no tengan experiencia de trabajo en proyectos medianos y grandes que son llevados por equipos de más de 10 personas.

El objetivo general del PIS es que los estudiantes afiancen y profundicen los conocimientos de ingeniería de software, contrastarlos con su aplicación práctica e integrarlos con conocimientos de las áreas mencionadas. Se busca que los estudiantes reflexionen sobre los conocimientos teóricos de la ingeniería de software a la luz de su aplicación práctica. Fomenta que los estudiantes aprendan del trabajo en equipo, la responsabilidad, el compromiso individual y colectivo, la evaluación de pares y la autoevaluación individual y grupal. En el programa del curso (InCo, 2017) no se describen de forma exhaustiva las habilidades blandas que son necesarias desarrollar para trabajar en la industria del software actual y específicamente en equipos que apliquen metodologías ágiles.

El curso dura dieciséis semanas y otorga quince créditos[1]. Cada estudiante debe dedicar al curso alrededor de 15 horas semanales.

El diseño pedagógico del curso contempla el aprendizaje: colaborativo por parte de los estudiantes; por medio de la guía y la reflexión en encuentros semanales con un docente y acorde al proceso aplicado; y la aplicación de los conocimientos adquiridos. Los principales actores del curso son los grupos de estudiantes, los docentes y el cliente.

Cada docente guía al menos a un grupo de estudiantes durante el proyecto. El cliente es un individuo u organización que presenta un proyecto, este puede pertenecer a la industria, la academia o a un organismo estatal. Los grupos de estudiantes tienen entre diez y quince integrantes que deben ejecutar un proyecto aplicando un proceso de desarrollo de software. Dado que en general se cuenta anualmente con 140 inscriptos se forman aproximadamente diez grupos.

Los clientes presentan los proyectos previo al comienzo del curso. Estos son evaluados y seleccionados por los docentes buscando que sean adecuados para el curso. Los grupos son propuestos por los propios estudiantes. Los docentes asignan a cada grupo el proyecto a ejecutar.

Se espera que el grupo de estudiantes genere un producto que satisfaga las necesidades y restricciones del cliente, y que tenga un nivel adecuado de calidad. Los procesos que se aplican en el curso son todos iterativos e incrementales. Estos son: el MUM que es dirigido por planes; procesos basados en Scrum definidos de forma ad hoc individualmente por cada docente o este con los estudiantes, o proporcionados por el cliente; y PISAgile, el proceso creado

[1]Crédito: unidad de medida del avance y finalización de la carrera, donde un crédito equivale a 15 horas de trabajo estudiantil.

en el marco de esta tesis.

De las dieciséis semanas del curso, se utilizan dos al comienzo para inducir a los estudiantes al curso, presentar los procesos, armar grupos y asignar proyectos. Las restantes catorce semanas son dedicadas al proyecto. Los estudiantes tienen clases de apoyo para algunas áreas de conocimiento, encuentros semanales con el docente asignado y consultas al mismo docente durante el transcurso de la semana, generalmente vía correo electrónico. No hay pautas preestablecidas respecto a los espacios de trabajo a utilizar por los estudiantes.

Hay disponibles herramientas de soporte: está publicada en la web la memoria organizacional de los proyectos ejecutados[1], también documentación y plantillas del MUM, mientras que PISAgile se encuentra especificado en archivos de texto y planillas, y una planilla para el registro de esfuerzo. El curso no tiene definidas herramientas de gestión ni de soporte para el desarrollo, estas son seleccionadas por los estudiantes.

La evaluación es continua durante el curso considerando los encuentros y entregas semanales. La calificación final es a nivel grupal e individual. Se considera el producto construido, proceso y metodologías utilizadas, y la presentación final del trabajo. Estos aspectos contemplan la visión de los distintos actores —docente del grupo, tribunal evaluador, cliente e integrantes del grupo—.

4.3. Experiencia 2020

Consideramos que una de las mejores formas de evaluar si un proceso resulta adecuado para sus fines, es mediante su uso en el entorno para el cual fue creado. Estar en un contexto académico de enseñanza, tiene el beneficio de poder contar con un espacio controlado en el cual realizar estas evaluaciones. Pero se debe tener especial cuidado en que ello no interfiera con el cumplimiento de los objetivos de aprendizaje del curso en el cual se está aplicando, si bien el proceso fue diseñado para cumplir con estos, con la experimentación queremos comprobar que así lo sea.

Durante el año 2020 —primera experiencia de PISAgile— se inscribieron 117 estudiantes los cuales se distribuyeron en diez grupos de doce estudiantes cada uno aproximadamente, participaron nueve docentes y diez clientes. Cada

[1] https://www.fing.edu.uy/inco/cursos/ingsoft/pis/memoria/index.htm

proyecto tuvo asignado un único grupo de estudiantes. Respecto a los procesos, tres grupos aplicaron MUM, cuatro procesos ágiles ad hoc, y tres aplicaron PISAgile. Cada proyecto que utilizó PISAgile tuvo el acompañamiento de un docente distinto —cumpliendo un rol de *coach* de grupo—; siendo uno de ellos la autora de esta tesis. Todos los docentes que participaron como *coach* de PISAgile cuentan con experiencia previa en la aplicación de procesos Scrum ad hoc en ediciones anteriores del curso.

De los grupos que aplicaron PISAgile, uno fue conformado por diez estudiantes y dos por doce estudiantes. Los tres clientes son empresas de desarrollo de software que aplican metodologías ágiles en sus proyectos. Dos de dichos clientes fue la primera vez que participaron en el curso, el otro participó en más de una oportunidad. Estos proyectos se seleccionaron de la forma habitual, teniendo en cuenta que el objetivo principal del proyecto sea desarrollar un software y que se puedan aplicar metodologías ágiles.

Para la primera experiencia de PISAgile se brindó a los actores del curso (estudiantes, docentes y clientes) un conjunto de materiales para que puedan aplicar el proceso, y se realizaron actividades de apoyo y seguimiento a su aplicación. Se entregaron dos tipos de materiales, unos que corresponden a la especificación del proceso y otros de soporte para su aplicación. La entrega del material se realizó de forma paulatina a medida que avanzaba el proyecto. La especificación del proceso se compone de una serie de documentos: la descripción general de componentes —describe el proceso desde su arquitectura, componentes que lo integran y sus relaciones—, la especificación general del proceso —se describen las disciplinas, roles y fases—, un documento de especificación para cada una de las fases —contiene una descripción detallada de la fase, objetivos y actividades—, y un resumen de los artefactos y entregas semanales que hay que realizar en cada una de las fases. Es importante aclarar que debido a que la especificación fue generada de forma paralela al curso, los actores no tuvieron disponible toda la documentación al comienzo, sino que se fue liberando a medida que era necesaria. El material de soporte a la aplicación se entregó únicamente al *coach* y se compone del diario del *coach* y la planilla diario del *coach* —que sirven de guía a su trabajo y a su vez para poder hacer seguimiento al trabajo del grupo—, y una planilla de mapeo de actividades, roles y disciplinas. En referencia a las actividades de apoyo, los estudiantes tuvieron las clases de apoyo del curso y el acompañamiento de su *coach* durante el proyecto. Este último también brindó soporte directo a los clientes por dudas

que pudieran surgir acerca del proceso. La autora de la tesis —creadora del proceso— cumplió también un rol de responsable del proceso y brindó apoyo directo a los *coach* dando una presentación general del proceso al comienzo del curso y espacios de intercambio —formales e informales— durante el semestre para guiarlos en la aplicación, resolver dudas, e intercambiar opiniones del proceso e informarse de cómo lo estaban aplicando los estudiantes.

Si bien en 2021 se continuó con el uso de PISAgile, gran parte de esta tesis fue escrita en paralelo con la nueva ejecución de PISAgile en 2021 y por ende solamente se presentan los datos de su uso en 2020.

4.3.1. Desafíos y variables

La edición 2020 del PIS contó con un contexto inusual. A raíz de la pandemia COVID-19, la duración del curso pasó de dieciséis a quince semanas, los proyectos de catorce a trece, y el curso pasó a modalidad virtual —esto incluye clases de apoyo, reuniones semanales con el docente, instancia final de evaluación y reuniones con su cliente—. Se aconsejó que las reuniones entre los estudiantes fueran virtuales. Los cambios mencionados son variables que se adicionan al curso junto con la aplicación del nuevo proceso. Sumado a ello, no estaba disponible el proceso a aplicar de forma completa a principios del semestre, mientras que los otros procesos que se aplican en el curso ya estaban acordados y eran conocidos desde un comienzo. A su vez, no se pudieron realizar adaptaciones en las clases de apoyo que consideraran las particularidades de PISAgile.

Capítulo 5

Descripción del proceso PISAgile

PISAgile surge para dar respuesta a la necesidad de tener en el curso un proceso de desarrollo ágil especificado y documentado. Para de esta forma dar respuesta, desde la academia, a las necesidades y restricciones de la industria de contar con profesionales que sean capaces de aplicar procesos de desarrollo que permitan crear software eficiente, de calidad y oportuno (considerando tiempos cortos de salida al mercado). Esto es reforzado por la solicitud de los propios clientes del PIS que presentan los proyectos en el curso.

En este capítulo se presenta el proceso de desarrollo PISAgile, el mismo se creó para ser aplicado en el contexto del curso de grado Proyecto de Ingeniería de Software (PIS). El objetivo es proporcionar una visión general en amplitud del proceso, resaltando sus principales particularidades sin brindar una especificación detallada. La especificación completa de PISAgile se encuentra publicada en el repositorio Zenodo[1].

En primer lugar se proporciona una descripción general del proceso y de su arquitectura. De esta última, se definen los componentes que la integran y sus relaciones, en particular los valores, principios y las dimensiones del proceso, y se especifica a nivel general cómo estos son instanciados en el proceso. Luego se detallan los principales elementos del proceso: disciplinas (líneas de trabajo), roles, artefactos y fases acompañados de una vista global acerca de cómo es concebida la planificación en el proceso, referencia a las métricas propuestas y pautas sobre la estimación en el proceso. Posteriormente se presentan a nivel

[1] https://doi.org/10.5281/zenodo.6730363

general todas las actividades de las fases del proceso. Para finalizar, se resumen los principales aspectos que se incluyeron de otros procesos en PISAgile.

Para la construcción de PISAgile, inicialmente nos basamos en el resumen de procesos y *frameworks* presentado en el Handbook of Software Engineering (Cha et al. 2019). Dado que nuestro interés era explorar procesos aplicados en la academia, decidimos realizar una revisión sistemática de la literatura. Dada la dificultad de encontrar estudios que reportaran detalles de los procesos de desarrollo de software aplicado y consideraciones para su creación, aplicamos otras alternativas. Resolvimos ir complementando con el estudio y evaluación de características generales, prácticas y experiencias de procesos considerando otras fuentes. Estas son: conocimiento adquirido por la autora de esta tesis en cursos realizados en el marco de la maestría en ingeniería de software[1], charlas con especialistas, una revisión de la memoria organizacional del curso, los proceso ágiles ad hoc que se venían aplicando, las experiencias de docentes con la que cuenta el PIS y la especificación del proceso disciplinado vigente en el curso. Todo ello lo profundizamos con literatura de referencia para cada uno de los procesos.

Para la creación de PISAgile optamos por seguir las recomendaciones presentadas en Hilburn y Humphrey (2002) para la creación de procesos en cursos basados en proyectos. Estas pautas las complementamos con distintos elementos que surgieron de la revisión de la literatura y de la evaluación de otros procesos ya definidos. Si se quiere ver un mayor detalle acerca de los aspectos incluidos de la revisión se puede consultar la sección 3.4 y para los incluidos de los otros procesos se puede ver la sección 5.4.

La creación y documentación del proceso la hicimos de forma paulatina e incremental conforme avanzaba su aplicación en la edición 2020 del PIS. A medida que se liberaban documentos, estos pasaban por un proceso de revisión y validación del cual participaban distintos perfiles de personas (docentes del curso, tanto actuales como de años anteriores, especialistas de la industria y la academia en procesos de desarrollo de software, ex-estudiantes del PIS y personas no vinculadas al desarrollo de software que ayudaron a evaluar la legibilidad del documento). La validación final fue realizada por uno de los docentes responsables del curso y tutor de esta tesis.

[1]Maestría en ingeniería de software de la Facultad de ingeniería, Universidad de la República, Uruguay. Maestría en la cual se enmarca este tesis.

5.1. Descripción general y arquitectura

El proceso PISAgile es un proceso ágil. Para su construcción se tuvieron en cuenta aspectos de distintos procesos y marcos de trabajo, y se complementaron con decisiones propias. Se seleccionaron aspectos de Scrum, del Proceso Modularizado Unificado y Medible (MUM), Agile unified process (AUP), eXtreme Programing (XP), el Team Software Process (TSP), Design Thinking, Kanban, y prácticas generales de la ingeniería de software. En la sección 5.4 se presenta la relación entre PISAgile y estos procesos.

PISAgile busca incorporar las mejores prácticas de la ingeniería de software vinculadas al desarrollo evolutivo; promueve experimentar de forma temprana para agregar valor al cliente y obtener retroalimentación lo antes posible para realizar las adaptaciones necesarias. Es un proceso para el desarrollo de software que realiza un abordaje que abarca desde la identificación de la necesidad del cliente hasta la transferencia de los productos construidos y los resultados obtenidos.

El proceso tiene dos dimensiones (dinámica y estática) en las cuales se clasifican sus componentes y subcomponentes. Ambas dimensiones se encuentran definidas sobre los cimientos del proceso: los valores y principios. La dimensión dinámica está definida por las fases e iteraciones. La dimensión estática se compone de actividades, roles, productos y artefactos, recursos de soporte, y de forma transversal a estos por las disciplinas (o líneas de trabajo) y las métricas e indicadores. En la figura 5.1 se puede visualizar la arquitectura de PISAgile.

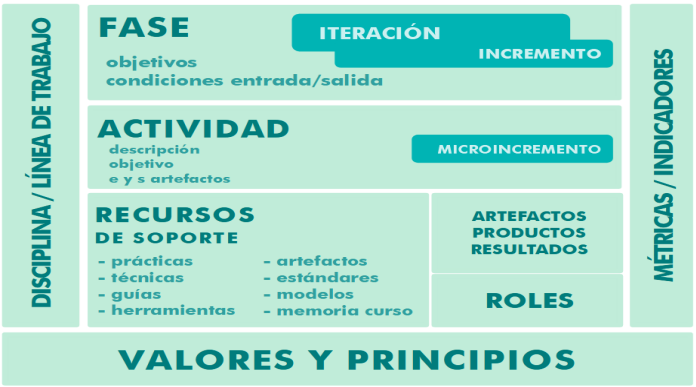

Figura 5.1: PISAgile: arquitectura

5.1.1. Valores y principios

Los valores son cualidades o atributos intangibles que deben adoptar y desarrollar los involucrados en el proceso PISAgile en forma individual y grupal. Los principios son reglas que guiaron la especificación del proceso y van a guiar la forma de trabajo al aplicar este proceso de software. Estos valores y principios involucran al equipo de desarrollo (grupo de estudiantes), al cliente, al *coach* (docente del curso) y a otros interesados (del inglés, *stakeholders*). En PISAgile los involucrados son aquellos interesados que tienen una participación más activa en el proceso (cliente, equipo de desarrollo y *coach*).

En el proceso se incluyeron cinco valores que son descritos a continuación:

Transparencia.- debe existir transparencia en el proceso y del equipo. En el proceso se vincula a la disposición y conocimiento por parte de los involucrados de todo material e información manejada dentro del alcance del proyecto. En cuanto al equipo, los integrantes deben ser transparentes en su forma de trabajo y percepciones (o sentimientos) vinculados al proyecto y al equipo.

Respeto.- debe existir el respeto mutuo entre los integrantes del equipo, el cliente y los otros interesados del proyecto.

Comunicación continua y abierta.- debe existir un ambiente propicio para que todos los involucrados (cliente, equipo de desarrollo y *coach*) sientan que pueden comunicarse de forma abierta con el resto.

Compromiso.- debe existir compromiso por parte de todos en hacer lo mejor posible en cada momento.

Foco.- al realizar cada actividad se debe estar específicamente concentrado en la ejecución de la misma pero sin perder objetivos a largo plazo.

En el proceso se incluyeron doce principios que se describen a continuación:

Equilibrio.- el proceso y su ejecución deben mantenerse simples de comprender pero manteniendo la rigurosidad en su aplicación, y debe mantenerse un equilibrio en el producto entre cubrir las necesidades del cliente y el uso de tecnologías.

Adaptabilidad.- el proceso debe permitir ser adaptado acorde a características del contexto, pero siempre se debe conservar su esencia.

Calidad.- se debe hacer foco en todo momento en la mejora de la calidad del proceso y del producto.

Medición y registro.- se deben generar métricas, indicadores y registros asociados a las actividades, artefactos y productos.

Evaluación.- durante la ejecución del proceso deben realizarse evaluaciones cuantitativas y cualitativas que permitan observar la brecha entre lo real y esperado, y colaboren en la autogestión del equipo.

Revisión.- el proceso, los artefactos y productos deben ser siempre revisados con foco en poder mejorarlos acorde a las expectativas sobre estos.

Reflexión.- promover la reflexión grupal e individual continuamente sobre la forma de trabajo y las oportunidades de mejora.

Retroalimentación.- se deben tener en cuenta los resultados, reflexiones y evaluaciones que se obtienen en cada ejecución para adaptar el proceso, proyecto y artefactos, y mejorar en la siguiente.

Mejora continua.- se debe apuntar de forma continua a planificar, ejecutar, evaluar y mejorar el proceso y todos sus elementos.

Proyección y entrega de valor.- se deben generar incrementos -en cada iteración- que entreguen valor al cliente sin perder de vista su impacto en los objetivos a largo plazo.

Seguimiento.- se debe realizar un seguimiento periódico de la evolución del producto: a corto plazo (semanal y por iteración) y a largo plazo considerando los resultados de forma acumulada.

Colaboración de equipo.- en todo momento debe estar presente la colaboración entre los involucrados y el espíritu de equipo con el fin de generar sinergia y obtener mejores resultados.

5.1.2. Dimensiones dinámica y estática

La **dimensión dinámica** del proceso define el orden temporal. Se compone de fases e iteraciones. Cada fase tiene objetivos, criterios de fin de fase y una duración sugerida (en semanas). Las fases se componen de iteraciones, cada iteración contribuye a la creación o evolución de incrementos que componen uno o más artefactos, productos o resultados del proyecto. Esta creación o evolución está guiada por un conjunto de actividades asociadas a las disciplinas (o líneas de trabajo) que deben ser ejecutadas para lograrla.

PISAgile consta de tres fases: fase inicial, fase de construcción y fase de transición. La fase inicial tiene como objetivos principales que los estudiantes tengan un primer acercamiento al proyecto y proceso, comiencen a conformarse como equipo, entiendan las necesidades del cliente y logren hacer una conceptualización básica del producto a construir. La fase es de una única iteración

de tres semanas.

La fase de construcción tiene como objetivos principales la construcción y la aceptación del producto de software. Esta fase dura 10 semanas, dividida en 5 iteraciones de 2 semanas cada una.

La fase de transición, tiene como objetivos hacer la transferencia final de los productos de software construidos al cliente, evaluar el resultado obtenido, y se reflexiona sobre el proceso y proyecto. Esta fase es de una única iteración de una semana.

En la vista general del proceso, figura 5.2, se pueden visualizar las fases y su orden. Una descripción más detallada de las fases se presenta en la sección 5.2.4, y la especificación completa se encuentra publicada en el repositorio Zenodo[1].

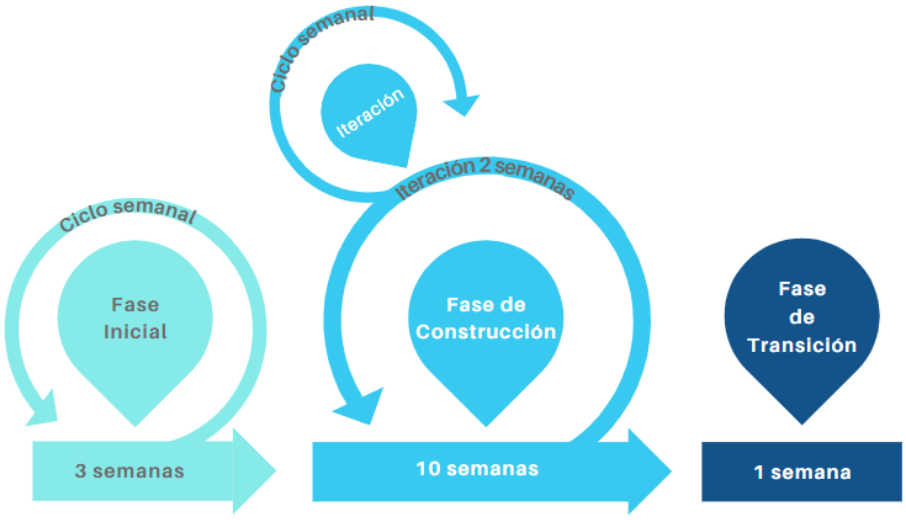

Figura 5.2: PISAgile: vista general

En la **dimensión estática** del proceso se describen las disciplinas (o líneas de trabajo), las actividades, recursos de soporte, los roles, los productos o artefactos, y las métricas e indicadores. También las relaciones entre estos componentes.

Las disciplinas o líneas de trabajo son una forma de clasificar y agrupar en temáticas el desarrollo del conocimiento. En el proceso se clasifican en dos tipos: disciplinas básicas y disciplinas de soporte. El desarrollo del conocimiento en las disciplinas se realiza mediante la ejecución de un conjunto de actividades que son llevadas a cabo por determinados roles de forma alineada a los

[1] https://doi.org/10.5281/zenodo.6730363

principios y valores definidos. Si bien las disciplinas tienen un alcance definido, hay puntos de intersección entre algunas de ellas. En la figura 5.3 se presentan las disciplinas de PISAgile. Las disciplinas básicas corresponden a las disciplinas de ingeniería de software e ingeniería que participan de forma directa en la línea principal de construcción del software y son parte esencial en su ciclo de vida. Estas son: requisitos, diseño/arquitectura, diseño/interacción de usuario, implementación, calidad del producto y verificación, e implantación. Las disciplinas de soporte son aquellas que están más enfocadas en la gestión y desarrollo de habilidades de distintos aspectos del proyecto y que colaboran con la aplicación de las disciplinas básicas, el desempeño del proceso, y el desarrollo de los principios y valores en el equipo. Estas son: gestión de proyecto, colaboración, gestión del conocimiento, formación y entrenamiento, calidad del proceso y gestión de la configuración.

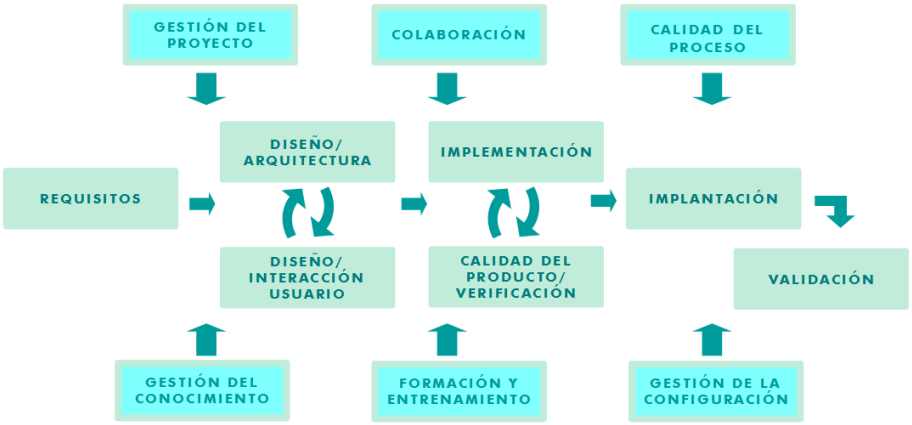

Figura 5.3: PISAgile: disciplinas (líneas de trabajo).- la disciplina validación se muestra aparte solo para efectos prácticos pero es parte de calidad del producto y verificación

Las actividades están asociadas al menos a una disciplina, cuentan con un nombre, descripción y objetivo, y precondiciones para su ejecución. Se establece en su especificación el vínculo con otros componentes del proceso: la entrada de artefactos que son necesarios consumir para el desarrollo de la actividad, y salida de los artefactos, resultados o productos generados durante la ejecución

de la actividad. También se define el conjunto de roles participantes en su ejecución. Para contribuir con el rendimiento y desempeño de la actividad (del inglés, *performance*), en algunas actividades se proponen recursos de soporte. Las actividades establecen un vínculo con la dimensión dinámica mediante la especificación de su duración, frecuencia de ejecución y ubicación temporal (fase, iteración, semana). Es importante notar que debido al diseño del proceso hay actividades recurrentes[1], repetitivas[2] o de ejecución única dentro de la ubicación temporal. Esto hace que el estado de evolución de los artefactos de entrada y salida pueda variar, incluyendo algunos casos en que un artefacto puede no estar creado.

Los artefactos son elementos tangibles que pueden estar compuestos por otros artefactos. Estos son consumidos, generados y modificados mediante distintas actividades o en la misma actividad siendo ejecutada en instancias, fases y/o iteraciones distintas. Los participantes usan los artefactos para ejecutar actividades y producir artefactos durante la ejecución de estas. Un ejemplo de artefacto es un documento que tenga un listado preliminar de los riesgos del proyecto.

Los artefactos entregables (productos) son un tipo particular de artefactos que tienen valor por sí mismos para algún interesado, o sea su construcción ya denota cierto avance acorde a las expectativas de este. Estos son entregados para el trabajo interno del equipo o de forma externa para otros interesados. Un ejemplo es el software construido o un documento que especifique la arquitectura.

Los resultados son elementos intangibles que serán entregados para el trabajo interno del equipo o de forma externa al cliente. Los resultados son salidas generadas en las actividades que tienen valor por sí mismos para algún interesado y no son candidatos a ser reutilizados. Un ejemplo de resultado puede ser cumplir con el objetivo de que el software construido sea descargado por 1.000 usuarios.

Al combinar los artefactos asociados a las actividades, con las instancias de ejecución, fases o iteraciones surgen los conceptos de incrementos y microincrementos. Los incrementos de la iteración[3] son los artefactos construidos hasta el momento en la iteración más las nuevas porciones de artefactos (construidas

[1]Ejecución que se realiza con una frecuencia especificada en el proceso.

[2]Ejecución reiterativa sin una frecuencia especificada en el proceso.

[3]En los casos que se quiera restringir a la construcción de un producto de software, será incremento de software o sistema.

durante la iteración, que se agregarán a los ya construidos) más lo que ya estaba construido en iteraciones anteriores. Los microincrementos son pequeñas porciones que componen los incrementos, y se agregan cada vez que se ejecuta una actividad. Con el objetivo de simplificar, en algunas ocasiones se va referir a la unión de todos los incrementos de los distintos artefactos o resultados para una iteración como el incremento de la iteración.

Los recursos de soporte incluyen: prácticas, técnicas, guías, herramientas, estándares y memoria del curso. Dado que el universo de estos elementos es muy amplio, en las actividades se especifican solo algunos recursos para tomar como referencia, pero deben ser complementados por el equipo con la guía del docente. A continuación se describe el alcance de distintos tipos de recursos en el contexto de PISAgile:

- prácticas: expresan de forma general cómo se debe realizar determinada acción, el comportamiento esperado, qué herramientas y artefactos se pueden utilizar para que las actividades se ejecuten de forma alineada a los valores y principios. A través de las prácticas se materializan en el proceso los valores y principios. Estas incluyen: dinámicas de grupos, pautas generales, especificación de uso de artefactos, etc.
- técnicas: indican un método específico para ejecutar una actividad.
- guías: presentan una serie de pasos concretos a seguir para lograr un objetivo específico.
- herramientas y elementos: se engloban los instrumentos necesarios para el desempeño de una actividad o práctica. Se incluyen: herramientas informáticas (entornos de desarrollo integrados (del inglés, Integrated Development Environment (IDE)), repositorios de código, herramientas de gestión de proyectos, herramientas colaborativas, etc.) y elementos que sirven para llevar a cabo prácticas determinadas (templates, cartas para estimación, etc.).
- estándares, normas y modelos: presentan convenciones definidas para abordar determinados aspectos vinculados al desarrollo del software y la estructura de algunos productos. Estas convenciones pueden ser definidas por organizaciones nacionales o internacionales o por docentes o estudiantes del curso.
- memoria del curso: es el historial de los proyectos que se han realizado en el curso. Allí se puede encontrar información de cada uno de ellos

para gestionar y transversalizar el conocimiento entre proyectos. Dado que PISAgile es un proceso de reciente aplicación en el curso, la memoria aún no contiene información de proyectos que lo aplicaron pero sí tiene de otros procesos.

El proceso PISAgile define diferentes roles entre los interesados. Cada rol tiene un conjunto de responsabilidades y funciones asociadas que se encuentran vinculadas a las disciplinas y actividades del proceso. Un mismo rol puede estar asociado a más de una disciplina y participar en varias actividades. Dadas las características del proceso, los roles pueden tener funciones asociadas a un seguimiento general de las disciplinas o funciones vinculadas a desarrollar cierta profundización de conocimiento acerca de la aplicación de determinada disciplina en el proyecto. El seguimiento apunta a tener claro cómo se va desarrollando la disciplina en el proyecto, mientras que la profundización de conocimiento implica que se tiene un mayor conocimiento y experiencia para poder ejecutar alguna actividad en el proyecto. Se definen once roles, donde siete de ellos deben ser cubiertos por el equipo de desarrollo. Los roles del equipo son: facilitador, desarrollador, tester, responsable de calidad, responsable de seguimiento, responsable de gestión de la configuración y responsable de diseño y construcción. El resto de los roles son: *coach* (docente del grupo de estudiantes), cliente, dueño del producto y usuario. Los roles llamados "responsable de..." tienen como responsabilidad asegurarse de que ciertas actividades (de ciertas disciplinas) son ejecutadas correctamente por parte del equipo.

Para mejorar y evolucionar un proceso es necesario contar con evidencia y parámetros que permitan evaluar su desempeño, para ello se definen métricas e indicadores. El desempeño del proceso se mide de forma holística considerando aspectos cualitativos y cuantitativos, y se evalúan aspectos vinculados a calidad del producto, del proceso y la adopción de los principios. Las métricas se diseñaron utilizando el enfoque *Goal Question Metric* (Basili, 1992) y se clasifican en métricas para medir el esfuerzo, el avance y la calidad. La especificación de cada métrica se compone de un nombre, la frecuencia con la que esta debe medirse, los registros o artefactos en los cuales debe estar la información necesaria para calcular la métrica, cómo debe calcularse y el análisis de la métrica. Este último incluye la descripción del objetivo de la medición y algunas recomendaciones respecto a su interpretación.

Las métricas e indicadores propuestos permiten medir a nivel de cada proyecto y a nivel general del curso. A nivel de cada proyecto para que el equipo

cuente con evidencia que le brinde soporte y mayor seguridad a la hora de tomar decisiones y adaptarse. Mientras que a nivel del curso es para evaluar y evolucionar el proceso a través de los años. Las métricas completas se pueden visualizar en el Anexo 1: *Métricas, indicadores y medidas.*

5.2. Especificación general del proceso

En esta sección se especifican de forma detallada los principales elementos del proceso: disciplinas de trabajo, roles y fases. También se presentan los niveles de planificación propuestos, las métricas para medir distintos aspectos del proyecto y pautas acerca de cómo realizar las estimaciones necesarias.

5.2.1. Disciplinas/líneas de trabajo

Las disciplinas, como ya se mencionó, se clasifican en dos tipos: disciplinas básicas y disciplinas de soporte. Las disciplinas básicas de PISAgile son: requisitos, diseño/arquitectura, diseño/interacción de usuario, implementación (construcción), calidad del producto y verificación (incluye validación) e implantación. En el caso de diseño/interacción de usuario no solo se incluye diseño de interfaz y de interacción sino que también se incluye el abordaje de la experiencia de usuario (UX).

En cuanto a las disciplinas de soporte la mayoría se corresponden con disciplinas habituales en procesos de desarrollo de software, a la gestión de proyectos, y al desarrollo del conocimiento y habilidades en los involucrados. Estas son: gestión de proyecto, gestión del conocimiento, formación y entrenamiento, calidad del proceso y gestión de la configuración. A estas disciplinas se agrega la colaboración que se definió específicamente para PISAgile. Su propósito es desarrollar todos los aspectos vinculados a fortalecer el trabajo en equipo, la comunicación e integración entre los distintos interesados del proyecto para que puedan realizar sus actividades de forma integrada y sistematizada. Aborda también la negociación y la resolución de conflictos. Es parte esencial para que la suma del trabajo de las partes logren los objetivos deseados de forma altamente satisfactoria y puedan superar obstáculos en conjunto y de forma enriquecedora. Su alcance comprende la promoción, y definición de métodos y herramientas que sean habilitadores de la colaboración y la generación de sinergia entre los integrantes del equipo, el cliente y otros interesados.

5.2.2. Roles

Los roles (ver figura 5.4) son entidades en las que se agrupan un conjunto de responsabilidades y funciones necesarias para aplicar PISAgile, que se encuentran vinculadas a las disciplinas y actividades del proceso. A continuación se describen los roles de PISAgile de forma resumida. La descripción completa se encuentra en el repositorio Zenodo[1].

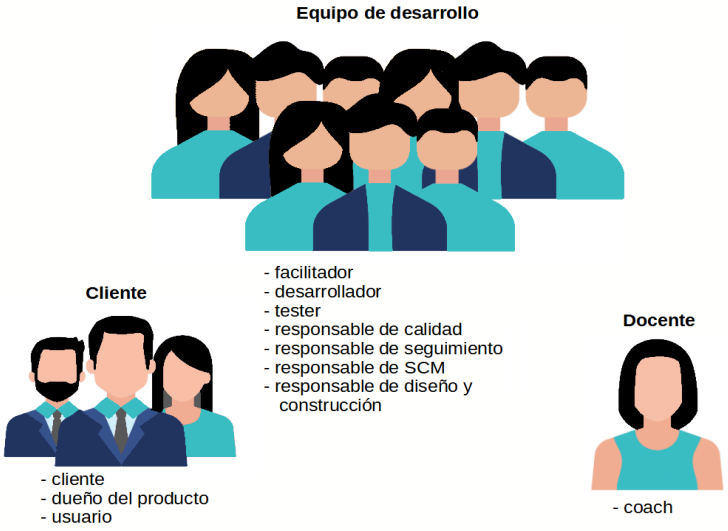

Figura 5.4: PISAgile: Roles

Coach.- Es el encargado de que los estudiantes que participan en el proyecto aprendan y adopten prácticas de trabajo en equipo, en un proyecto de desarrollo de software mediante la aplicación de un proceso de software. Debe motivar, guiar y colaborar para que cada rol de PISAgile se desarrolle según su participación en el proceso y las características del proyecto.

Cliente.- Son los representantes de la empresa u organización que presentó la propuesta a realizar. Estos pueden participar aportando acerca de lo que se espera del proyecto y su validación. Hay dos roles fundamentales que debe definir el cliente: el dueño del producto y usuario.

Dueño del producto.- Es el responsable de definir y expresar claramente en qué consiste el producto que se debe construir: cuáles son sus características y funcionalidades con el nivel de detalle necesario según cada etapa del proceso e indicando su prioridad. También tiene la responsabilidad de validar periódicamente que el equipo está interpretando de forma correcta sus definiciones y cumpliendo con lo que él espera del producto.

[1] https://doi.org/10.5281/zenodo.6730363

Usuario.- Es quien va a hacer uso del producto construido o quien va a simular serlo. Es una fuente muy importante para definir las necesidades y los requisitos asociados al producto, su principal función es describirlos, y colaborar con el dueño del producto y el equipo para que puedan comprenderlos y especificarlos.

Facilitador.- Es quien, en colaboración con el *coach*, ayuda al equipo y al dueño del producto a entender y aplicar el proceso. Es responsable por facilitar algunas de las actividades del proceso, particularmente las vinculadas a la revisión y reflexión, por promover la colaboración entre el equipo y el dueño del producto, por identificar impedimentos y promover su tratamiento, y por que el equipo desarrolle ciertas características de autogestión.

Desarrollador.- Es responsable por llevar a cabo la construcción del producto acorde a las expectativas del dueño del producto. Sus responsabilidades son: colaborar en la obtención, comprensión y especificación de lo que se espera del producto, definir tareas a realizar, estimar el esfuerzo de dichas tareas y registrar el esfuerzo real, realizar el diseño arquitectónico y de la interacción de usuario, implementar el código, realizar revisiones, diseñar e implementar las pruebas unitarias, e integrar los nuevos desarrollos a los ya construidos.

Tester.- Se encarga de realizar la verificación del producto y colabora en su validación acorde a las expectativas del dueño del producto. Es responsable por: asegurarse en la completitud en las especificaciones particularmente los requisitos y criterios de aceptación (que sean claros y medibles), identificar tareas a realizar, estimar el esfuerzo de dichas tareas y registrar el esfuerzo real, diseñar, ejecutar y reportar resultados de las pruebas a nivel de integración y del sistema, y colabora con el dueño del producto en las pruebas de aceptación.

Responsable de calidad.- Es responsable por asegurarse de que el equipo defina su estrategia de calidad y verificación, los artefactos y recursos de soporte necesarios para llevarla a cabo. Promueve que el equipo evalúe y detecte mejoras, y tenga conocimiento acerca de cómo se va desarrollando la estrategia, y la calidad del producto y del proceso.

Responsable de seguimiento.- Es responsable por asegurarse de que el equipo identifique y gestione los riesgos del proyecto, que defina la estrategia de gestión de proyecto, y los recursos de soporte necesarios para la gestión del producto, proyecto, actividades y tareas. Promueve y colabora en el seguimiento del proyecto, construcción de producto, y logro de objetivos impulsando la reflexión y mejora sobre estos. Debe velar porque los interesados conozcan el

estado general en que se encuentra el proyecto y el producto.

Responsable de gestión de la configuración.- Es responsable de asegurar que el equipo defina la estrategia y la hoja de ruta para llevar a cabo la gestión de la configuración, defina y monte la infraestructura necesaria. Debe promover una gestión controlada de los artefactos y los cambios sobre estos, y ser referente en el manejo de las herramientas de soporte.

Responsable de diseño y construcción.- Es responsable por asegurar que el equipo defina y evolucione en cada iteración la arquitectura del producto, y de ser necesario colaborar en la coordinación de su construcción. Se encarga de promover: la identificación oportuna de riesgos asociados a la arquitectura, y requisitos no funcionales, el análisis de impacto de nuevos requisitos asociados a un nuevo incremento en el producto, evaluación periódica del diseño y necesidades de refactorización.

Al asignar los roles en el equipo de desarrollo se deben tener una serie de consideraciones:

- un rol no tiene por qué estar asociado de forma directa y exclusiva a un integrante del equipo durante todo el proyecto, y un integrante no tiene por qué ejecutar un único rol.
- un estudiante no debería estar asignado a más de dos roles por iteración. Esto es para prevenir sobrecarga de trabajo en algún estudiante o la existencia de una falta de foco en su aporte principal al proyecto por tener muchas responsabilidades, previniendo que se generen desvíos en el proyecto por ello o perjudique su aprendizaje.
- al menos durante una iteración todos deben haber desempeñado el rol de desarrollador. Esto es para que todos experimenten una de las actividades principales de un processo de desarrollo de software, y para prevenir divisiones en el equipo, por ejemplo entre testers y desarrolladores, o responsables y desarrolladores.
- es posible incorporar nuevos roles, en caso de que sea necesario por las características del proyecto. Por ejemplo se puede incorporar el rol responsable de diseño/interacción de usuario, en aquellos casos donde sea muy importante para el dueño del producto que se realice un trabajo intenso con los potenciales usuarios del producto.

Las posibles combinaciones de roles, que puede desempeñar una misma persona durante una iteración, se pueden visualizar en la tabla 5.5. Se debe

tener en cuenta que las filas corresponden al rol principal del actor y las columnas son los posibles roles secundarios a combinar, teniendo en cuenta las consideraciones mencionadas.

Posibles combinaciones de roles											
	Rol										
Rol	Coach	Cliente	Dueño del producto	Usuario	Facilitador	Desarrollador	Tester	Responsable de calidad	Responsable de seguimiento	Responsable de SCM	Responsable de diseño y construcción
Coach											
Cliente											
Dueño del producto											
Usuario											
Facilitador											
Desarrollador											
Tester											
Responsable de calidad											
Responsable de seguimiento											
Responsable de SCM											
Responsable de diseño y construcción											
Leyenda											
		Se sugiere combinar			Se puede combinar			Se sugiere no combinar			

Figura 5.5: PISAgile: Posibles combinaciones de roles

5.2.3. Artefactos

Cuando un grupo de estudiantes utiliza PISAgile en su proyecto, crean y evolucionan artefactos. Algunos de los artefactos son descritos en el proceso y otros los definen el equipo y el dueño del producto acorde a las necesidades del proyecto. En esta sección se brinda una breve definición de los principales artefactos. Para conocer todos los artefactos en detalle de PISAgile se debe consultar la especificación completa de PISAgile que se encuentra publicada en el repositorio Zenodo[1], en el "Anexo 1 - Artefactos detalle" del documento "05 - Especificación de la Fase de transición PISAgile 2020 v1.0.pdf"

Los artefactos asociados de forma directa al **producto de software (o sistema)** o el propio producto son:

Producto final.- es el conjunto de todos los artefactos entregables (productos) en su versión final, o sea que no serán evolucionados en el marco del proyecto.

Visión global del producto (visión, público objetivo, necesidades, producto, objetivos de negocio y restricciones).- es un artefacto cuyo objetivo es ayudar a describir, visualizar y validar la visión del producto, y la estrategia, generando una visión común de lo que se espera.

[1] https://doi.org/10.5281/zenodo.6730363

Descripción del problema/necesidad.- es una descripción de la necesidad del dueño del producto abstrayéndose de posibles soluciones.

Lista de criterios de aceptación generales.- es una lista de criterios de aceptación generales que el producto debe satisfacer para ser aceptado por el dueño del producto y otros interesados.

Producto mínimo viable (PMV), (del inglés, Minimum Viable Product -MVP-).- es un producto que tiene las funcionalidades y características mínimas para satisfacer las necesidades del dueño del producto y otros interesados.

Plan de liberación, (del inglés, release).- corresponde a un plan que indica las liberaciones que se van a realizar al dueño del producto. Para este proceso una liberación es una entrega del incremento que cumple con características acordes a una puesta en producción.

Incremento (de la iteración).- mencionando en sección 5.1.2

Microincremento.- mencionando en sección 5.1.2

Conceptualización técnica.- es un conjunto de diagramas, bosquejos, prototipos, etc., de alto nivel que muestre una conceptualización inicial de la arquitectura y diseño, infraestructura y el contexto del sistema para ser un punto de partida para la arquitectura base e identificar las necesidades técnicas y formativas.

Conceptualización del diseño de la interacción.- se compone de la identificación de pautas generales y construcción de prototipos para el diseño de las interfaces de usuario y la interacción, y actividades vinculadas a la experiencia de usuario y diseño de interfaz.

Descripción del producto de software (o sistema).- es un documento que resume la descripción general y específica del producto -requisitos funcionales y no funcionales-, y sugerencias de posibles evoluciones del mismo.

Descripción del diseño de la arquitectura y técnico.- artefacto donde se especifica la arquitectura, el diseño y la infraestructura del sistema. Este artefacto es una evolución del artefacto "Conceptualización técnica".

Descripción del diseño de la interacción de usuario.- artefacto donde se especifica el modelado (prototipos, pautas, guías, etc.) del diseño de la interfaz de usuario y la interacción. Este artefacto es una evolución del artefacto "Conceptualización del diseño de la interacción".

Los artefactos asociados al **proceso y el proyecto, y su ejecución** son:

Propuesta de proyecto.- breve descripción de lo que el cliente espera del

proyecto, sus principales características, e información general del cliente.

Trade off sliders.- artefacto resultante de aplicar la técnica con el mismo nombre. La técnica ayuda a priorizar visualmente las dimensiones del producto o líneas de trabajo.

Backlog del producto (BP).- es el mismo concepto que en Scrum, la diferencia está en la especificación de los ítems que lo componen. Pueden estar vinculados directa o indirectamente al producto de software (o sistema), otros artefactos que no son de software, o logro de resultados. Los ítems del backlog del producto (IBP) pueden estar especificados a distinto nivel de refinamiento (se va ajustando a medida de que avanza el proyecto), tienen una descripción, un tipo (tipos de IBP), criterios de aceptación y un conjunto de datos para su gestión. Los tipos de IBP asociados a productos de software se clasifican según tipos de requisitos y nivel de refinamiento, estos son: característica, épica, historia de usuario, escenario de uso, IBP técnico, historia técnica e IBP de mejora. Los tipos de IBP no asociados (de forma directa) a productos de software no son de construcción pero deben realizarse, estos son *spike* e investigación. Los criterios de aceptación aplican ya sea que el ítem sea construido o ejecutado.

Backlog de la iteración (BI).- está compuesto de dos conjuntos de ítems: ítems del backlog del producto para lograr el objetivo de la iteración, e ítems correspondientes a otras tareas que debe realizar el equipo durante la iteración en general vinculadas a las disciplinas de soporte o actividades puntuales (participar en reuniones, evaluación de las líneas de trabajo, etc.). Cada ítem tiene detallado el conjunto de tareas necesarias para su construcción o ejecución, y debe ser de tipo historia de usuario, escenario de uso, historias técnica, *spike*, investigación e IBP mejora, u otras tareas. O sea, se excluyen los tipos de ítems del BP que no han tenido refinamiento. En el BI se visualiza todo el trabajo que tiene que realizar el equipo durante la iteración.

Definición de listo (del inglés, Definition of ready (DoR)).- es el criterio utilizado para indicar que un ítem del backlog del producto está listo para comenzar su construcción o ejecución, o sea cuando está pronto para ser incluido en el backlog de la iteración.

Definición de hecho (del inglés, Definition of done (DoD)).- es el criterio utilizado para indicar que el trabajo sobre el incremento de una iteración o un ítem del backlog del producto está terminado, y que todos los involucrados comprendan y estén alineados con el significado e implicancia de que el ítem

o incremento esté "terminado".

Acuerdos de equipo.- son reglas establecidas y acordadas de forma colaborativa por todo el equipo que reflejan el comportamiento esperado de forma individual y grupal, basados en los valores y principios de PISAgile.

Plan de acciones y actividades de equipo.- acciones y actividades que el equipo decide incluir para conocerse y desarrollar la integración, y su organización. Pueden estar vinculadas al trabajo o ser de confraternización.

Acuerdos de trabajo.- son acuerdos en donde se reflejan las adaptaciones y definiciones realizadas al proceso y definen el entorno de trabajo.

Estrategia y *roadmaps*.- su objetivo es guiar el trabajo del equipo, brindar pautas generales, y recursos básicos necesarios para las siguientes líneas de trabajo: gestión de proyecto, calidad del producto y verificación, y gestión de la configuración. Se pretenden orientar a los involucrados del proyecto a un nivel general acerca de los pasos que se deben dar, qué aspectos se tendrán en cuenta, la forma de trabajo y la instanciación que realicen del proceso a su proyecto. Su construcción es evolutiva en las fases inicial y de construcción.

Documento de riesgos.- documento usual de gestión de riesgos.

Plan de la iteración.- debe contener el objetivo de la iteración, esfuerzo total e ítems del backlog de la iteración y calendario con principales actividades con fechas límite de entrega, etc. Esto es para que todos los participantes tengan claro cómo y qué hacer considerando las diferentes líneas de trabajo. El objetivo de la iteración se logra cuando se ejecutan o construyen los ítems del backlog de la iteración, y da sentido y orientación al trabajo.

Plan de mejora de la iteración.- se compone de acciones que el equipo planifica realizar a partir de la próxima iteración y sus responsables.

Plan y cronograma de transferencia.- debe dar una visión global y específica del alcance acordado con el dueño del producto sobre la transferencia, y el cronograma indicando las tareas a realizar y su ubicación temporal.

Plan de fase de transición.- análogo al "Plan de la iteración", dado que la fase es considerada como una única iteración.

Los artefactos más relevantes para el **seguimiento y reporte** continuo se listan a continuación. Dos de estos artefactos son base para el seguimiento del proyecto: backlog del producto y el backlog de la iteración (ya mencionados).

Informe semanal.- debe proporcionar el estado del proyecto desde una mirada holística incluyendo el trabajo en las disciplinas, la aplicación del proceso,

el avance en la conformación del equipo, y el desarrollo a nivel técnico y metodológico de forma individual y grupal.

Informe de la iteración.- análogo al "Informe semanal" pero con otra ventana temporal. Debe proporcionar elementos que permitan visualizar el trabajo realizado durante la iteración y los objetivos logrados, es decir el resultado de la iteración. Se debe presentar el cálculo y análisis de las métricas, aspectos importantes vinculados a la revisión del incremento, y una conclusión general.

Tabla de registro de esfuerzo.- debe contener el registro del esfuerzo de cada estudiante de forma desagregada y agregada respecto a las tareas.

Tablero de seguimiento de la iteración.- es un tablero kanban para visualizar el avance hacia el logro del objetivo de la iteración. Dado que hay distintos tipos de ítems del backlog del producto, es conveniente contar con distintos flujos de trabajo asociados a cada uno. En el caso de los que aportan a la construcción del incremento, el flujo de trabajo queda definido por la definición de hecho y el límite del flujo de trabajo en progreso (WIP) definido.

Gráfico del trabajo pendiente de la iteración (puntos de esfuerzo/días).- en este gráfico se visualiza el trabajo realizado de forma conjunta con el que queda por realizar para hacer los ajustes necesarios de forma oportuna. Se consideran todos ítems del backlog de la iteración a excepción de los que son del tipo otras tareas.

Gráfico de las horas de trabajo pendiente de la iteración (horas/días).- es similar al gráfico del trabajo pendiente de la iteración, con la diferencia de que se contabiliza en horas de trabajo (hst) y se incluyen todos los ítems del backlog independientemente de su tipo.

Gráfico del trabajo realizado para el producto final (puntos de esfuerzo/iteración).- el objetivo de este gráfico es colaborar con el seguimiento del trabajo realizado para la obtención del producto final. Se visualiza el avance en el trabajo realizado vinculado con el alcance comprometido. Se construye indicando el trabajo realizado durante la iteración, para contemplar cambios de alcance en el proyecto. Para este gráfico no se consideran los ítems del backlog de la iteración de tipo otras tareas .

Gráfico de la Velocidad del equipo.- se muestra la evolución de la velocidad del equipo a través de las iteraciones para ser utilizado como instrumento para negociar alcances. La velocidad del equipo es una medida de la cantidad de trabajo que el equipo puede completar durante una iteración. Se calcula al final de la iteración sumando los puntos de esfuerzo de todos los ítems del

backlog de la iteración que han sido aceptados, sin considerar los que son del tipo otras tareas.

Informes finales.- los informes finales del proyecto, la gestión de la configuración, y de calidad y verificación deben presentar los resultados obtenidos, la evaluación de la brecha entre la estrategia y *roadmap* y lo que sucedió realmente, y el desarrollo del proyecto a través de las fases e iteraciones.

Encuesta de satisfacción al cliente.- se deben evaluar aspectos vinculados al producto, proceso y proyecto, e ir dirigida al cliente y otros interesados.

Lecciones aprendidas.- son para transferir el conocimiento sobre el aprendizaje, experiencias y recomendaciones, que se tuvo durante el proyecto para que otros las puedan aprovechar. Se debe contemplar el aprendizaje de las disciplinas, y los valores y principios.

Presentación final.- tiene como objetivo brindar una visión global del trabajo y el aprendizaje del equipo sobre el proceso y el producto. Esta presentación será utilizada en la instancia de evaluación final del curso donde el grupo realizará la presentación ante un tribunal.

5.2.4. Fases

Fase inicial

Esta es la primer fase del proceso en la cual se forma el equipo -los integrantes se conocen-, se toma el primer contacto con el proceso, el proyecto, el cliente (dueño del producto) y el *coach*. El equipo y el cliente comienzan a tomar contacto con el proceso y se toman definiciones sobre la instanciación del proceso al proyecto.

Los objetivos de la fase son:

- Conceptualizar el problema y la necesidad del dueño del producto a la cual se debe dar respuesta. El objetivo de ello es abstraerse de posibles alternativas de soluciones que el dueño del producto pueda plantear y que no permitan evaluar una mejor solución.
- Conceptualizar la solución y generar una visión global del producto para alinear la visión de todos los involucrados acerca de los objetivos, productos y resultados esperados.
- Comenzar a conformarse como equipo para poder generar sinergia entre los integrantes y así obtener mejores resultados, de forma eficiente y que

todos se sientan parte del logro de los objetivos del proyecto.

- Comenzar a promover un entendimiento del producto a desarrollar, una alineación del trabajo y el entendimiento entre el equipo, el dueño del producto y el resto de participantes del proyecto.
- Acercamiento de los integrantes del equipo a los conocimientos generales que debe tener cada integrante tanto a nivel tecnológico como metodológico.
- Conocer el proceso para poder ajustar algunas características del mismo a la realidad del proyecto, sus objetivos, restricciones y al equipo que lo va a llevar a cabo. Por ejemplo, encuentros con el dueño del producto, encuentros del equipo, etc.
- Identificar los principales riesgos asociados al proyecto y al producto para comenzar a gestionarlos desde un principio.
- Definir pautas generales y una conceptualización del sistema que permitan comenzar a razonar sobre una arquitectura base de alto nivel, la interfaz de usuario, y los lenguajes y tecnologías principales a utilizar, así como también validar algún aspecto con el dueño del producto y también para evaluar riesgos.
- Definir estrategias de alto nivel y comenzar a delinear hojas de ruta, contemplando los aspectos necesarios para el desarrollo de las disciplinas involucradas con el objetivo de contar con lineamientos generales acerca del trabajo a realizar y una guía para seguir durante el proyecto. No se espera que se definan planes detallados, sino que sirvan para contar con una hoja de ruta del trabajo y una descripción de cómo van a trabajar en las diferentes disciplinas.

Fase de construcción

La fase de construcción es la fase central del proceso. Esta consiste en la construcción y aceptación del producto de software (o sistema) que da respuesta a la necesidad del cliente y de otros interesados, así como también la construcción de otros productos o generación de resultados que hayan sido identificados. La fase está compuesta por cinco iteraciones y cada una se ejecuta en un conjunto de semanas. Cada iteración genera un nuevo incremento que incluye lo que se construyó hasta la iteración anterior más lo que se construya en la propia iteración.

Los objetivos de la fase son:

- Refinar los requisitos, restricciones y criterios de aceptación asociados a los productos a construir y resultados a obtener.
- Aplicar, evaluar y ajustar estrategias y hojas de ruta del proyecto, que sirvan de guía para el abordaje de las disciplinas involucradas.
- Diseñar, implementar, evaluar y ajustar la arquitectura completa, teniendo en cuenta aspectos físicos y lógicos.
- Minimizar la ocurrencia o impacto de los riesgos asociados al producto y proyecto, y aumentar la acumulación de entrega de valor al cliente.
- Conformar un equipo que trabaje sinérgicamente para el logro de un objetivo conjunto y haya adoptado el proceso basándose en la incorporación de los valores y principios.
- Medir, evaluar y adaptar de forma continua el proceso, para mejorar la calidad del mismo y por consiguiente la calidad del producto, aplicando las métricas e indicadores definidos.
- Definir, acordar el alcance y construir un producto mínimo viable que tenga las funcionalidades y características mínimas para satisfacer las necesidades del cliente y otros interesados (en las dos primeras iteraciones debería quedar acordado con el dueño del producto el PMV).
- Realizar una gestión continua del alcance alineándose con las expectativas del dueño del producto, y obtener la construcción completa de los productos acordados y lograr su aceptación por parte de este.
- Realizar una gestión del conocimiento adecuada durante el proyecto, generando los recursos (documentación, herramientas, etc.) necesarios para la colaboración entre integrantes del equipo y también con el dueño del producto.

Fase de transición

Esta es la última fase del proceso. En esta se realiza el traspaso definitivo de los productos al cliente y a la memoria organizacional, y se evalúa y reflexiona sobre lo acontecido en el proyecto.

Los objetivos de la fase son:

- Realizar el traspaso final de los productos al cliente, debe incluir todo lo necesario para que lo pueda operar y evolucionar según lo que se haya acordado.

- Asegurar que todos los productos finales tengan la calidad adecuada para su transferencia.
- Evaluar la brecha entre las estrategias y hojas de ruta trazadas por disciplina y lo que se logró, para comparar lo definido versus lo que sucedió realmente.
- Evaluar cómo resultaron las adaptaciones al proceso y reflexionar sobre ellas.
- Evaluar el aprendizaje que dejó el proceso y el proyecto, a nivel individual y grupal. Este debe ser evaluado por el equipo y el *coach*.
- Generar las lecciones aprendidas del proyecto, y el proceso y su aplicación.
- Aportar a la gestión del conocimiento del curso traspasando la información del proyecto a la memoria organizacional de la asignatura.
- Recolectar y evaluar el nivel de satisfacción respecto al producto por parte del cliente y otros interesados.

5.2.5. Niveles de planificación y evaluación de progreso

La planificación, seguimiento y la evaluación del progreso debe hacerse a tres niveles: ciclo de vida del proyecto, ciclo de vida de la iteración y tareas. Esto brinda al equipo una visión general de objetivos a largo plazo que lo guíen, y pueda trabajar focalizado en objetivos cortos, y adaptarse rápidamente a los cambios. Los tres niveles no son independientes, las decisiones que se tomen en cada uno va a impactar en otro.

El nivel **ciclo de vida del proyecto** es una planificación y evaluación de largo plazo. La ventana temporal de la medición es la duración de una iteración, y el foco debe estar en lograr los objetivos de la fase, la obtención de los artefactos (principalmente el producto final) y resultados finales. Se debe evaluar el progreso en el incremento hacia la obtención del producto final, en la aplicación del proceso, en las estrategias y hojas de ruta de cada línea de trabajo, y el estado de los riesgos. Se deben realizar las evaluaciones definidas, que deben ser cualitativas (actividades de reflexión) y cuantitativas (métricas, indicadores y registros), y apoyarse en los artefactos claves de seguimiento.

La planificación y evaluación del progreso asociada al **ciclo de la iteración** es a corto plazo. Cuenta con dos ventanas temporales: (i) espacio entre las reuniones de sincronización y (ii) semanal. La primera se focaliza en ayudar a

que el equipo pueda colaborar para sortear obstáculos en la realización de las tareas y la segunda en el logro del objetivo de la iteración. La primera tiene en cuenta la evaluación del progreso a nivel personal y la segunda teniendo en cuenta el estado del tablero de seguimiento y el incremento.

A nivel de **tareas** la planificación y evaluación del progreso es individual y por el periodo comprendido entre dos reuniones de sincronización. Por lo tanto cada integrante del equipo debe planificar, medir, registrar y evaluar su trabajo diario; avance que ha tenido y dificultades en las que se encuentre.

Al momento de realizar la planificación, evaluación y hacer ajustes se debe considerar: las estrategias y hojas de ruta; otros artefactos de seguimiento según el nivel que corresponda (ver figura 5.6); ser frecuente respetando las ventanas temporales; y asegurar que todos los integrantes del equipo, el dueño del producto y *coach* puedan visualizar el estado actual de la planificación y el progreso mediante los artefactos que brindan transparencia y las actividades.

En la figura 5.6 se pueden visualizar los distintos niveles de planificación (ciclo de vida del proyecto, ciclo de la iteración y tareas/actividades), artefactos entregables claves (incremento y producto final) y los principales artefactos de seguimiento. El producto final está asociado al ciclo de vida del proyecto, su construcción está guiada por el backlog del producto, y las estrategias y *roadmaps* del proyecto. En las iteraciones (ciclo de la iteración) se genera el incremento guiado por el backlog de la iteración, definición de hecho y el plan de la iteración. Durante cada iteración se planifican y ejecutan tareas que son guiadas por los ítems del backlog de la iteración y las actividades definidas en el proceso, de dicha ejecución surgen los microincrementos que serán parte del incremento.

5.2.6. Métricas y medidas

En PISAgile se proponen una serie de métricas para ser aplicadas durante el proyecto pudiendo incluir otras, dependiendo de las características del proyecto y las estrategias definidas para cada disciplina. Se recomienda definir una forma de registro que permita obtener las métricas de la forma más automatizada posible. Las métricas se dividen en tres tipos: esfuerzo, avance, y calidad del producto y del proceso. Las métricas de esfuerzo tienen por objetivo evaluar el esfuerzo medido en horas de trabajo (hst). Las de avance del proyecto debe analizarse de dos perspectivas; una considerando el avance

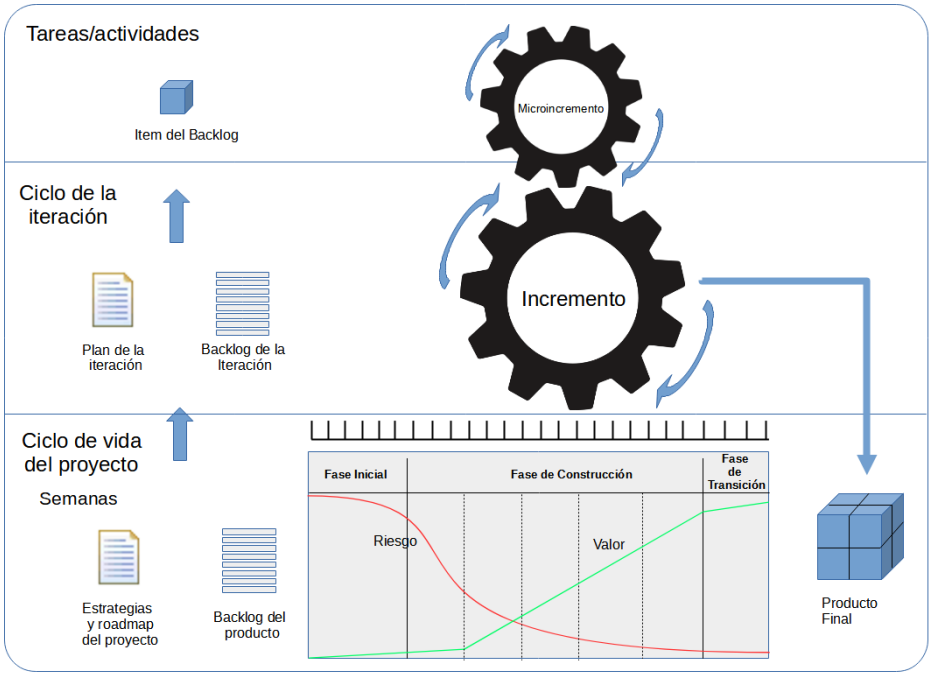

Figura 5.6: PISAgile: Artefactos entregables y niveles de planificación

que se logró en determinada ventana temporal para lograr los objetivos del proyecto, y otra es para evaluar qué tan ajustado está estimando el equipo. Respecto a las de calidad del producto y del proceso se deben considerar para evaluar causas y mejoras de manera cuantitativa y cualitativa. En el Anexo 1 *Métricas, indicadores y medidas* se pueden ver las métricas definidas.

5.2.7. Estimación en el proceso

El proceso de estimación del trabajo a realizar durante el proyecto tiene en cuenta los distintos niveles de planificación para acordar el alcance, compromiso y mantener una visión global a largo plazo. La estimación en el ciclo de vida de la iteración es para acordar el trabajo previsto para la iteración, y en el ciclo de vida del proyecto es para acordar el producto mínimo viable y mantener una visión global (dada por el backlog del producto).

La estimación para la iteración se realiza a dos niveles y en actividades diferentes; la estimación de ítems del backlog del producto (que se van a incluir en próximas iteraciones), durante la actividad refinar y estimar los ítems del backlog del producto, y la estimación de cada ítem del backlog de la iteración, durante la actividad planificar la iteración.

Para las estimaciones se utilizan dos unidades de medida, puntos de esfuerzo (pe)[1] y horas de trabajo (hst)[2].

Para estimar el esfuerzo de un ítem del backlog del producto (IBP) la unidad de medida utilizada son los pe. Las estimaciones deben ser por analogía y relativas, y en consenso del equipo. Es importante que el equipo vaya generando un histórico con las estimaciones, su velocidad por iteración y su desviaciones. Para estimar los IBP se proponen dos métodos distintos según estén o no asociados al producto de software.

Para realizar la estimación de un IBP asociado a un producto de software, este debe tener asociadas las tareas acorde a la definición de hecho y estar asociada a determinado rol. Se propone aplicar la técnica *Planning poker*, el consenso es a nivel de rol para todas sus tareas y luego se sumarizan los resultados de todos los roles.

Para la estimación de un IBP no asociado a un producto de software se propone un procedimiento similar al de la estimación de IBP asociado a producto de software, a diferencia que la estimación se realiza considerando el ítem completo (no se divide en tareas) y el consenso es a nivel de todo el equipo (independientemente del rol).

En la estimación de cada ítem del backlog de la iteración (IBI) la unidad de medida utilizada son las hst, y debe estimarse de forma individual por el responsable de cada tarea y en caso de ser necesario de forma grupal, para cada tarea asociada al IBI. Esta estimación es aplicable luego de asignar los responsables a cada tarea definida en cada ítem del backlog de la iteración. Esto debe hacerse para todos los ítems del backlog de la iteración.

Para mejorar el ajuste de las estimaciones y para la negociación con el dueño del producto es importante que el equipo se mida y registre los resultados de la ejecución real. Para ajustar las hst se deben registrar las horas invertidas durante la ejecución o construcción del los ítems y cuando se completan las tareas e ítems se debe registrar su completitud y fecha. Al momento de acordar el alcance de la iteración, el equipo debe definir la velocidad del equipo a la cual se puede comprometer considerando la evolución de la misma y la estimación de cada ítem del backlog del producto (en pe).

[1]Unidad de medida utilizada para realizar estimaciones relativas, expresa el esfuerzo requerido para implementar/ejecutar un ítem del backlog del producto(IBP) o parte de este, teniendo en cuenta diversos parámetros: cantidad de trabajo, complejidad y riesgo.

[2]Unidad de medida para expresar cantidad de horas estimadas y reales de trabajo que lleva ejecutar una tarea de un ítem del backlog de la iteración.

La estimación a nivel del ciclo de vida, es principalmente para acordar el alcance del producto mínimo viable. Se hace al final de la segunda iteración o principios de la tercera de la fase de construcción. El equipo realiza una estimación general de los IBP, partiendo de los de mayor prioridad y considerando el esfuerzo al cual se pueden comprometer, negocian con el dueño del producto un producto mínimo viable, y de ser posible pueden identificar un alcance mayor. Los IBP se estiman por analogía y no es necesario un refinamiento previo de cada uno (aunque ayuda a tener mayor información). Se deben seleccionar los ítems a incluir considerando: la velocidad del equipo, las iteraciones que quedan, el valor asignado al IBP y el resultado de la *User Story Mapping* (si lo tienen).

5.3. Especificación de las actividades

En esta sección se presenta el modelo de especificación definido para describir de forma detallada las fases y actividades con el fin de mostrar cómo fueron especificadas en la documentación completa de PISAgile. También se realiza una descripción de las actividades de cada fase y cómo estas interactúan. Para el caso de la fase de construcción se incluye un diagrama de interacción de los artefactos. La especificación tanto de las fases como de las actividades se ha resumido resaltando los elementos más importantes para su comprensión. Su especificación completa se puede ver en el repositorio Zenodo[1].

El modelo de especificación de fase es una plantilla en formato documento, diseñado para describir de forma detallada una fase y ser usado por todos los involucrados, acompañado de una plantilla auxiliar en formato de planilla electrónica donde se resumen las actividades y artefactos -Planilla de construcción de artefactos y entregas semanales-.

En el documento se especifica para la fase: la descripción general, objetivos, duración, criterios de fin de fase, artefactos a utilizar, un detalle de las iteraciones y actividades junto con un diagrama que presenta las relaciones entre ellas. En el caso de la fase de construcción se agrega una vista general de los artefactos y sus relaciones.

Con el fin de ayudar a la lectura del proceso por parte de los interesados, se dotó al documento de referencias entre los elementos para brindar una mayor

[1] https://doi.org/10.5281/zenodo.6730363

navegabilidad. En la descripción se indica su razón de ser y finalidad, y los criterios de fin de fase permiten verificar si se cumplieron con los objetivos y el trabajo a realizar durante la fase. Se listan y detallan los principales artefactos a utilizar durante la fase, estos pueden estar vinculados al producto de software (o sistema) incluyendo el propio software y otros artefactos necesarios para conocer sus características y su estado constructivo. Se indican también otros artefactos que se encuentran asociados al proceso/proyecto y se resaltan los principales artefactos de seguimiento y reporte a considerar durante la fase. El diagrama de relaciones de actividades se diseñó utilizando la herramienta Bizagi modeler[1]. El objetivo es mostrar las relaciones entre las actividades a modo de proceso para cada fase. Las actividades son descritas utilizando el modelo de especificación de actividades.

En la planilla electrónica se resumen las actividades, los artefactos y resultados esperados en cada una de ellas junto con el nivel de desarrollo que estos deben tener, la iteración y semana a la cual corresponde la ejecución, y también se indican aquellos artefactos que deben ser entregados.

El modelo de especificación de actividad es una plantilla en formato tabular (ver tabla 5.1) donde se presenta en detalle todo lo necesario para la ejecución de la actividad. Este modelo es utilizado para todas las actividades del proceso.

La tabla contiene: el nombre de la actividad; su objetivo; se brinda una descripción detallada acerca de cómo debe ejecutarse dicha actividad resaltando aquellos aspectos sobre los que hay que tener especial cuidado junto con algunas recomendaciones; se describen las disciplinas con las que se vincula la actividad; los roles que van a participar (participantes); las precondiciones que son necesarias para ejecutar la actividad (si corresponde); un estimativo de la duración y frecuencia en los casos que corresponda, aunque ello va a depender del proyecto; la ubicación temporal de la actividad para brindar una pauta más específica acerca de en qué momento ejecutarla.

En el caso de la actividad presentada como ejemplo (tabla 5.1), como es de ejecución única en la iteración se indica el momento en que debe ejecutarse asociándolo a la iteración y a la actividad predecesora. También se definen las entradas que se utilizan para el desarrollo de la actividad, y las salida de los artefactos, resultados o productos generados durante la ejecución de la actividad.

Se proporcionan determinados recursos de soporte que incluyen: prácticas,

[1]https://www.bizagi.com/es/plataforma/modeler

técnicas, guías, herramientas, estándares y memoria del curso. Algunos de estos recursos se definieron específicamente para PISAgile como es el caso del ejemplo "Template para especificar ítems del Backlog del Producto (IBP)" y otros como ser la técnica de *Planning Poker* ya existen en el ámbito de la ingeniería del software. Dado que estas son recomendaciones, en algunos casos son genéricas para que los propios involucrados la definan. En el ejemplo, se da este caso en "Uso de aplicaciones que permitan gestionar el backlog del producto" donde no se indica cuál es la aplicación de gestión a utilizar.

Tabla 5.1: PISAgile: modelo de especificación de una actividad

Refinar y estimar los ítems del backlog del producto (IBP)
Objetivos Examinar y analizar los potenciales ítems que serán necesarios para lograr el objetivo de la iteración siguiente inmediata. Identificar dependencias entre los ítems, evaluar si es necesario incorporar nuevos ítems (e incorporarlos) o particionar los existentes, identificar tareas necesarias para que cada ítem pueda ser construido o ejecutado, y estimar el esfuerzo total que implicará construir o ejecutar el ítem. El objetivo principal es desarrollar los ítems hasta lograr unidades que sean posibles estimar para contar con los elementos necesarios para poder acordar y negociar con el dueño del producto el alcance de la siguiente iteración.
Descripción Para la ejecución de esta actividad es importante que previamente se haya ejecutado recurrentemente la actividad "Actualizar y refinar el backlog del producto", para contar con el refinamiento y comprensión adecuada de los ítems necesarios. Esta actividad se puede llevar a cabo en dos instancias: 1. Verificar que todos los ítems identificados como potenciales para la iteración inmediata siguiente hayan sido ajustados y validados por el dueño del producto. También asegurar de tener el refinamiento y detalle necesario para comprenderlos. Esta instancia debe ser llevada a cabo por el responsable de calidad y testers (acorde a lo definido en la Estrategia y *roadmap* de calidad y verificación). 2. Realizar una reunión de trabajo donde participe el equipo para poder cumplir con el objetivo de la actividad.
Continúa en la página siguiente...

Tabla 5.1: (continuación)

La reunión de trabajo debe abordar las siguientes tareas: - Particionar los ítems que tienen un alcance muy grande y no puedan ser completados en una única iteración. - Identificar los ítems que el equipo considere necesarios para ser construidos o ejecutados en la iteración inmediata siguiente. Entre ellos, spikes necesarios para desarrollar los ítems, ítems de investigación que el equipo necesita abordar para adquirir conocimientos, IBP técnicos que aporten a varias historias de usuario o escenarios, IBP de mejora y pueden surgir algunos vinculados a la infraestructura necesaria. El equipo debe asignar prioridades. - Identificar y asociar las dependencias que existen entre ítems (los refinados en la actividad "Actualizar y refinar el backlog del producto" y los nuevos que se deben agregar). - Estimar el esfuerzo asociado a la construcción o ejecución de los ítems. Para ello se deben identificar las tareas necesarias. Identificar tareas necesarias En el caso de los ítems que aportan a la construcción del producto de software de forma directa (historias de usuario, escenarios, historias técnicas, etc) las tareas están asociadas a la definición de hecho. Mientras que en los otros casos que no están asociados directamente en la construcción (spike, investigación) se deben identificar cuáles serían las tareas específicamente en cada caso. Estimar el esfuerzo En Estimación de cada ítem del backlog del producto se encuentra el detalle de cómo estimar cada ítem del backlog del producto.
Disciplina requisitos, calidad del producto y verificación, gestión del proyecto
Participantes - facilitador, desarrollador, tester, responsable de calidad, responsable de seguimiento, responsable de gestión de la configuración, responsable de diseño y construcción.
Duración y frecuencia, ubicación temporal - única ejecución en la iteración, en la iteración i debe completarse para la i+1 - debe ser realizada luego de que la salida de la actividad "Actualizar y refinar el backlog del producto" se encuentra en un estado final, esto es que el refinamiento de los potenciales ítems de la iteración i+1 e i+2 está pronto
Precondiciones -
Entrada
Continúa en la página siguiente...

Tabla 5.1: (continuación)

- Backlog del producto que tiene detallados los IBP que potencialmente se utilizarán en la iteración inmediata siguiente, y de ser posible dos iteraciones hacia delante. - Potenciales objetivos de la iteración que se utilizarán en la iteración inmediata siguiente, y de ser posible dos iteraciones hacia delante.
Salida - Backlog del producto que tiene refinados, estimados y asociadas las dependencias y prioridades (valor) de los ítems que potencialmente se utilizarán en la iteración inmediata siguiente.
Recursos de soporte Técnicas: - *Planning Poker* Herramientas: - Uso de aplicaciones que permitan gestionar y visualizar el backlog del producto. - Uso de aplicaciones que sean soporte a la estimación, ej: la app Scrum Pocker. - Template para especificar ítems del Backlog del Producto (IBP)

5.3.1. Fase inicial

Durante la fase inicial se construyen algunos artefactos que son clave para cumplir con sus objetivos y para la evolución del proyecto, estos se pueden visualizar en el Anexo 2 *Fases y sus artefactos*.

En la figura 5.7 se pueden visualizar las actividades de la fase inicial, el orden de ejecución básico, y si son recurrentes, repetitivas o de ejecución única dentro de la fase. Para interpretarlo se debe tener en cuenta que varias de las actividades son recurrentes o repetitivas, por lo cual el orden expresado no es exhaustivo, pretende dar una vista general de la ejecución y las actividades no generan precedencia en todos los casos sino que son insumos posibles para la ejecución de otra actividad. Para complementar la visión global proporcionada por el diagrama, es recomendable verlo junto con la planilla de construcción de artefactos y entregas semanales, que se puede ver en la documentación de PISAgile publicada en el repositorio Zenodo.

La actividad "Aprender proceso y tecnología" tiene como objetivo formar y nivelar conocimientos, técnicos y metodológicos de los integrantes del equipo y el cliente para llevar a cabo el proyecto. Se espera también que construyan un prototipo que contemple las tecnologías identificadas. [*repetitiva, al menos*

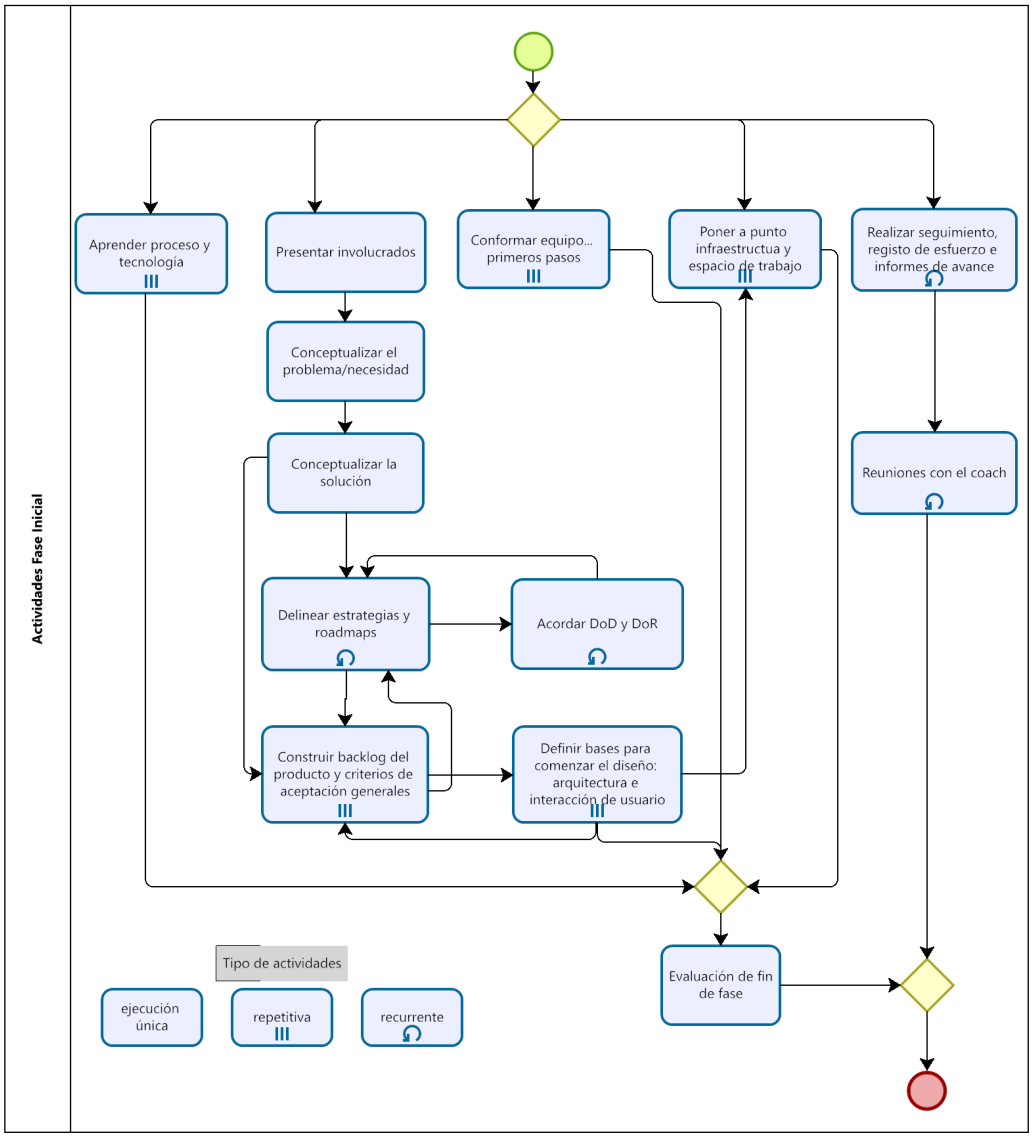

Figura 5.7: PISAgile: actividades de la fase inicial

una vez por semana según necesidad]

La actividad "Presentar involucrados" tiene como objetivo que los involucrados -el equipo, el coach y el cliente- en el proyecto se presenten y que el cliente cuente a nivel general la propuesta presentada para el proyecto. [*única vez, a principios de la primer semana*]

El objetivo de "Conceptualizar el problema/necesidad" es conocer y entender la necesidad real del cliente, abstraerse de la solución y lograr realizar una especificación de la necesidad y/o problema a resolver, ya que cuanto más ajustado a la realidad sea, soluciones más ajustadas se van a obtener. [*única*

vez, a principios de la primer semana]

Mediante "Conceptualizar la solución" se busca generar una visión global de la solución (productos, resultados) entre los involucrados, conocer el valor que la misma aporta y alinear las perspectivas de los interesados para lograr una visión común desde el punto de vista del negocio, incluyendo: visión, público objetivo, necesidades, producto, objetivos de negocio de la solución. [*única vez, a principios de la primer semana*]

La actividad "Conformar equipo ... primeros pasos" tiene como objetivo generar acuerdos de equipo, un plan de acciones y actividades que sean habilitadores y promuevan la conformación de un equipo, la adopción de los valores y principios del proceso, y la autogestión. A su vez, ejecutar al menos la primer actividad del plan. [*repetitiva, en las tres semanas -acorde al plan-*]

La actividad "Construir backlog del producto y criterios de aceptación generales" tiene como objetivo construir el backlog del producto (bp) en un estado inicial para tener una visión global del producto y criterios de aceptación generales para satisfacer al dueño del producto y otros interesados. El bp debe estar compuesto por un conjunto de ítems que reflejen las necesidades y requisitos asociados al producto, y que permita comprender y tener una visión general del mismo, así como también ciertas pautas de su evolución. [*repetitiva, en la segunda y tercer semana*]

La actividad "Delinear estrategias y *roadmaps*" tiene como objetivo contar con estrategias y hojas de ruta donde se refleje la forma en que se van a abordar las líneas de trabajo, dando pautas generales de alto nivel, guía sobre los pasos a dar en cada una, y algunos procedimientos y recursos básicos necesarios para llevarla a cabo. A su vez, deben ser identificados los principales riesgos y artefactos de soporte. Dependiendo de la disciplina se espera obtener la estrategia y *roadmap* correspondiente -gestión del proyecto, calidad y verificación, gestión de la configuración-, los acuerdos de trabajo, tablero de seguimiento de la iteración, y documento de riesgos. [*recurrente, durante las tres semanas*]

Durante la actividad "Acordar DoD y DoR" se espera que los involucrados acuerden la definición de listo y la definición de hecho que les permitan evaluar cuándo está terminado un artefacto, tipo de artefacto, microincremento o incremento, definiendo así el alcance de las actividades a realizar durante la construcción, brindando transparencia y promoviendo la calidad.[*recurrente, tercer semana*]

La actividad "Definir bases para comenzar el diseño: arquitectura e inter-

acción de usuario" tiene como objetivo definir pautas generales y realizar una conceptualización, tanto técnica de alto nivel como de la interacción (o interfaz) de usuario, que permitan definir las bases generales y las prioridades sobre las cuales se comenzará a definir el diseño y la construcción del producto. Esto es a nivel arquitectónico y de interacción con el usuario. Se espera que puedan construir un prototipo del diseño de la interacción con el usuario, un prototipo de aspectos técnicos y arquitectónicos que contemple el *stack* tecnológico a desplegar, y al menos la construcción de algún ítem del backlog. [*repetitiva, durante la segunda y tercer semana*]

La actividad "Poner a punto infraestructura y espacio de trabajo" tiene como objetivo definir, dejar instalada, configurada y disponible, la infraestructura y los ambientes -individuales y grupales- necesarios para comenzar a desarrollar el software, toda herramienta de soporte necesaria para el proyecto y asegurarse que todos los involucrados sepan utilizarlos. En la última ejecución de esta actividad repetitiva debe quedar todo pronto. [*repetitiva, durante las tres semanas*]

La actividad "Evaluación de fin de fase" tiene como objetivo definir el pasaje a la siguiente fase, para ello el equipo y el *coach* evalúan si se lograron los objetivos de la fase, se construyeron todos los productos y se lograron los resultados. Es una actividad que va a estar presente en todas las fases y se debe tener en cuenta el estado del proyecto (informes, evaluación de líneas de trabajo), y artefactos y resultados generados. [*única vez, última semana de la fase*]

Actividades de seguimiento semanales

Mediante "Realizar seguimiento, registro de esfuerzo e informes avance" se espera generar evidencia y conocer el estado general de lo que está ocurriendo en el proyecto (considerando todas las líneas de trabajo), y de las estrategias y hojas de ruta definidas. La finalidad de ello es medir, evaluar, adaptar, mejorar, y promover la colaboración y circular el conocimiento entre el equipo y el *coach*. [*recurrente, diario en el registro y semanal en el seguimiento*]

La "Reunión con el *coach*" tiene como objetivo que el equipo pueda intercambiar, reflexionar y definir acciones junto al *coach* acerca de distintos aspectos del proceso y proyecto para que este pueda ayudar en el aprendizaje, y desarrollar capacidades individuales y grupales para lograr mejores resultados. Para ello se utiliza la evidencia generada en la actividad "Realizar seguimiento,

registro de esfuerzo e informes avance", productos y resultados obtenidos. Se debe reflexionar sobre ajustes al proceso, proyecto, y tomar acciones sobre dificultades encontradas y entrenamiento del equipo. En ocasiones esta reunión se utiliza para la presentación de artefactos y resultados al *coach* . [*recurrente, todas las semanas*]

5.3.2. Fase de construcción

En la fase de construcción se evolucionan artefactos asociados al producto de software, hasta convertirlos en productos finales. Dado que es un proceso evolutivo, no todos los artefactos estarán presentes en todas las iteraciones. Estos se pueden visualizar en el Anexo 2 *Fases y sus artefactos*.

La fase de construcción consta de cinco iteraciones. Para cada una se define un objetivo vinculado a la entrega de un incremento al dueño del producto. Este incremento puede entregarse pronto para liberar en un ambiente productivo o no, esto debe acordarse con el dueño del producto. A su vez, en la iteración pueden existir objetivos secundarios donde el aporte es sobre el ciclo de vida del proyecto y no exclusivamente a la iteración, y todas las iteraciones van a heredar los objetivos de la fase de construcción y del proyecto. Durante cada iteración se ejecutan las actividades que se pueden visualizar en la figura 5.8 con un orden de ejecución básico, y si son recurrentes, repetitivas o de ejecución única dentro de una iteración. En cada iteración hay actividades cuyo foco está en la iteración actual, otras en futuras iteraciones (representadas en la figura con i+1 e i+2), algunas en la iteración inmediata anterior (representadas con i-1) y otras vinculadas al ciclo de vida del proyecto en general. A continuación se describen las actividades de esta fase.

Actividades con foco en incrementos futuros (obligatoriedad inmediato siguiente)

La actividad "Actualizar y refinar el backlog del producto" tiene como objetivo que el dueño del producto con el apoyo del equipo revise, actualice, refine su detalle, priorice y valide el backlog del producto, y se identifiquen los objetivos, para la iteración inmediata siguiente. El objeto de esto es obtener, especificar y validar los requisitos que definen el producto de software (o sistema) y otros artefactos entregables de forma iterativa. Se priorizan los ítems del backlog considerados para las siguientes dos iteraciones (siendo obligatorio

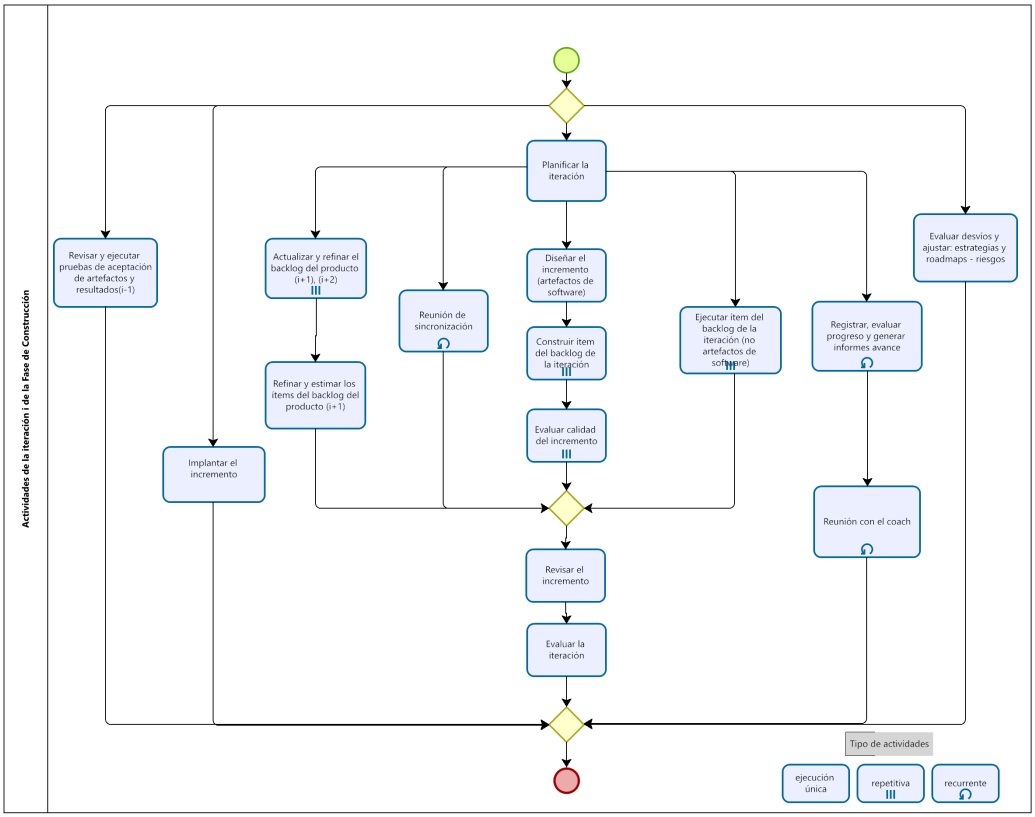

Figura 5.8: PISAgile: actividades de la iteración i de la fase de construcción

para la iteración siguiente) para que el equipo cuente con el detalle necesario. Se debe tener en cuenta el backlog del producto y el reporte de defectos encontrados, para actualizar el backlog del producto con el detalle de los ítems y los objetivos de iteración que potencialmente se utilizarán en la iteración inmediata siguiente (y de ser posible dos iteraciones hacia delante). [*repetitiva hasta lograr refinamiento esperado en el backlog*[1]*, comienzo de la iteración*]

Durante "Refinar y estimar los ítems del backlog del producto" el equipo debe examinar, analizar, refinar con el detalle necesario (o particionar) los ítems del backlog del producto hasta que sea posible estimarlos y estime el esfuerzo para construir (o ejecutar) cada uno de ellos que potencialmente serán necesarios para lograr el objetivo de la próxima iteración. Se debe utilizar el backlog del producto con el refinamiento logrado luego de finalizada la actividad "Actualizar y refinar el backlog del producto" y obtener un backlog del producto que tiene refinados, estimados, y asociadas las dependencias y

[1]El refinamiento de los potenciales ítems de la iteración i+1 (y si es posible i+2) están listos (según los criterios establecidos en la definición de listo).

prioridades (valor) para negociar con el dueño del producto el alcance de la siguiente iteración. [*ejecución única luego que el backlog del producto obtenido en "Actualizar y refinar el backlog del producto" se encuentra su estado esperado, iteración i debe completarse para la i+1*]

Actividades con foco en el Incremento actual

La actividad "Planificar la iteración" tiene como objetivo definir, negociar, acordar y planificar el alcance de la iteración (objetivo, backlog y plan de la iteración, y definición de hecho). Para lograrlo, consta de dos instancias: (i) acordar el objetivo, el backlog de la iteración con el dueño del producto, y revisar y validar la definición de hecho; (ii) el equipo basándose en (i) define y documenta el plan y backlog de la iteración que refleje lo acordado y las tareas que se deben realizar para el logro del objetivo de la iteración y del proyecto. O sea, contar con un backlog de la iteración que tenga responsables asignados a las tareas, estimación ajustada, y las tareas de seguimiento vinculadas a las estrategias y *roadmaps* de las disciplinas, y contar con un cronograma de ejecución. [*única vez, primer actividad de la iteración*]

La actividad "Reunión de sincronización" tiene como objetivo sincronizar y hacer una puesta a punto del trabajo que viene realizando el equipo desde la última reunión de sincronización y el que piensa realizar hasta la siguiente, para conocer el estado actual, evaluar el progreso en los artefactos y resultados, conocer las dificultades a las que se está enfrentando el equipo y disparar acciones necesarias para el logro del objetivo de la iteración. Es importante utilizar y actualizar los artefactos de seguimiento —objetivo, backlog y plan de la iteración, tablero de seguimiento, y gráficos del trabajo pendiente y de las horas de trabajo pendiente de la iteración— y utilizar los artefactos generados —incremento y microincrementos, otros generados o evolucionados durante la iteración y resultados obtenidos—. Previo a la reunión, de forma individual cada uno debe registrar el esfuerzo en los ítems, evaluar el progreso, identificar dificultades o riesgos. [*recurrente, 2 o 3 veces semanales espaciadas equitativamente*]

La actividad "Diseñar el incremento (artefactos de software)" incluye definir, evaluar y actualizar la arquitectura —componentes, relaciones e interfaces—, el conjunto de tecnologías y el diseño de la interacción del usuario. Se debe considerar el backlog de la iteración y la evolución de los artefactos —conceptualización técnica, conceptualización del diseño de la interacción, in-

cremento (de la iteración anterior, i-1) y los *trade off sliders*—. Esto es para que el incremento resultante satisfaga los requerimientos y restricciones, los microincrementos sean fácilmente integrables, y se evolucione la arquitectura, el diseño y la interfaz de usuario. [*única vez, antes de comenzar a construir el incremento de la iteración*]

La actividad "Construir ítem del backlog de la iteración (artefactos de software)" tiene como objetivo construir (análisis, diseño, implementación, revisiones y pruebas unitarias) un microincremento que implementa un ítem del backlog de la iteración (IBI) -de tipo historias de usuario, escenarios de uso o IBI de mejora- e integrarlo al incremento de la iteración que corresponda al software (o sistema). Para iniciar esta actividad es importante contar con el diseño del incremento. Se puede ejecutar varias instancias de la actividad donde la suma de los resultados generan el microincremento. Es importante que se utilicen y actualicen los artefactos de seguimiento. [*repetitiva, hasta que todos los microincrementos estén integrados, se ejecuta durante toda la iteración*]

La actividad "Ejecutar ítem del backlog de la iteración (no artefactos de software)" tiene como objetivo ejecutar las tareas vinculadas a un ítem del backlog de la iteración que no están asociados (de forma directa) a la construcción del incremento de software (o sistema) de la iteración con el fin de generar un artefacto u obtener un resultado, o aportar a un incremento (que no sea de software). Esta actividad se puede ejecutar en varias instancias hasta finalizar la ejecución del ítem, es importante que se utilicen y actualicen los artefactos de seguimiento. [*repetitiva, se ejecuta durante toda la iteración*]

La actividad "Evaluar calidad del incremento" (revisión y pruebas) tiene como objetivo evaluar, ajustar y mejorar la calidad del incremento de la iteración acorde a los criterios de aceptación (generales y de cada ítem del backlog de la iteración) y el alcance de la iteración. Evaluar la calidad del incremento implica evaluar todo artefacto y resultado que va a integrar ese incremento, y el propio incremento. Para lograrlo, consta de dos partes: (i) una donde el responsable de calidad y los testers planifican la evaluación, evalúan y reportan, y (ii) en caso de ser necesario realizar ajustes correspondientes por parte del rol implicado en el cambio. Se espera que se utilice y actualice el registro de incidentes, el incremento en su estado actual o el incremento de la iteración (en su versión final) -este producto es el resultado de la última ejecución de la actividad antes de la revisión- y los artefactos de seguimiento según corresponda [*repetitiva, se ejecuta luego de cada integración y una final cuando el*

incremento este finalizado]

La actividad "Revisar el incremento" tiene como objetivo revisar de forma conjunta con el dueño del producto el incremento de la iteración para recibir retroalimentación sobre el incremento, validar si se cumplió el alcance, objetivo y backlog de la iteración acorde a los criterios de aceptación, y la definición de hecho. Se identifican cambios que puedan impactar en las siguientes iteraciones, particularmente en la siguiente inmediata (i+1) lo cual se debe actualizar en el backlog del producto. La revisión consta de tres partes: (i) preparar la presentación del incremento, (ii) la propia revisión, y (iii) reportar la evolución que se tuvo con el incremento respecto al proyecto. Se debe actualizar el tablero de seguimiento de la iteración reflejando la aceptación (o no) de los ítems y la velocidad del equipo. [*única vez, sobre el final de la iteración*]

La actividad "Evaluar la iteración" (retrospectiva y evaluación general de la iteración) tiene dos objetivos donde el equipo: (i) evalúa lo logrado en la iteración considerando el incremento de la iteración, el esfuerzo incurrido, el avance real obtenido, y la calidad del proceso y producto; (ii) se debe inspeccionar a si mismo, realizando una retrospectiva acerca de lo que sucedió en la iteración (qué funcionó y qué no). Se debe utilizar como evidencia los distintos artefactos de seguimiento como por ejemplo las métricas de la iteración, y generar un informe y el plan de mejora de la iteración. [*única vez, última actividad de la iteración*]

Actividades con foco en el incremento anterior

Durante "Revisar y ejecutar pruebas de aceptación de artefactos y resultados" el dueño del producto (u otros interesados) con el apoyo del equipo deben realizar revisiones y pruebas de aceptación de usuarios para validar que el incremento de la iteración inmediata anterior (i-1) cumple con sus expectativas. Se debe realizar el reporte de las incidencias encontradas (defectos o mejoras). [*única vez, en cualquier momento de la iteración*]

Actividades con foco en el ciclo de vida del proyecto

La actividad "Implantar el incremento" (según plan de liberaciones) tiene como objetivo realizar la implantación del incremento de software y la entrega del incremento (no software) al dueño del producto que corresponden a lo generado en la iteración inmediata anterior (i-1) acorde a lo establecido en el plan de liberación, a través del proceso y el ambiente definido para tal fin.

[*única vez, se ejecuta acorde a lo establecido en el plan de liberación*]

La actividad "Evaluar desvíos y ajustar: estrategias y *roadmaps* - riesgos" tiene como objetivo evaluar (progreso, cumplimiento de objetivos y desvíos), evolucionar y ajustar: las estrategias y hojas de ruta, y otros artefactos (por ejemplo los acuerdos de equipo) y recursos de soporte definidos. Se debe utilizar y actualizar las estrategias y hojas de ruta de la gestión del proyecto, calidad y verificación y la gestión de la configuración, y todos los artefactos de seguimiento y reporte. Esta actividad presenta un objetivo particular al final de la segunda o comienzo de la tercer iteración: debe negociarse y acordarse el alcance del producto mínimo viable con el dueño del producto. [*única vez, en cualquier momento*]

Mediante "Registrar, evaluar progreso y generar informes de avance" se espera generar evidencia y conocer el estado general de lo que está ocurriendo en el proyecto (considerando todas las líneas de trabajo), y de las estrategias y hojas de ruta definidas. La finalidad de ello es medir, evaluar, adaptar, mejorar, y promover la colaboración y circular el conocimiento entre el equipo y el *coach*. Se espera que se hagan ajustes y se generen informes de avance teniendo en cuenta los distintos niveles de planificación y evaluación de progreso (ciclo de vida del proyecto y el ciclo de la iteración) utilizando los principales artefactos para seguimiento y reporte. Se compone de varias partes: (i) registro de esfuerzo, (ii) evaluar el progreso, ajustar y generar informes de la iteración y (iii) evaluar el progreso, ajustar y generar informes del proyecto. [*recurrente, (i) diario - (ii) semanal - (iii) final de la iteración*]

La actividad "Reunión con el *coach*" es la misma definida en la fase inicial. [*recurrente, todas las semanas*]

La actividad "Evaluación de fin de fase"[1] es análoga a la que se ejecuta en la fase inicial. [*única vez, última semana de la fase*]

5.3.3. Fase de construcción, vínculos de artefactos

En la figura 5.9 se puede observar una vista global de las relaciones entre los principales artefactos de seguimiento y reporte, como estos van evolucionado y las interacciones con las distintas actividades del proceso. La fase de construcción inicia con un backlog del producto (bp) compuesto por ítems del backlog del producto (IBP) en distintos estados de refinamiento, algunos de

[1] No se muestra en la imagen ya que no pertenece a la ejecución de una iteración

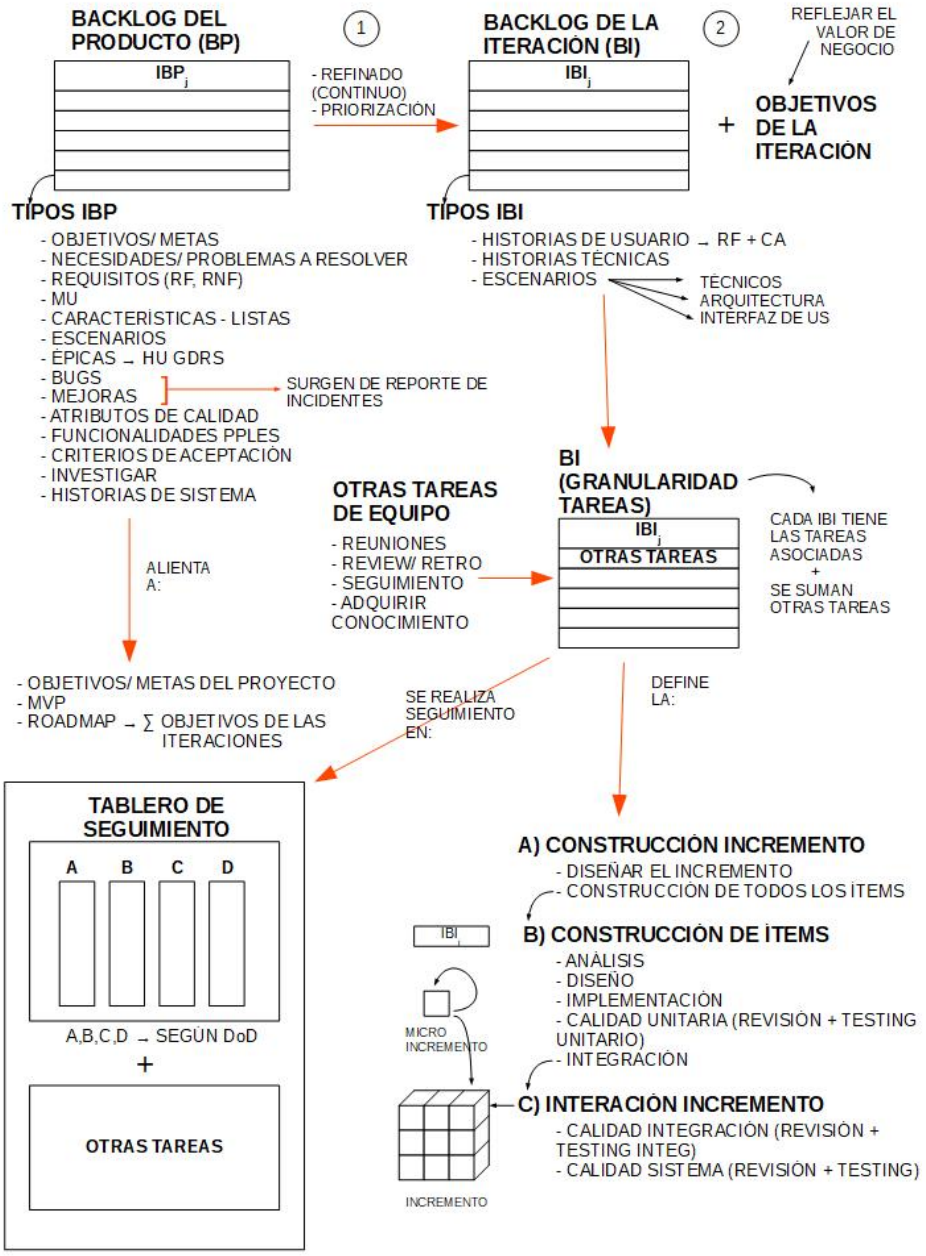

Figura 5.9: PISAgile: vista global de la fase de construcción

ellos son objetivos generales y otros son historias de usuario ya refinadas. Cada IBP tiene asociado un tipo de IBP para dar una pauta del nivel de refinamiento

y si están asociados a un producto de software (sistema) u otro artefacto. El backlog del producto alimenta los objetivos y metas del proyecto, sirve como base para comenzar a definir un producto mínimo viable e ir definiendo una hoja de ruta para los objetivos de las iteraciones. El bp tiene un refinamiento y priorización continua. Esa priorización se toma como base para generar en cada iteración el backlog de la iteración (BI) y los objetivos de la iteración. Entre ambos artefactos se complementa la visión del negocio brindada por el dueño del producto para la iteración. Los IBI ya se encuentran refinados y deben estar prontos para poder construirse, en este caso los tipos de IBI son tres: historias de usuario, historias técnicas y escenarios.

Hasta el paso 2 de la figura el backlog de la iteración (BI) comprende únicamente la visión de los artefactos que tienen valor para el dueño del producto. Sin embargo, el equipo debe realizar otras tareas durante la iteración: actividades, reuniones, adquirir conocimiento, etc. Por lo tanto se toman los IBI que se tenían en el BI (paso 2), se le asocian tareas concretas a realizar teniendo en cuenta la definción de hecho, y se agregan estas otras tareas identificadas. Esto hace que en el paso 3 se tenga un BI donde cada IBI tiene asociadas las tareas para su cumplimiento, "BI (granularidad tareas)". El "BI (granularidad tareas)" define la construcción del incremento de la iteración (paso 4A) que incluye el diseño del incremento y la construcción de todos los ítems que lo componen. Luego la construcción de cada ítem (paso 4B) -incluye el análisis, diseño, implementación, revisión y testing unitario-, da por resultado un micro-incremento el cual es integrado (paso 4C) pasando a ser parte del incremento. La integración del incremento incluye la revisión y el testing a nivel de integración, y de sistema. Para realizar seguimiento al avance del BI se cuenta con un tablero de seguimiento de la iteración. En el tablero de seguimiento se cuenta con una sección para hacer el seguimiento de los IBI que están comprendidos por la definición de hecho y otra que contempla las otras tareas realizadas por el equipo.

5.3.4. Fase de transición

En la fase de transición se realiza el traspaso de los artefactos; tanto asociados al producto de software como otros artefactos generados. Los artefactos de esta fase se pueden visualizar en el Anexo 2 *Fases y sus artefactos*

En la figura 5.10 se presentan las actividades que se ejecutarán en la fase

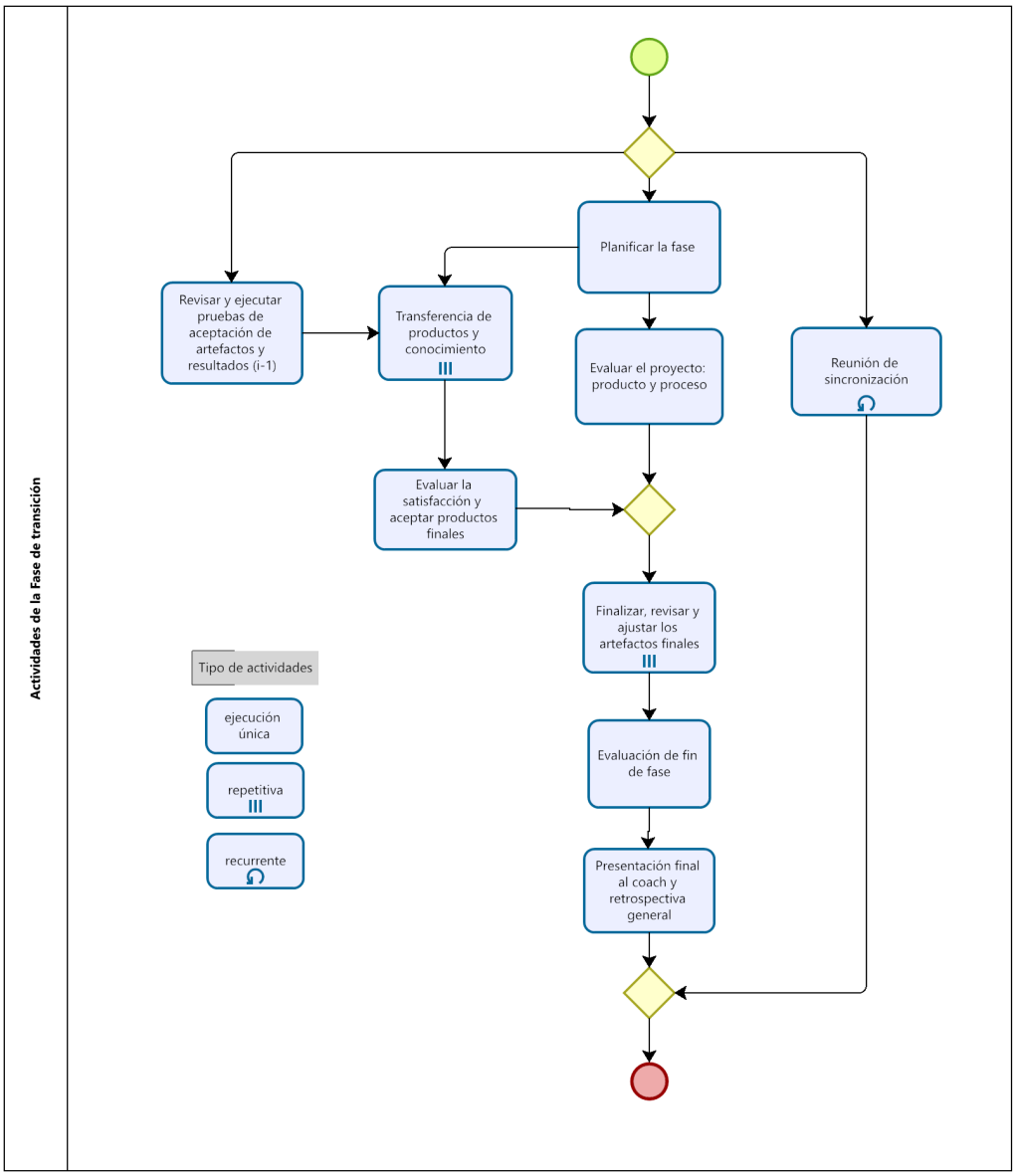

Figura 5.10: PISAgile: actividades de la fase de transición

de transición y pueden ser recurrentes, repetitivas o de ejecución única dentro de la fase. Las principales actividades de esta fase son aquellas con foco en la evaluación y reflexión sobre el proyecto, el proceso, y los productos y resultados obtenidos. También hay actividades con foco en asegurar la calidad y completitud final de los principales productos involucrados en el proyecto, y otras actividades de transferencia de conocimiento del producto acordadas con el dueño del producto. Dado que esta fase tiene una semana de duración, no hay fijada una actividad "Reunión con el *coach*". Inicialmente esa instancia se

suple por "Presentación final al *coach* y retrospectiva del curso", sin embargo puede ser necesario agregarla. A continuación, se describen las actividades.

La actividad "Planificar la fase" tiene como objetivo planificar, negociar y acordar el alcance de la fase de transición (backlog de la iteración, el plan de fase de transición, y el plan y cronograma de transferencia). Esta actividad consta de tres partes: reunión de planificación de la fase donde el equipo estima el esfuerzo, acordar plan y cronograma de transferencia con el dueño del producto, y elaborar el plan de la fase de transición que refleje lo acordado y la asignación de responsabilidades. [*única vez, primera actividad de la fase*]

La actividad "Reunión de sincronización" ya fue descrita en la fase de construcción.[*recurrente, 2 o 3 veces espaciadas equitativamente*]

La actividad "Transferencia de productos y conocimiento" tiene como objetivo transferir al cliente, dueño del producto, usuarios u otros interesados los resultados obtenidos, la versión final del producto de software (o sistema) y todo otro producto, y conocimiento necesario para su uso, operación, administración y evolución. [*repetitiva, hasta finalizar transferencia*]

La actividad "Evaluar la satisfacción y aceptar los productos finales" tiene como objetivo evaluar mediante una encuesta (y otros métodos) la satisfacción del dueño del producto y otros interesados acerca de los productos finales y del proyecto en general, obtener la aceptación final de estos, y conocer el estado general de defectos y propuestas de mejoras. [*única vez*]

La actividad "Evaluar el proyecto: producto y proceso" (retrospectiva y evaluación general) consta de dos objetivos. El primero es evaluar el resultado final logrado en el proyecto y cómo fue evolucionando el trabajo durante el mismo. Se debe evaluar el producto de software (o sistema) y el proceso. El segundo consiste en que el equipo reflexione acerca de lo acontecido durante el proyecto (qué funcionó, qué no y qué se podría haber hecho) considerando las distintas "estrategias y hojas de ruta" y generar las lecciones aprendidas del proyecto. Esto se debe plasmar en la estrategia y *roadmap* de la gestión del proyecto, la estrategia y *roadmap* de calidad y verificación, la estrategia y *roadmap* de gestión de la configuración, el documento de riesgos, las lecciones aprendidas, la encuesta de satisfacción al cliente, el informe final del proyecto, el informe final de calidad y verificación, el informe final de la gestión de la configuración, y otros artefactos generados. Se deben utilizar los principales artefactos de seguimiento y reporte, y artefactos y productos generados. [*única vez*]

La actividad “Finalizar, revisar y ajustar los artefactos finales” tiene como objetivo finalizar, revisar y ajustar los artefactos finales asociados al proceso y proyecto para mejorar su calidad, incluyendo su completitud y consistencia. Asegurar que los mismos reflejen el desarrollo del proyecto y que sean comprensibles para quienes no hayan formado parte del proyecto, ya que pasarán a ser parte de la memoria organizacional del curso. Se toma en cuenta la versión final de los artefactos generadas luego de la última ejecución de la actividad “Evaluar el proyecto; producto y proceso”. [*repetitiva, se ejecuta a medida que haya versiones finales de artefactos*]

La actividad “Evaluación de fin de fase” es análoga a la que se ejecuta en la fase de construcción, la diferencia se debe a que en este caso al ser la última fase no se evalúa un pasaje de fase sino la finalización del proyecto. [*única vez*]

La actividad “Presentación final al *coach* y retrospectiva general” consta de dos objetivos; (i) realizar una presentación al *coach* sobre lo que fue el desarrollo del proyecto, y (ii) reflexionar en conjunto acerca de la asignatura y el proceso aplicado para obtener propuestas de mejoras a la asignatura y propuestas de mejoras al proceso. La presentación debe contemplar aspectos del proceso y del proyecto, y es una instancia para que el equipo reciba retroalimentación para la presentación final evaluativa del curso. Se debe tener en cuenta que, previo a realizar la presentación y la reflexión, el equipo debe realizar la preparación de dicha presentación [*única vez, preparación durante toda la iteración y presentación al final de la iteración*]

Actividades con foco en el incremento anterior

La actividad “Revisar y ejecutar pruebas de aceptación de artefactos y resultados“ ya fue descrita en la fase de construcción. [*única vez, los primeros días de la fase*]

5.4. Creación de PISAgile: aspectos incluidos de otros procesos

El proceso PISAgile es un proceso ágil que cumple con el Manifiesto por el desarrollo ágil de software, y sus 12 principios[1]. En PISAgile se integran distintos procesos con adaptaciones para que pueda ser aplicado dentro de un curso

[1] https://agilemanifesto.org/iso/es/manifesto.html

de grado. Para su definición se consideraron aspectos de Scrum, del Proceso Modularizado Unificado y Medible (MUM), Agile unified process (AUP), eXtreme Programing (XP), el Team Software Process (TSP), Design Thinking, Kanban, y prácticas generales de la ingeniería de software. Cada uno de estos procesos (o marcos de trabajo) aportaron a distintos aspectos de PISAgile.

Scrum (Schwaber y Sutherland, 2020) fue el marco de trabajo que se tomó como base para definir aspectos vinculados a la gestión del desarrollo. Scrum parte desde el primer sprint realizando la construcción del producto y todos los sprints son iguales, no propone nada específico para el abordaje inicial de un proyecto, su cierre, y cómo abordar varias de las disciplinas de la ingeniería de software.

Al contar con un equipo de trabajo que recién se conforma y solo con un breve resumen del proyecto, se considera necesario realizar algunas adaptaciones y agregados teniendo en cuenta: que se está en un contexto de enseñanza, la falta de experiencia en estos proyectos por parte de los estudiantes y la importancia del aprendizaje holístico de la aplicación de la ingeniería de software en un proyecto.

Principalmente se adicionan actividades que tienen que ver con un abordaje inicial del proyecto -generar una visión global y conceptualizar las necesidades el cliente-, incluir el abordaje de las distintas disciplinas, y con el cierre del proyecto -realizar una transferencia de producto, y conocimiento definitiva al cliente y al curso-. Es por este motivo que se tomó como base el concepto de fases y sus objetivos, el abordaje de algunas disciplinas básicas y de soporte, y roles de otros procesos como lo son el Agile Unified Process (AUP)[1], el Proceso Modularizado Unificado y Medible (MUM), y el Team Software Process (TSP) (W. S. Humphrey, 2005). Esto fue complementado con algún aspecto técnico vinculado a la construcción del producto propuesto en eXtreme Programing (XP)[2], prácticas vinculadas a la conceptualización del problema y la necesidad propuestas por Design Thinking, y un tablero Kanban para visualizar el seguimiento de las tareas.

La creación de un proceso que toma algunos aspectos de otros llevó a que existan determinadas similitudes y diferencias entre los procesos (o marcos de trabajo) y PISAgile. A continuación se resaltan los principales aspectos que se adoptaron para cada componente de PISAgile y algunas de las principales

[1] http://www.ambysoft.com/unifiedprocess/agileUP.html
[2] http://www.extremeprogramming.org/

diferencias en su adopción.

Uno de los principales elementos que se tomaron como base de Scrum y se adaptaron para PISAgile son: los pilares y principios para definir principios y valores de PISAgile. Respecto a los pilares y principios de Scrum, algunos se conservaron como tales en PISAgile, otros se les adaptó su nivel de detalle y alcance, y otros no fueron reflejados de forma concreta sino que se particionaron en distintos principios y valores de PISAgile. Ambos procesos comparten la importancia de que los valores se adopten y desarrollen de forma individual y grupal para poder llevar a cabo el proceso.

En PISAgile se adoptaron todos los pilares de Scrum: la transparencia se definió como valor y se modificó su alcance para considerarla respecto a los integrantes de equipo; la inspección no está presente como tal sino que se refleja en otros principios de forma más concreta; la adaptabilidad del proceso en PISAgile se define de forma acotada, ya que se pueden realizar ajustes pero deben conservarse sus principales características del proceso.

El único valor de Scrum no adoptado por PISAgile es el coraje, y cambia la especificación de la apertura en PISAgile ya que se define por medio de tener una comunicación continua y abierta, e incluye tanto a individuos como al entorno que se genera, mientras que Scrum deja libre la interpretación. A estos se adicionaron otros valores y principios teniendo en cuenta los objetivos de aprendizaje del curso, algunos de ellos se complementaron con los valores de XP. Estos son la comunicación y la retroalimentación.

Para poder especificar el proceso se partió por definir una arquitectura base en la cual se pudiera organizar y visualizar los componentes de PISAgile. Los valores y principios son el sustento que le da sentido a los componentes del proceso, por lo tanto son los cimientos sobre los cuales van a estar el resto de los componentes. Para organizarlos se adoptó del proceso RUP (también tomada por el MUM), la idea de contar con dos dimensiones -dinámica y estática- donde se clasifican y agrupan los otros componentes y subcomponentes de PISAgile.

Como se mencionó anteriormente Scrum no hace distinción ni brinda pautas acerca del abordaje de varias de las distintas disciplinas de la ingeniería de software dejando libertad en su aplicación. Dado que los estudiantes están en un proceso de formación es necesario que cuenten con una guía más explicita que contemple una visión holística del abordaje de las distintas disciplinas para que entiendan cómo aporta cada una al proyecto.

Las disciplinas básicas y de soporte fueron adoptadas desde el proceso MUM, sus definiciones y alcance se complementaron con lo definido en el SWEBOK (Bourque y Fairley, 2014), y se realizaron modificaciones para que colaboren en el desarrollo de los valores y principios de PISAgile. A nivel general las disciplinas tomadas de MUM mantienen su funcionalidad aunque en PISAgile se definen según su propósito general (tomando como base el SWEBOK) en lugar de definirlas instanciadas al proceso tal como sucede en el MUM. También se realizan algunos ajustes en sus alcances. Las disciplinas que se agregan son:

- Diseño/interacción de usuario la cual en el MUM se aborda como parte en la disciplina requerimientos mientras que en SWEBOK es parte del Diseño. El alcance en ambos casos está más centrado en lo que es la interfaz de usuario mientras que en PISAgile se apunta también a la experiencia del usuario al interactuar con el sistema y cómo este lo hace sentir.
- La colaboración como disciplina más amplia que la Comunicación propuesta en el MUM para apuntar a todos los aspectos vinculados a fortalecer el trabajo en equipo, la integración, resolución de conflictos, etc. y no solo a la comunicación y sus métodos.
- La gestión del conocimiento que si bien no aparece en el MUM ni en el SWEBOK es una parte importante en los proyectos que muchas veces se deja de lado.
- En otros casos como la gestión de la calidad propuesta en el MUM se particionó en calidad del producto y verificación -unificándose con validación-, y calidad del proceso. Este cambio se debe a, poder focalizar en todo lo que tiene que ver con la mejora del producto y por experiencias anteriores donde los estudiantes tienden a unificar el tratamiento de la calidad del producto y la verificación.

En cuanto a los roles, scrum define un equipo multifuncional y asume que los miembros en su conjunto (sin distinción) tienen todas las habilidades necesarias para cumplir con los objetivos y generar valor durante el sprint, y define tres roles. En PISAgile se tomaron estos roles -haciendo ajustes de alcance debido al contexto-:

- El rol de scrum master por estar en un ambiente de aprendizaje se encuentra dividido entre el facilitador y el *coach* -es responsable también

porque el equipo aprenda y entienda el proceso-.

- El abordaje de la calidad queda distribuido en varios roles y no solo en el desarrollador agregando otras visiones.

Se agregaron algunos roles con responsabilidades específicas como son los responsables de las disciplinas, *testers*, *coach*, y otros roles vinculados al cliente. Los roles de responsabilidad se definieron con el objetivo de que el equipo pueda organizarse mejor teniendo claras sus funciones de forma más específica, y generar un sentido de pertenencia al grupo y su forma de trabajo.

Estos roles fueron tomados del MUM -reduciendo su número, y ajustando su definición y nombre contemplando las características de PISAgile-, comparten en algunos casos algunas definiciones de XP y en el caso de los responsables se tomó el concepto de TSP respecto a estos roles y su definición. Si bien se ajustaron algunas responabilidades de TSP, en escencia se toman todos los gestores exceptuando el rol de líder de equipo, por más que en cierta forma, parte de su alcance sea cubierto por el *coach*.

Por lo expresado con anterioridad respecto al inicio y cierre de un proyecto, entendemos que es importante contar con fases que permitan focalizar el trabajo en distintos estados de evolución del proyecto. Para definir las fases y sus objetivos se tomaron como referencia MUM, AUP, y se complementaron con algunos aspectos de XP.

De MUM y de AUP se tomaron los objetivos principales de la fase inicial, la principal diferencia es que no se estiman costos y cronograma sino que se elabora un plan de *release* de alto nivel teniendo en cuenta los objetivos del negocio similar al definido en XP, y no se hace un estudio de la viabilidad del proyecto.

Tanto AUP como MUM tienen una fase de elaboración la cual no se consideró adoptar por PISAgile ya que se busca un desarrollo más evolutivo desde un principio, pero sí se agrega a la fase inicial la identificación de riesgos técnicos y comenzar a delinear una arquitectura base para la solución -objetivos de la fase de elaboración-.

En la fase de transición los objetivos también son similares, aunque en PISAgile se hace mayor énfasis en la evaluación y reflexión sobre los resultados obtenidos que el que se hace en AUP. Esto lo adopta de forma similar a la propuesta de MUM. Por otro lado, en AUP se plantea que recién en la fase de transición se realiza un *deploy* del sistema en un entorno productivo. En

PISAgile puede suceder durante la fase de construcción ya que depende de lo acordado con el dueño del producto, y tampoco se especifica un recomienzo del proceso con una nueva *release* como se hace en AUP, ya que si se definen más de una *release* estas se van liberando durante la fase de construcción.

La fase de construcción de PISAgile, las iteraciones y actividades están fuertemente basadas en Scrum. Las iteraciones se definen de forma análoga a los sprints donde la principal diferencia se encuentra en que todos los eventos de Scrum contribuyen al incremento que se genera en el sprint en el cual se está llevando a cabo el evento, mientas que las actividades que se ejecutan en una iteración de PISAgile pueden contribuir al incremento de la iteración, incrementos futuros y al incremento anterior.

En cuanto a las otras dos fases se tomaron actividades de MUM y de AUP adaptándolas y agregando otras acordes a las características de PISAgile, y se complementaron para el caso de la fase inicial con prácticas de Design thinking que permitan comprender las necesidades y el problema del cliente. El agregado de las fases hace que PISAgile tenga una gestión inicial del proyecto y una de cierre de proyecto, mientras que Scrum no distingue una gestión particular que dependa de la ubicación temporal.

En lo que refieren a las actividades de PISAgile para la fase de construcción vinculadas a la gestión del desarrollo e inspección de los artefactos de la iteración, se tomaron como base los eventos de Scrum. A estos se les realizaron algunas adaptaciones:

- Se definieron actividades que incluyan los eventos pero con algún objetivo adicional o se particionaron los eventos teniendo en cuenta el contexto.
- Se agregaron directrices acerca de cómo llevar a cabo las actividades.
- Se agregó la visión de las distintas disciplinas.
- Se enfatiza en la necesidad de contar con evidencia para poder realizar la inspección, evaluación y adaptación del proceso.

Teniendo en cuenta que Scrum no da lineamientos o pautas concretas referentes al abordaje de los requisitos, del diseño, la implementación y pruebas, siendo estas actividades indispensables en un proceso de desarrollo, es que se tomaron algunos elementos de XP para completar estos aspectos. XP plantea algunas reglas vinculadas a actividades como la planificación, gestión, diseño, codificación y pruebas, y proporciona las relaciones entre una y otra regla para lograr los objetivos. En PISAgile no se incluyeron como reglas sino como

actividades, prácticas vinculadas a las mismas o en las definiciones que tienen que tomar en el marco del proceso.

En PISAgile se conservan todos los artefactos de Scrum. En el caso de PISAgile el backlog del producto no es la único artefacto donde se especifica el trabajo que debe realizar el equipo ya que hay otras tareas que no se especifican allí por no ser parte del objetivo de la iteración, pero sí son reflejadas en el backlog de la iteración.

La noción manejada de incremento puede ser distinta dependiendo de la interpretación que se haga de Scrum -la guía no es del todo específica-. En PISAgile se compone de la unión de distintos tipos de artefactos, los cuales algunos son de software y otros no lo son.

En el caso de Scrum la terminación del incremento está asociada a la definición de hecho y en general la misma detalla criterios vinculados al producto de software. A estos artefactos se les agrega un tablero de seguimiento de las tareas de la iteración como un artefacto más de transparencia basado en kanban, y otros necesarios para la aplicación de PISAgile, entre ellos la definición de las estrategias de las distintas disciplinas donde para definirlos se tomaron los planes propuestos en el MUM.

Las distintas adaptaciones y agregados realizados se consideraron necesarios para que el proceso pueda ser aplicado en un entorno de enseñanza. Esto es por los motivos ya expresados y a su vez para que los estudiantes al estar en formación puedan contar con un proceso que les de instrucciones más detalladas y proporcione una visión holística del abordaje de las distintas disciplinas de la IS en un proyecto.

También tuvimos en cuenta la dedicación horaria -la cual es inferior a los proyectos de desarrollo de software en la industria- y su distribución en los días de la semana que en general no es regular por parte de los estudiantes y los clientes del PIS, lo que dificulta particularmente los comienzos y cierres de iteraciones.

Para la especificación de requisitos se incluyó en PISAgile el uso de historias de usuario (HU), las diferencias más sustantivas con respecto a la propuesta de XP se da a nivel conceptual y del detalle de las funcionalidades al momento de elaborar el plan de releases. En el caso de PISAgile es una aproximación inicial y a diferencia de XP se hace con los ítems del backlog del producto, el cual puede contener ítems que son HU y otros que no lo son.

En PISAgile no hay una instancia de estimación específica y está más orien-

tada a definiciones de prioridades por parte del dueño del producto proporcionando una versión inicial temprana pero con una incertidumbre más alta que en XP ya que no se realiza una estimación.

En cuanto al diseño de software XP y PISAgile comparten la idea de darle relevancia al mismo, pero en PISAgile no se hace énfasis explícito en que sea un diseño simple. Una diferencia importante entre ambos procesos, es que XP se apoya fuertemente en la refactorización para mantener el diseño con ciertos criterios de calidad, mientras que PISAgile hace énfasis en el análisis previo y en un diseño consensuado teniendo siempre en mente la vista global del producto. Al igual que en XP, PISAgile adoptó la inclusión de *spikes* para reducir riesgos técnicos, o entender mejor algunas HU y su estimación.

Respecto a la codificación en PISAgile no se adoptaron reglas restrictivas como lo plantea XP, sino que se definen como parte de las estrategias, como lo es la definición de estándares de codificación, la ejecución de pruebas unitarias -pero no se define que su creación deba ser previo a escribir el código-, la integración continua -se deja a definir por el proceso de integración en la gestión de la configuración-, y la presencia del cliente -que es menor a la propuesta en XP-.

Las actividades vinculadas a la pruebas se hacen con una definición más amplia que XP incluyendo la calidad y difiere la implementación de algunos niveles de pruebas.

Las pruebas de aceptación en XP son utilizadas para aceptar las HU de cada iteración, se ejecutan en la propia iteración a la que pertenece la HU y no se considera completa hasta no pasar dichas pruebas. En PISAgile la aceptación de las HU y las pruebas de aceptación son implementadas de forma distinta. Las pruebas de aceptación no son condicionante en la aceptación de las HU -la aceptación de las HU se realiza por el dueño del producto en la actividad "Revisar el incremento" teniendo en cuenta los criterios de aceptación-. Las pruebas de aceptación son ejecutadas en la iteración siguiente.

En cuanto a las pruebas unitarias, en XP se construyen antes de escribir el código del software a construir, mientras que en PISAgile se indica que deben ejecutarse previo a la integración del código pero no en el momento en que deben diseñarse y crearse. La automatización depende de lo definido en la Estrategia de calidad y verificación.

Al reportar defectos, PISAgile no establece que deban realizarse pruebas de aceptación para ayudar a describir el problema a los desarrolladores.

Capítulo 6

Evaluación de PISAgile

La evaluación del proceso de desarrollo de software PISAgile se realizó aplicándolo directamente en el contexto para el cual fue creado: el curso de grado Proyecto de ingeniería de software (PIS) de la Facultad de Ingeniería de la Universidad de la República. Los datos que se analizan en este capítulo surgen del curso del año 2020. Estos datos se recolectaron mediante encuestas individuales y anónimas, las cuales fueron respondidas por los distintos actores que intervienen en el curso: estudiantes, docentes y clientes.

En el marco de este trabajo, la evaluación de PISAgile es presentada desde una mirada parcial ya que contempla de forma exclusiva la perspectiva de los estudiantes, considerando la percepción de estos sobre distintos aspectos a evaluar, principalmente en las habilidades blandas.

Para el proceso de diseño, implementación y recolección de datos de las encuestas se utilizó la herramienta *surveymonkey*[1]. Para el análisis de las respuestas recolectadas se utilizó la propia herramienta y los datos extraídos en crudo de cada participante de forma individual.

PISAgile fue aplicado por tres de los diez grupos que participaron en la edición 2020 del PIS con proyectos, docentes y clientes distintos.

En este capítulo se presentan: los objetivos y aspectos a evaluar de la aplicación del proceso; la caracterización de los proyectos y cómo se realizó la recolección de los datos; los resultados obtenidos y reflexiones sobre estos respondiendo las preguntas de investigación planteadas; las limitaciones y comentarios finales de la evaluación.

[1]Solución tecnológica para el diseño, implementación, recolección y análisis de encuestas por medio de la web. https://www.surveymonkey.com/

6.1. Objetivos y aspectos a evaluar

Con este estudio se busca evaluar el proceso propuesto PISAgile desde la mirada de los tres actores que participan en el proceso (estudiantes, docentes y clientes). Sin embargo, por aspectos vinculados al alcance de la tesis el análisis se realiza únicamente desde la perspectiva de los estudiantes. Se quiere conocer la percepción de estos acerca de si PISAgile es de utilidad para los fines buscados con su creación: colaborar en el aprendizaje de los estudiantes y en el desarrollo de software.

Se definieron dos objetivos principales para la evaluación de PISAgile. El primer objetivo es evaluar si mediante la aplicación de PISAgile se cumplieron los objetivos del curso: i) "Afirmar y profundizar los conocimientos de ingeniería de software, contrastarlos con su aplicación práctica e integrarlos con otros conocimientos de unidades curriculares previas" (InCo, 2017), y ii) el desarrollo por parte de los estudiantes tanto de habilidades blandas (ej: responsabilidad) como habilidades técnicas (ej: técnicas para obtención de requisitos) necesarias para la aplicación práctica. El segundo objetivo es conocer si PISAgile sirve para que grupos de estudiantes del PIS logren construir productos de software de calidad en los plazos definidos.

Para la selección de las habilidades técnicas a evaluar, se tuvo en cuenta el temario del curso priorizando las vinculadas a las disciplinas básicas del desarrollo de software para acotar el alcance de la evaluación. Para la selección de las habilidades blandas a evaluar, se tomó como referencia un estudio que examina el efecto que tiene un curso *capstone* basado en proyectos en la adquisición de las habilidades blandas necesarias para trabajar en la industria del software por parte de estudiantes en un proyecto real (Khakurel y Porras, 2020). Se seleccionó este artículo porque también presenta una síntesis de distintos estudios realizados para relevar las habilidades blandas requeridas en la industria del software, incluyendo uno realizado con empresas de Uruguay. Además, se selecionaron y agregaron otras habilidades que entendemos resultan interesantes para nuestro contexto.

A continuación se detallan las dieciséis habilidades blandas seleccionadas:

- comunicación del equipo con cliente, *coach* y otros *stakeholders*
- comunicación dentro del equipo de estudiantes
- habilidad personal para establecer un buen relacionamiento con clientes, equipo, *coach*, etc.

- pensamiento crítico y resolución de problemas: capacidad de identificar y razonar de forma crítica sobre los problemas y necesidades del cliente, proponiendo ideas que atiendan estos problemas y necesidades
- trabajo en equipo y colaboración
- habilidad personal para organizar de forma eficaz y eficiente el trabajo
- habilidad para trabajar independientemente
- apertura a los cambios y adaptabilidad
- compromiso y responsabilidad
- liderazgo
- orientación al cliente: refiere a la habilidad de poder entender y considerar al cliente, sus necesidades y expectativas, y actuar en consecuencia de ello
- capacidad para trabajar bajo presión
- manejo de conflictos: capacidad de resolver problemas de relacionamiento que puedan ocurrir en pos de generar una solución pacífica que no dañe el vínculo entre los participantes
- negociación: capacidad a lo largo del proyecto de hacer acuerdos con el cliente y otros *stakeholders*
- capacidad de expresión en la escritura y reporte
- capacidad de expresión oral y presentación

En el caso del aprendizaje de las habilidades técnicas y su aplicación, se seleccionaron:

- abordaje de los requisitos
- calidad, verificación y validación
- arquitectura de software

6.2. Diseño de la encuesta y recolección de datos

Los proyectos sobre los cuales se hizo la evaluación tuvieron una duración de trece semanas, tienen como objetivo principal la construcción de software, cada uno fue presentado por un único cliente y tuvo asignado un único grupo de estudiantes. Uno de los grupos fue conformado por diez y dos grupos por doce integrantes. Para el desarrollo del software se aplicaron distintas tecnologías y

frameworks, los cuales podían no ser conocidos por los estudiantes. La tabla 6.1 presenta los clientes, una muy breve descripción del proyecto, las tecnologías utilizadas y la cantidad de estudiantes del grupo de estudiantes.

Tabla 6.1: Proyectos 2020 que aplicaron PISAgile

Cliente	**Proyecto**	**Tecnologías**	**Equipo**[a]
December Labs	Sistema para validar la experiencia de usuarios finales (UX) frente a un nuevo diseño.	Web, React	10
Effectus software	Red social empresarial para incentivar el involucramiento de los miembros de la empresa.	Móvil, React, React Native, Ruby on rails	12
Neo coast	Facilitar la interacción entre usuarios y voluntarios para efectuar diversas tareas en contexto de COVID-19, ej: compras	Web y Móvil, Ruby, React, React Native	12

[a] Cantidad de estudiantes que participaron del proyecto

Para la selección del método de recolección se tuvo en cuenta que permitiera obtener información acerca de cada aspecto a evaluar, el contexto, el costo-beneficio de su aplicación, y poder tener una visión de los distintos actores. Considerando estos aspectos se optó por realizar una encuesta voluntaria y anónima luego de finalizar el curso. El hecho de realizar una sola encuesta al final del curso se debió a que el método de recolección fue seleccionado luego de comenzado el curso y no se seleccionó otro método como por ejemplo análisis de métricas del proyecto por no contarse con datos procesables de todos los grupos. Se evaluó la posibilidad de que la encuesta fuese obligatoria, sin embargo, esta opción se descartó por considerarse que podría ser un factor que influenciara en las respuestas brindadas por los actores y por consiguiente en la evaluación de la aplicación del proceso. Si bien la evaluación se aplicó solo a PISAgile entendemos que puede servir para evaluar otros procesos.

Se confeccionaron tres encuestas, una para estudiantes, otra para docentes y una tercera para los clientes. Si bien las preguntas entre una y otra encuesta tienen alguna diferencia, comparten los aspectos a evaluar para contar con las perspectivas de los distintos actores que intervinieron, tanto desde un punto de vista cualitativo como cuantitativo. Como se mencionó anteriormente en el marco de esta tesis nos centraremos en conocer la percepción de los estudiantes.

El detalle de las encuestas realizadas y el proceso está disponible en el servidor Zenodo[1], esto hace posible replicar la experiencia.

En la encuesta se consulta a los estudiantes acerca de distintos aspectos, buscando en cada pregunta su reflexión y percepción. Se compone de veinte preguntas donde se incluyen de tipo: texto libre, múltiple opción y de escala. Particularmente, se les pregunta acerca de:

- información general: roles en el proyecto y sobre su experiencia previa.
- percepción del aprendizaje: evolución de su nivel de conocimiento, adquisición de ciertas habilidades blandas y técnicas.
- principios del proceso y su aplicación: colaboración y relacionamiento, métricas, evaluación y reflexión por parte del equipo.
- evaluación de la descripción y contenido del proceso (roles, fases y actividades).

Para consultar acerca de las habilidades blandas, se presentó el listado de las dieciséis habilidades blandas mencionadas anteriormente, y una escala de aplicación y desarrollo para indicar cuál afirmación se adapta mejor a lo que entiende que ocurrió durante el curso para evaluar si PISAgile colaboró en la adquisición o mejora de las mismas. Las afirmaciones que se dieron como opciones de respuesta son:

1. La apliqué en el PIS y entiendo que he logrado incorporar la habilidad blanda debido al PIS
2. La apliqué en el PIS, pero ya la tenía incorporada antes del PIS y he logrado mejorarla
3. La apliqué en el PIS, pero ya la tenía incorporada antes del PIS
4. No la apliqué durante el PIS (no la aprendí o afiancé durante el el curso)
5. No conozco esta habilidad blanda

Con el fin de confirmar la asimilación y comprensión de aquellas habilidades que el estudiante menciona que incorporó o mejoró, se solicitó que reflexione acerca de las situaciones en las que notó su aplicación durante el curso.

Respecto a las habilidades técnicas, se consultó acerca de la percepción que tienen en la evolución en el conocimiento y los resultados obtenidos.

Al no ser una encuesta obligatoria, se optó por la estrategia de incentivar la participación en la misma. Se planteó en varias oportunidades el valor y

[1] https://doi.org/10.5281/zenodo.6730363

la importancia de la respuesta de cada uno, para la evaluación del proceso, el curso, así como retroalimentación para el trabajo con futuros estudiantes. Durante la ejecución de la encuesta se fue monitoreando el avance de la misma y se les solicitó su participación en tres oportunidades en un período de quince días.

6.3. Síntesis y extracción de datos

Teniendo en cuenta que en la encuesta hay preguntas cerradas —se brindan determinadas opciones para su respuesta— y otras abiertas, el proceso de extracción y síntesis fue distinto.

En el caso de las preguntas de carácter cualitativo, se realizó una extracción de los términos claves presentes en cada una de las respuestas, luego se agruparon aquellos conceptos que tienen una misma raíz conceptual. Para algunas de las preguntas se realizó alguna categorización en determinadas áreas temáticas o aspectos particulares a los cuales hacen referencia los términos encontrados, por ejemplo: documentación, enseñanza y proceso. Dadas las características del análisis cualitativo, la asociación de conceptos tiene una importante cuota de interpretación del analista y por lo tanto cierta subjetividad y margen de error de interpretación. Lo ideal en estos casos sería poder validar con el encuestado la interpretación realizada, pero esto no fue posible.

Luego el análisis de cada respuesta se realizó en dos niveles: (i) un nivel grupal donde se contabilizaron número de respuestas, los términos presentes en las respuestas de forma general o en la categoría en la cual se incluyeron, para luego comparar entre las distintas categorías y (ii) a nivel individual estudiando el contenido concreto de cada respuesta en particular.

En el caso de las preguntas cerradas, se contabilizaron las respuestas por opción. Como parte del proceso de síntesis se realizaron algunos cruces entre preguntas (independientemente si son cerradas o no) ya que algunas de ellas son complementarias, entonces adquiere más valor su reflexión conjunta.

La extracción de datos es desde la herramienta de recolección, posteriormente se realizó un análisis exploratorio utilizando la biblioteca Pandas Profiling de Python[1] y luego se utilizó una planilla de cálculo para su procesamiento. Para identificar las respuestas de un mismo estudiante en nuestro análisis, se

[1] https://pypi.org/project/pandas-profiling/

le asignó un número único.

6.4. Resultados y reflexiones

La encuesta de estudiantes tuvo un nivel de adhesión general de un 82 % considerando el total de convocados a realizarla, variando por cada grupo como se puede observar en la tabla 6.2. Teniendo en cuenta que la encuesta no es obligatoria es un muy buen nivel de adhesión.

Dos estudiantes (E14 y E25) realizaron la encuesta de forma parcial, es decir, si bien contestaron las primeras preguntas de información general no siguieron con las sustanciales para nuestro análisis (percepción del aprendizaje, principios del proceso y su aplicación, y evaluación de la descripción y contenido del proceso). Por lo tanto, se excluyen de los resultados y las reflexiones, y el análisis se termina realizando sobre el 76 % de los estudiantes que aplicaron PISAgile en el curso. Hubo un tercer estudiante (E12) que si bien no contestó toda la encuesta, si respondió algunas preguntas que son foco de nuestro estudio y por ende se incluye en el análisis.

Tabla 6.2: Nivel de adhesión de estudiantes a encuesta

Grupo nro	Cursaron[a]	Respondieron[b]	Adhesión (%)
2	10	7	70
3	12	11	92
5	12	10	83
Total	34	28	82

[a]Cantidad de estudiantes del grupo

[b]Cantidad de estudiantes que respondieron la encuesta

El análisis de resultados está guiado por tres preguntas de investigación que permiten reflexionar sobre los principales objetivos del estudio. Las preguntas a responder son:

I ¿El proceso sirve para que se cumplan los objetivos de aprendizaje del curso?

II ¿El proceso sirve para que se construya un producto de software con la calidad esperada?

III ¿Cuáles son las fortalezas o debilidades al momento de comprender o aplicar el proceso?

Cada una de estas preguntas de investigación es respondida desde varias ópticas con las preguntas de la encuesta. Para delimitar la extensión de esta tesis la primera de las preguntas se abordará con mayor profundidad que las otras dos. Se deja como trabajo futuro su estudio en profundidad.

6.4.1. ¿El proceso sirve para que se cumplan los objetivos de aprendizaje del curso?

El curso busca que mediante la ejecución de un proyecto de software guiado y con restricciones análogas a las de la industria, los estudiantes puedan afirmar, reflexionar e integrar sus conocimientos teóricos y prácticos de otros cursos vinculados a la ingeniería de software y el propio PIS (InCo, 2017). Esto incluye aspectos generales, habilidades blandas y técnicas asociados a las diferentes áreas temáticas que preparen a los estudiantes en su práctica profesional.

Los resultados a presentar se dividirán en dos niveles de abstracción. Uno con un abordaje general acerca de la evolución del aprendizaje durante el semestre sin limitar a ningún aspecto o área temática en particular. El otro específico para presentar los resultados del aprendizaje vinculado exclusivamente a determinadas habilidades blandas previamente seleccionadas.

En los proyectos de software la aplicación práctica en entornos reales y la experiencia juega un rol fundamental, por lo que se les consultó a los estudiantes acerca de su experiencia previa[1].

P3:¿Previamente a tu participación en el curso, trabajabas en proyectos vinculados al desarrollo de software? (Sí, No)

Esta pregunta fue respondida por los 28 estudiantes que se adhirieron a la encuesta. De los resultados surge que un 32 % de los estudiantes no ha tenido experiencias laborales previas en proyectos vinculados al desarrollo de software y un 68 % que sí lo tuvo. Por lo cual, en el primer caso tampoco han aplicado ningún tipo de proceso para desarrollo de software directamente en proyectos, ni sus conocimientos en entornos con características y restricciones reales. Con

[1]Las preguntas se presentan con la letra P seguida de un número. La correspondencia de estos números con los de la encuesta en SurveyMonkey se da con un desfasaje de una unidad. O sea, la pregunta identificada en esta tesis como P1 es la número 2 en la encuesta en SurveyMonkey, esto es porque nosotros no consideramos el consentimiento como una pregunta.

esto se presenta un desafío no menor que es lograr el equilibrio para que exista aprendizaje tanto para los que tienen experiencia como para los que no la tienen. Esto refuerza el hecho de que el proceso no solo debe ser útil para desarrollar software de calidad sino que también debe ser un proceso mediante el que los estudiantes puedan adquirir o aplicar conocimientos y habilidades vinculadas al desarrollo de software independientemente de su experiencia.

6.4.1.1. Aprendizaje general durante el curso

Para evaluar el aprendizaje, se incluyeron en la encuesta las siguientes preguntas:

P4:¿Consideras que haber participado de un proyecto de desarrollo de software aplicando el proceso PISAgile te ayudará en tu práctica profesional (actual o futura)? (Sí, No)

P5:Reflexiona y describe brevemente cuáles consideras que fueron tus principales aprendizajes durante el proyecto. (Texto libre)

Los 26 estudiantes que respondieron perciben que el haber participado en un proyecto de desarrollo de software aplicando PISAgile en un contexto similar a uno real es de utilidad en su formación como profesionales (P4). Justamente este es uno de los principales motivos por el cual contar con un curso de este tipo.

Con la pregunta P5, se buscó que los estudiantes pudieran reflexionar sobre los principales aprendizajes durante el proyecto.

Hubo 25 estudiantes, de los 26 que respondieron afirmativamente la pregunta P4, que contestaron la pregunta P5. Estos 25 estudiantes reflexionaron y describieron los principales aprendizajes que tuvieron durante el curso. 23 de ellos identificaron diferentes aspectos aprendidos durante el curso. Los otros 2 estudiantes no indicaron aprendizajes específicos o la interpretación puede ser dudosa, por lo cual se excluyen del análisis. Uno de ellos (E18) nos deja alguna duda acerca de qué aprendió, el otro (E21) no indica un aprendizaje en particular.

De los estudiantes que no identificaron aprendizajes específicos uno (E18) comentó: “La gente es complicada y los trabajos en equipo que no tienen alguien a la cabeza no funcionan bien”. Esto se puede interpretar de diferentes

formas pero sería aventurarse en especulaciones.

El otro estudiante (E21) considera que aprendió de forma limitada a causa de su experiencia previa, "Hace ya casi 4 años que trabajo con metodologías ágiles, Scrum en particular. Por esto considero que mis aprendizajes durante el curso sobre el proceso estuvieron muy limitados pero esto se debe a mi gran conocimiento previo del asunto. Imagino igual que algo aprendí pero nada particular se me viene a la mente. Para personas que no han usado la metodología o hace poco la usan, me parece un buen primer acercamiento".

Clasificación de términos en el análisis cualitativo de P5

Luego de extraer y analizar los términos claves presentes en las reflexiones de los estudiantes, estos se clasificaron en cuatro categorías: conceptos vinculados al aprendizaje de habilidades blandas —*Habilidades blandas*—, conceptos vinculados a habilidades técnicas —*Habilidades técnicas*—, conocimientos técnicos generales —*Conocimientos técnicos*—, y *Otros aspectos*. Esta última se define para contemplar aspectos identificados por los estudiantes que no son catalogados en las anteriores categorías pero son de valor para el análisis. Una vez clasificados los términos presentes se unificaron según similitud semántica.

En algunos casos, definir el tipo de habilidad a la que pertenece un término no fue trivial. Esto es porque dependiendo de la interpretación, algunos de los términos pueden pertenecer a una u otra habilidad o implicar una conjunción de algunas habilidades. A modo de ejemplo, un estudiante (E1) expresó: "Mis principales aprendizajes fueron en torno a los requisitos, sea en la adquisición de los mismos con el cliente...", en este caso la obtención de requisitos tiene una parte vinculada a habilidades técnicas —aplicar métodos o herramientas que me permitan obtenerlos— y otra vinculada a habilidades blandas —como la comunicación—.

Para la clasificación en estos casos se resolvió tener en cuenta dos criterios: considerar todo el contexto —no solo el término extraído— y en otros casos si hace referencia a una disciplina del proceso se considera habilidad técnica y si es a los valores se considera habilidad blanda.

A continuación se analizan las reflexiones de los 23 estudiantes que respondieron a la pregunta P5 detalladamente.

Habilidades y conocimientos adquiridos, y su distribución

Del análisis de las reflexiones de los 23 estudiantes surgió la identificación de términos clave que fueron agrupados en las cuatro categorías ya mencionadas.

Se identificaron doce términos asociables a *Habilidades blandas*. Estos son: *Comunicación, Colaboración*, *Trabajo en equipo*, *Relaciones personales en un equipo*, *Negociación con cliente*, *Relacionamiento con cliente*, *Organización personal*, *Confiar en otros las tareas*, *Asignación responsabilidades*, *Capacidad de expresión oral y presentación*, *Aplicar valores y principios*, y *Tomar decisiones*. Como se puede apreciar los términos abordan distintos niveles de granularidad, ya que el término *Comunicación* hace referencia a una única área temática completa mientras que la *Capacidad de expresión oral y presentación* hace referencia a dos actividades en concreto.

Por otro lado, se identificaron diez términos asociables a *Habilidades técnicas*, principalmente estas tienen que ver con las principales disciplinas asociadas a la ingeniería de software. Estos son: *Requisitos, Gestión del proyecto*, *Medición y registro*, *Calidad del producto y proceso*, *Verificación*, *Gestión del conocimiento*, *División del trabajo en roles*, *Actividades del proceso*, *Preparación de demos*, y *Formación y entrenamiento*.

En las respuestas figuraron cinco términos asociables a *Conocimientos técnicos*: *Conocimientos tecnológicos*, *Nuevas tecnologías*, *Lenguaje de programación*, *Herramientas de gestión de versiones* y *Pasar a producción aplicaciones*.

Asociados a la categoría *Otros aspectos* se identificaron dos términos: *Trabajar en proyecto real* y *Aplicar un proceso de desarrollo de software*. Esta categoría nuclea términos muy amplios y que refieren explícitamente al carácter vivencial de la experiencia por la cual pasaron los estudiantes y su importancia.

De los estudiantes que indicaron su aprendizaje, un 96 % identificaron aprendizajes vinculados a habilidades blandas y técnicas necesarias para llevar a cabo un proyecto de software y un estudiante identificó aspectos catalogados en otros aspectos junto con conocimientos técnicos pero no habilidades. En cuanto a los que mencionan que adquirieron conocimientos técnicos, siempre fueron acompañados de aspectos incluidos en otras categorías.

Como se puede observar en la distribución por categorías de los aspectos del aprendizaje identificados (ver figura 6.1), un 83 % de los estudiantes identificó habilidades blandas vinculadas al proceso, un 74 % identificó habilidades técnicas, un 26 % conocimientos técnicos y un 26 % identificó otros aspectos.

Al observar la distribución de los aspectos del aprendizaje según tipo de habilidad (ver figura 6.2), el 22 % de los estudiantes perciben que su aprendizaje se basó únicamente en habilidades blandas, 3 estudiantes únicamente en

aspectos vinculados a habilidades técnicas y un 61 % de ellos en la combinación de ambos tipos de habilidades. Mientras que un estudiante no identificó aspectos pertenecientes a estas categorías.

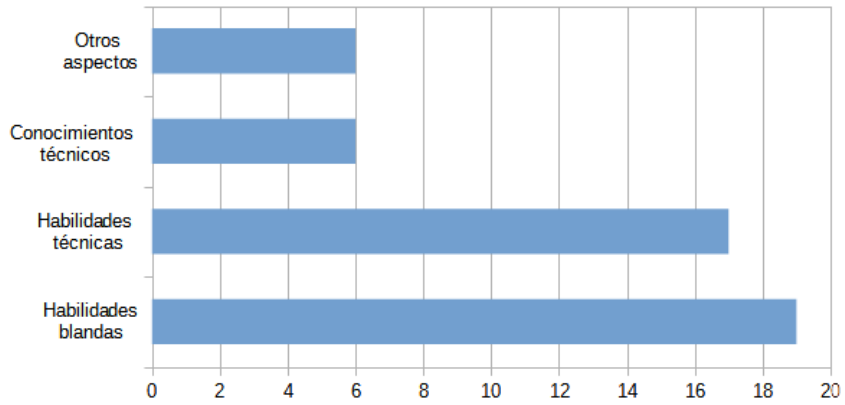

Figura 6.1: Distribución de los aspectos identificados agrupados por categorías

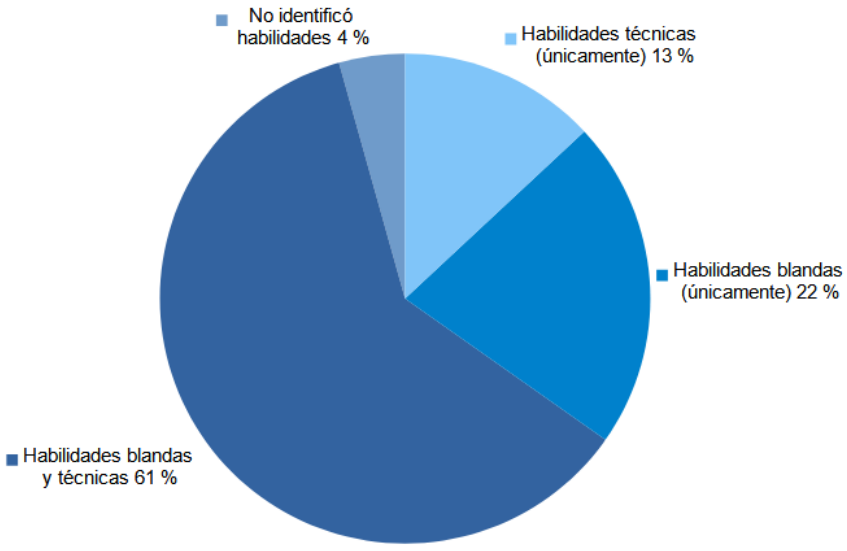

Figura 6.2: Distribución de los aspectos de aprendizaje según tipo de habilidad

De las habilidades blandas, los aspectos más identificados por los estudiantes fueron el *Trabajo en equipo*, *Comunicación* y *Colaboración*. En la nube de términos (ver figura 6.3) se puede apreciar la incidencia de todos los aspectos. En un segundo lugar se identificaron aspectos vinculados directamente con el cliente —relacionamiento y negociación— y a la organización personal. Los que identificaron como aprendizaje el trabajo en equipo resaltaron el desafío de trabajar en equipos grandes. Por ejemplo, un estudiante (E26) indicó “Al no

tener experiencia previa en el ámbito laboral, la materia me ayudó un montón a entender cómo es el trabajo en un equipo grande. ..", y otro en referencia al vinculo con el cliente (E6) comentó "Aprendí a saber cómo manejar prioridades como también cómo manejarme con un cliente real".

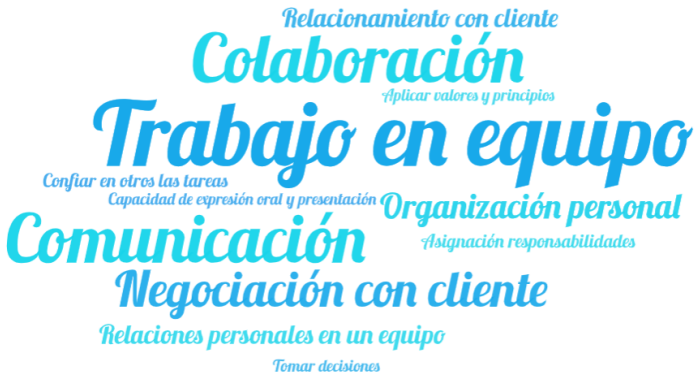

Figura 6.3: Nube de términos vinculados a habilidades blandas del proceso

En cuanto a los aspectos vinculados a las habilidades técnicas del proceso, en la nube de términos (ver figura 6.4) se puede apreciar que prevalece el aprendizaje vinculado a la *Gestión de proyectos*, la *Medición y registro*, y la *División de trabajo en roles*. El aprendizaje sobre las *Actividades del proceso*, *Requisitos*, *Calidad y verificación* aparecen en un segundo grupo de incidencia. El estudiante E6 comentó que "Principalmente aprendí a que la gestión de un proyecto de software tiene sus complejidades. Aprendí a saber cómo manejar prioridades...", mientras que otro (E1) mencionó "Mis principales aprendizajes fueron en torno a los requisitos, sea en la adquisición de los mismos con el cliente, la forma de describirlos en tarjetas y disponer tiempo en refinarlos."

Figura 6.4: Nube de términos vinculados a habilidades técnicas del proceso

Al analizar los aspectos vinculados a la categoría *Otros aspectos*, en nuestra opinión vemos relevante que algunos estudiantes, sin preguntarles directamente, valoren en sus reflexiones la importancia a nivel vivencial el haber participado de una experiencia similar a la real en un entorno más protegido. Un 26 % de los estudiantes resaltaron el aprendizaje que les dejó el trabajar en un proyecto real con un cliente real. Un estudiante (E12) indicó "Como primera aproximación a proyectos reales es genial. Se aprende muchísimo de muchas cosas,...", otro (E11) escribió que "[el aprender a] realizar un proyecto de comienzo a fin con todo lo que eso conlleva." Para contar con una visión más general de todos los estudiantes, se podría agregar en una próxima encuesta una pregunta para que reflexionen concretamente acerca de qué les dejó la experiencia.

A nivel general, todos los aspectos identificados por los estudiantes son parte del aprendizaje buscado en el curso. En ellos se visualizan los principios y valores, disciplinas, y actividades propuestas en PISAgile para lograr los objetivos de aprendizaje del curso.

Algunas reflexiones de los estudiantes llamaron la atención, estas se presentan a continuación. Un estudiante (E13) hizo referencias a "ceremonias de Scrum" siendo que el proceso aplicado fue PISAgile. Esto se podría llegar a interpretar como que el estudiante no entendió del todo el proceso o porque algunas de las actividades propuestas en PISAgile tienen características en común con las ceremonias de Scrum. Un estudiante (E19) identificó aprendizajes vinculados a desafíos presentados por consecuencia de la pandemia porque el trabajo durante el 2020 debió ser ejecutado de forma remota: "Tiempo que lleva facilitar proyecto con equipos remotos". En cuanto a la valoración de la forma colaborativa del aprendizaje y la oportunidad de contar con una «experiencia real» un estudiante (E26) comentó acerca del valor que tuvo para él: "Al no tener experiencia previa en el ámbito laboral, la materia me ayudó un montón a entender cómo es el trabajo en un equipo grande. Aplicar los valores y principios nombrados en los documentos fue fundamental y con resultados muy positivos, sobre todo la comunicación. Principalmente porque habían varios integrantes en el grupo que ya tenían bastante experiencia tanto en metodologías similares como en la tecnología usada, y otros tantos (incluyéndome) no teníamos nada de experiencia por lo que la colaboración ayudó a que ya luego del primer sprint de construcción, estuviéramos todos con un nivel aceptable y con facilidad para llevar el proceso correctamente.

Los facilitadores también fueron de mucha ayuda por lo que considero que es un rol clave."

6.4.1.2. Aprendizaje de habilidades blandas durante el curso

Para evaluar de forma específica el aprendizaje de habilidades blandas se incluyeron en la encuesta las siguientes preguntas:

P6: A continuación se presenta un listado de ciertas habilidades blandas. Indica para cada una de ellas cuál de los siguientes comentarios se adapta mejor a lo ocurrido: (el listado de habilidades blandas se presentó en la sección 6.1, y la escala de aplicación y desarrollo para que los estudiantes indiquen cuál comentario se adapta mejor se presentó en la sección 6.2)

P7: Para cada una de las habilidades blandas de las que has incorporado o mejorado en el PIS, reflexiona y describe cómo o en qué situaciones notaste que la has aplicado y/o aquellas que te ayudaron a incorporarla o mejorarla.

La pregunta (P6) la respondieron 25 estudiantes. La tabla 6.3 presenta los porcentajes obtenidos para cada habilidad blanda. Respecto a la pregunta (P7) cabe aclarar no será analizada en el marco de esta tesis.

Lo primero a destacar es que todos los estudiantes manifiestan conocer las 16 habilidades presentadas. Esto se debe a que las conocían previamente al curso o las conocieron durante el curso.

Las habilidades que los estudiantes incorporaron más debido al PIS son *comunicación del equipo con cliente, coach y otros stakeholders*, y *orientación al cliente*. Esto es consistente con el acercamiento que están teniendo a un proyecto real aplicando un proceso donde la comunicación entre los distintos stakeholders que participan del proyecto es fundamental para su aplicación y con el hecho de que la *comunicación continua y abierta* es uno de los valores de PISAgile.

En cuanto a las que ya tenían incorporadas y mejoraron, hay dos habilidades que presentaron los niveles más altos donde aproximadamente un 50 % considera que en las habilidades en las que sucedió esto fue en: *trabajo en equipo y colaboración*, y *apertura a los cambios y adaptabilidad*. Al igual que en el caso anterior se vinculan con los cimientos sobre el cual se construyó PISAgile, en este caso se vinculan con el principio *Colaboración de equipo* y el de *Adaptabilidad*.

Tabla 6.3: Distribución porcentual (%) de respuestas de estudiantes acerca de la aplicación y aprendizaje de habilidades blandas (P6), un estudiante corresponde a un 4%

Habilidades blandas	Apliqué e incorporé debido al PIS[1] (%)	Apliqué y mejoré debido al PIS[2] (%)	Apliqué en el PIS e incorporé antes del PIS[3] (%)	No la apliqué en el PIS[4] (%)	No la conozco[5] (%)
comunicación del equipo con cliente, coach y otros stakeholders	28	36	32	4	0
comunicación dentro del equipo de estudiantes	4	44	52	0	0
habilidad personal para establecer un buen relacionamiento con clientes, equipo, coach, etc.	0	44	56	0	0
pensamiento crítico y resolución de problemas	4	32	60	4	0
trabajo en equipo y colaboración	0	52	48	0	0
habilidad personal para organizar de forma eficaz y eficiente el trabajo	12	44	44	0	0
habilidad para trabajar independientemente	4	16	80	0	0
apertura a los cambios y adaptabilidad	4	48	44	4	0
compromiso y responsabilidad	0	20	80	0	0
liderazgo	4	28	28	40	0
orientación al cliente	24	24	28	24	0
capacidad para trabajar bajo presión	8	24	56	12	0
manejo de conflictos	4	24	36	36	0
negociación	16	24	24	36	0
capacidad de expresión en la escritura y reporte	8	44	44	4	0
capacidad de expresión oral y presentación	8	28	32	32	0

[1]La apliqué en el PIS y entiendo que he logrado incorporar la "habilidad blanda" DEBIDO al PIS

[2]La apliqué en el PIS, pero ya la tenía incorporada ANTES del PIS y he logrado mejorarla

[3]La apliqué en el PIS, pero ya la tenía incorporada ANTES del PIS

[4]No la apliqué durante el PIS (no la aprendí o afiancé durante el el curso)

[5]No conozco esta habilidad blanda

Un 80 % de los estudiantes considera que hay 2 habilidades que aplicó durante el PIS pero que ya las tenía incorporadas y no hubo mejoras, estas son: *habilidad para trabajar independientemente*, y *compromiso y responsabilidad*. En ambos casos tiene mucho sentido dado el avance que tienen los estudiantes en su carrera al momento de realizar el curso. A su vez, el primer caso puede deberse a las características de PISAgile que se centra en reforzar la habilidad de trabajar en un equipo antes que el trabajo individual. En cuanto al segundo caso también puede ser a causa de todo lo que implica para una persona definir su compromiso y responsabilidad ante otros.

La mayoría de las habilidades blandas que son objetivo de aprendizaje del curso fueron aplicadas por casi todos los estudiantes. Diez de ellas se pusieron en práctica durante los proyectos por el 100 % de los estudiantes aproximadamente y dos de ellas fueron aplicadas por el 76 % y el 88 %, conservando ambas altos niveles de aplicación. Las cuatro restantes fueron puestas en práctica por aproximadamente un 60 % de los estudiantes. .

Las habilidades de *liderazgo*, *manejo de conflictos* y *negociación* no fueron aplicadas por aproximadamente un 40 % de los estudiantes, y un 32 % no aplicó *capacidad de expresión oral y presentación*. A diferencia de las otras habilidades, es factible que estas no haya sido necesario aplicarlas. Esto se relaciona con lo mencionado anteriormente del trabajo en equipo y con el hecho de que los estudiantes desempeñaron distintos roles, entonces, dependiendo del rol, puede ser que se utilicen o no. Por ejemplo, si un estudiante no tuvo contacto con el cliente y no realizó presentaciones, puede ser que no haya aplicado *negociación* y *capacidad de expresión oral y presentación*. Respecto al *manejo de conflictos* puede suceder que dado el contexto del proyecto no haya sido necesaria, ya sea por no existir situaciones que lo requieran o porque si bien se intenta que la experiencia sea lo más real posible, no se fuerza para que se dé en el contexto académico toda la complejidad del comportamiento humano.

Luego de contar con una síntesis general de los resultados de las respuestas de los estudiantes, se analizaron los resultados en busca de evaluar la evolución en el desarrollo y fortalecimiento de las habilidades.

Análisis según evolución del aprendizaje percibida por los estudiantes

Para su análisis, las respuestas de los estudiantes sobre las cinco opciones de la pregunta acerca de la habilidades (P6) se agruparon según habilidad y

evolución del desarrollo de esta durante el curso. La evolución del desarrollo indicada por los estudiantes fue reagrupada en tres categorías —se tuvo en cuenta si el impacto fue positivo, neutro o negativo— estas son: *Hubo mejora*, *No hubo mejora* y *No la aplicó*. En la primera se incluyen las primeras dos opciones brindadas al estudiante y en la última las dos últimas. Como se puede observar en la figura 6.5 de distribución de las respuestas brindadas por los estudiantes, en varias de las habilidades hay cierta paridad en la cantidad de estudiantes que consideran que mejoraron o incorporaron determinada habilidad, los que consideran que no hubieron mejoras y/o los que no la aplicaron durante el curso. Se estableció un valor de un 16 % para considerar que hay paridad en la distribución de las respuestas brindadas por los estudiantes. Esto corresponde a una diferencia de cuatro estudiantes.

En todas las habilidades hubo estudiantes que percibieron mejoras, aproximadamente el 50 % de los estudiantes percibió mejoras en ocho habilidades, y entre un 30 % y 40 % lo hizo en seis de ellas. La cota mínima de estudiantes que mejoraron una habilidad fue del 20 % para dos habilidades. Estos números adquieren mayor relevancia si se tiene en cuenta que el 68 % de los estudiantes ya cuenta con experiencia en proyectos de desarrollo de software.

Hay dos habilidades en las cuales la cantidad de estudiantes que consideran que la mejoraron debido al PIS supera a los que no tuvieron mejoras, estas son: *comunicación del equipo con cliente, coach y otros stakeholders* y *orientación al cliente*. Son cuatro las habilidades en las que la cantidad de estudiantes que consideran que no mejoraron durante el curso supera a los que sí tuvieron mejoras; estas son: *habilidad para trabajar independientemente*, *compromiso y responsabilidad*, *pensamiento crítico y resolución de problemas* y *capacidad para trabajar bajo presión*. En las dos primeras hay un 80 % de los estudiantes que indica que no hubo mejoras, mientras que en las otras dos un 60 % aproximadamente. El resto de los resultados para las otras habilidades no son determinantes debido a la paridad en la distribución de las respuestas entre las distintas categorías.

En resumen, se observa que los estudiantes perciben una alta aplicación de habilidades blandas durante el proyecto. Sumado a ello hay un buen porcentaje de estudiantes que perciben mejoras en ellas a causa de la experiencia en el curso, a excepción de cuatro en las que hay un alto porcentaje de estudiantes que las aplicaron pero no percibieron mejoras. El hecho de que los estudiantes hayan tenido la necesidad de poner en práctica cada una de estas habilidades,

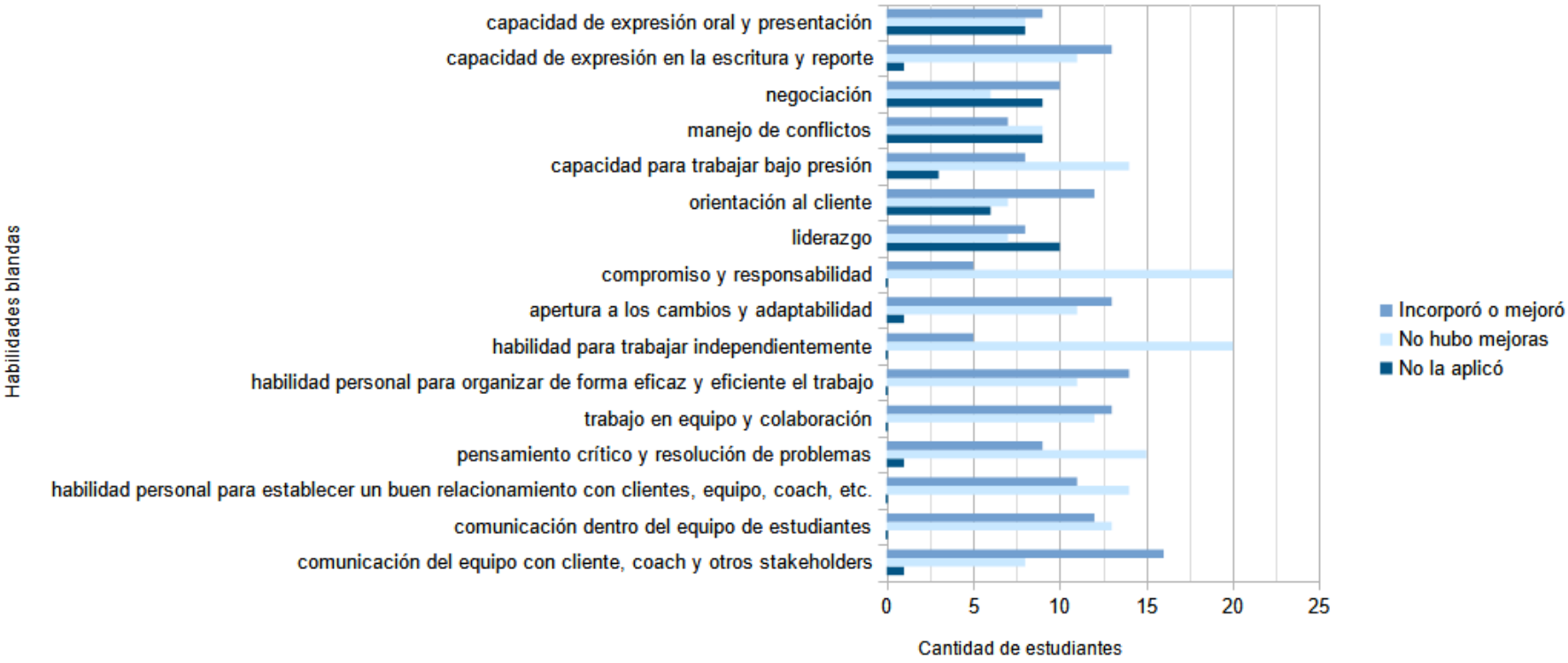

Figura 6.5: Distribución de respuestas por habilidad blanda

no es un aspecto menor. Su importancia radica en que son justamente las habilidades blandas que los profesionales necesitan (Khakurel y Porras, 2020). Por lo tanto, la percepción de los estudiantes, sobre la aplicación de PISAgile es que, considerando el uso de habilidades blandas, la experiencia vivida en el PIS se asemeja de forma considerable a la realidad.

6.4.2. ¿El proceso sirve para que se construya un producto de software con la calidad esperada?

La principal finalidad de un proceso de desarrollo de software es guiar la construcción de productos de software de calidad acorde a los criterios y las necesidades del cliente y en los plazos acordados. Para que ello suceda, la calidad debe estar enfocada tanto en el proceso como en el producto. Para conocer la percepción de los estudiantes sobre este tema se consideran las siguientes dos preguntas:

P11: ¿Cuán de acuerdo estás con la siguiente frase: "el desarrollo de las fases, las actividades y otros elementos de PISAgile colaboraron en que el equipo pueda evaluar y reflexionar sobre el proceso para realizar mejoras en el mismo"? La pregunta cerrada tuvo las siguientes posibles respuestas: Muy en desacuerdo, En desacuerdo, Ni acuerdo ni desacuerdo, De acuerdo, Muy de acuerdo.

P13: ¿Cuán de acuerdo estás con la siguiente frase: "Consideras que la calidad del producto es adecuada y estuvo acorde a los parámetros y criterios de aceptación esperados con el cliente"? Esta pregunta tuvo las mismas opciones de respuesta que la pregunta P11.

Existen otras dos preguntas P12 y P14 vinculadas a responder la misma pregunta de investigación. Sin embargo, detectamos al momento del análisis que P12 tiene una interpretación similar a P13. Optamos por analizar P13 y descartar P12. Además, corregimos la encuesta para estudios futuros.

La pregunta P14 es abierta. Excluimos la misma del análisis para limitar el alcance de esta tesis.

Para obtener software de calidad es importante mejorar los procesos de desarrollo utilizados (W. Humphrey, 2000). Justamente, la pregunta P11 busca conocer la percepción de los estudiantes acerca del aporte de los elementos del proceso a la mejora del mismo. Esta pregunta la respondieron 24 estudiantes.

La tabla 6.4 presenta las respuestas de los estudiantes dividida por grupos y totales.

Tabla 6.4: Percepción de los estudiantes sobre si los elementos de PISAgile colaboran en la mejora del proceso (P11)

Grupo nro	Muy de acuerdo[a]	De acuerdo[a]	Ni acuerdo ni desacuerdo[a]	En desacuerdo[a]	Muy en desacuerdo[a]	N/C[a]
2	0	3	2	0	0	2
3	0	6	2	0	1	2
5	0	4	3	3	0	0
Total	0	13	7	3	1	4

[a]Cantidad de estudiantes

El 54 % de los estudiantes está de acuerdo en que los elementos de PISAgile colaboran para que el equipo pueda trabajar en la mejora del proceso, un 29 % no está de acuerdo ni en desacuerdo y el 17 % no está de acuerdo.

Cuando se diseñó y construyó PISAgile se tomó en cuenta la inclusión de varias actividades y otros elementos, que faciliten la evaluación y reflexión para así promover la mejora del proceso durante su aplicación. Dado los resultados, aproximadamente la mitad de los estudiantes no perciben que los elementos de PISAgile colaboren con la mejora del mismo. Que esto sea así, nos conduce a la necesidad futura de profundizar en los motivos por los cuales sucede esto con el fin de entenderlo y de ser necesario hacer mejoras en PISAgile.

Al analizar la pregunta P13, encontramos que todos los estudiantes están de acuerdo (35 % muy de acuerdo y 65 % de acuerdo) en que la calidad del producto es adecuada y acorde a los criterios de aceptación esperados por el cliente.

En definitiva, todos los estudiantes que respondieron P13 consideran que se construyó un producto de software con la calidad esperada en un proyecto que tuvo una duración de 13 semanas aplicando el proceso PISAgile. Por lo tanto, basándonos en la percepción de los estudiantes, si bien no es determinante, podemos decir que PISAgile sirve para que se construya un producto de software con la calidad esperada acorde a los parámetros establecidos por el cliente. Lo que no podemos inferir de este análisis es en que medida lo es, ya que nos está faltando analizar la visión del cliente. Dicha evaluación se realizará en el futuro cercano.

6.4.3. ¿Cuáles son las fortalezas o debilidades al momento de comprender o aplicar el proceso?

De igual forma que en PISAgile se marca la importancia de evaluar, revisar y reflexionar de forma continua su aplicación en el proyecto para poder identificar y llevar a cabo acciones de mejoras, es importante aplicar el mismo enfoque de mejora continua en la definición de PISAgile para el curso. Particularmente, al ser la primera vez que es aplicado, nos interesa conocer la percepción de los estudiantes acerca de la comprensión o aplicación de PISAgile para a partir de ello analizar oportunidades de mejora. A los estudiantes se les consultó:

P17: ¿Tuviste dificultades a la hora de comprender o aplicar el proceso? (Sí, No)

P18: Con respecto a la pregunta anterior: ¿Cuáles fueron? ¿Propondrías algún cambio al proceso PISAgile para mejorarlo y evitar esas dificultades? (Texto libre)

P19: ¿Qué características o elementos de PISAgile consideras que son un acierto y que te ayudaron a la hora de comprender o aplicar el proceso? (Texto libre)

La pregunta P17 fue respondida por 23 estudiantes y existe una paridad entre los que presentaron y no presentaron dificultades, como se puede observar el la tabla 6.5. Solo con estos datos no podemos determinar si PISAgile presenta dificultades (o no) en su comprensión y aplicación. La pregunta P17 no brinda información acerca de las causas, a su vez no es generalizable ya que los procesos de aprendizaje son complejos.

Algo que llama la atención es que el comportamiento no es igual al analizar por separado cada grupo. El 80 % de los integrantes del grupo 5 plantean haber presentado dificultades, mientras que un 87 % del grupo 3 manifiesta lo contrario. Dado que a todos se les entregó la misma especificación, a futuro resultaría interesante investigar qué otros aspectos pueden estar incidiendo en la comprensión y aplicación del proceso.

En el análisis cualitativo de las preguntas P18 y P19 se identificaron términos y elementos relevantes en la reflexión de los estudiantes acerca de las dificultades y los aciertos de PISAgile para su comprensión. Se encontraron visiones contrapuestas. Algunos estudiantes plantean como dificultad que la documentación es extensa y difícil de leer, mientras que otros indican como un

Tabla 6.5: Distribución de respuestas de la pregunta P17

Grupo nro	Sí[a]	No[a]	N/C[a]	Total[a]
2	2	3	2	7
3	1	7	3	11
5	8	2	0	10
Total	11	12	5	28

[a]Cantidad de estudiantes

acierto la documentación del proceso y que la misma es clara. Sin embargo, es importante tener en cuenta que 7 de los 11 estudiantes que manifestaron haber tenido dificultades para comprender o aplicar PISAgile, indicaron dentro de sus reflexiones aspectos vinculados a la documentación.

Respecto a los aciertos, los elementos que aparecen con mayor frecuencia en las reflexiones de los estudiantes son: las fases e iteraciones, roles, y que el proceso incorpore la mejora continua particularmente por medio de las actividades de evaluación y reflexión —principalmente las reuniones con el *coach* y la retrospectiva al realizar la evaluación de la iteración—. Algunos estudiantes resaltaron como acierto que el proceso tome elementos de otros procesos para su construcción, como ser Scrum y MUM. Esto puede tornarse un riesgo si los estudiantes toman ciertas asunciones de ello y terminan aplicando otro proceso que no es el especificado; un estudiante (E21) indicó "asumí que PISAgile era Scrum con más documentación y no me preocupé en investigar la metodología". Otros lo ven como una oportunidad; un estudiante (E10) mencionó en sus reflexiones que "Presenta un punto intermedio entre Scrum y MUM, lo que en sí mismo para mí es un avance, ya que por falta de experiencia nos podemos perder un poco en Scrum, y MUM puede ser muy tedioso y ajustarse poco a la realidad actual".

6.5. Limitaciones

Las principales limitaciones al evaluar PISAgile a través de la percepción de los estudiantes se vinculan al método de recolección de datos y a la disponibilidad paulatina del proceso durante el curso. Estas limitaciones principalmente son a consecuencia de dificultades en el ajuste al calendario de clases y por limitar el alcance de esta tesis.

El análisis realizado corresponde a un estudio cuantitativo y cualitativo

de las reflexiones de los estudiantes. El factor interpretativo inherente a todo análisis cualitativo siempre es un riesgo al momento de analizar resultados. Una limitación está dada por la inexperiencia en este tipo de estudios por parte de la única persona que intervino —la autora de la tesis—, y también un posible sesgo por ser la creadora de PISAgile y *coach* de uno de los grupos. A futuro, se podría hacer una posterior validación con estudiantes y una revisión de a pares incluyendo otro investigador.

El hecho de que los estudiantes no tuvieran el proceso de forma completa al comienzo del proyecto les hizo tener una visión parcial al arrancar y en consecuencia del trabajo a realizar. Si bien no es una limitante directa de la evaluación, es algo que puede afectar los resultados obtenidos.

Una limitante de las encuestas es que se basan en autoinformes de los encuestados sobre sus comportamientos y actitudes, esto puede sesgar los resultados de diversas maneras. Sumado a ello las personas no son buenas registrando los eventos (Lethbridge et al. 2005). Para mitigar el impacto de ello se generaron preguntas donde los estudiantes tuvieran que describir las situaciones asociadas a otras preguntas sobre el mismo aspecto.

Al ser encuestas voluntarias es posible que haya un comportamiento cooperativo de los estudiantes para el resultado exitoso de la evaluación que se está realizando y eso sesgar algunas de sus respuestas. Rosnow y Rosenthal (1997) hacen referencia al "efecto del buen sujeto", en el cual los participantes voluntarios tienden a tener un gran interés en el resultado exitoso del experimento, siendo particularmente sensibles en apoyar el trabajo. Considerando que los estudiantes conocen la finalidad y el contexto en el cual se realizó la encuesta esto puede impactar en los resultados.

Por último, la encuesta solo fue aplicada para los estudiantes que participaron de PISAgile, no pudiendo realizar un análisis comparativo con otros procesos aplicados en el curso.

6.6. Comentarios finales

Todos los estudiantes perciben que lograron construir un producto de calidad acorde a lo solicitado por el cliente, en un proyecto con una duración media y trabajando en un grupo grande de estudiantes, pese al desafío que algunos expresan que esto último les implicó. La mitad de dichos estudiantes no percibe que los elementos de PISAgile colaboren con la mejora del proceso,

abriendo así una línea interesante de investigación futura. Nuestro resultado coincide con el reportado por Burden et al. (2019), donde los estudiantes crearon valor para otros aplicando un proceso ágil "que les funcionó" con grupos de tamaño medio.

Tal como lo reportado por Germain et al. (2002), Khakurel y Porras (2020), Burden et al. (2019), y Paasivaara et al. (2019) el haber participado en un curso basado en proyectos aplicando un proceso de desarrollo definido ayuda a los estudiantes a prepararlos en su práctica profesional y a mejorar determinadas habilidades blandas. En nuestro estudio, todos los estudiantes perciben que el haber participado del proyecto aplicando PISAgile los ayudará en su práctica profesional y cuando reflexionan de forma libre sobre cuál fue su aprendizaje, ocho de las doce habilidades blandas que hacen referencia coinciden con las seleccionadas por nosotros por ser importantes en los proyectos de desarrollo de software. De las habilidades que seleccionamos, la mitad de los estudiantes percibieron mejoras en ocho de ellas y algo menos de la mitad de ellos en seis. Tanto en las reflexiones libres como en el listado de habilidades proporcionado, resaltan aprendizaje en la comunicación, el trabajo en equipo y la colaboración.

Casi todos los estudiantes no percibieron mejoras en dos de las habilidades —habilidad para trabajar independientemente, y compromiso y responsabilidad—, nuestro resultado coincide con lo reportado por Khakurel y Porras (2020). Entendemos que estas habilidades fueron adquiridas, a un alto nivel, en cursos anteriores y también en la práctica profesional (en el caso de los estudiantes que trabajan en la industria).

En referencia a la percepción de los estudiantes sobre el proceso, la mitad de los estudiantes no presentó dificultades en su comprensión. Sin embargo, entendemos en base a las reflexiones —de la otra mitad de los estudiantes— que hay oportunidades de mejora en la documentación del proceso. Al analizar la encuesta notamos que las propuestas de mejora se solicitaron solo a los estudiantes que presentaron dificultades, este es un aspecto a corregir en la encuesta. Menos de la mitad de los estudiantes que presentaron dificultades plantearon propuestas de mejoras. Las mejoras propuestas abordan aspectos vinculados a los entregables —no hacer algunos de ellos y contar con plantillas para su elaboración— y a determinados aspectos del curso —incluir clases de presentación de PISAgile, tener más de una reunión semanal con el *coach* y cambiar los días de entrega—. Respecto a las fortalezas, al igual que lo reportado por Hsu et al. (2019) los estudiantes perciben que la existencia

de la actividad de retrospectiva es importante. Nuestros estudiantes también perciben como otras fortalezas: contar con fases e iteraciones, roles, la mejora continua del proceso en si mismo, y los encuentros con el *coach*.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

La enseñanza en ingeniería de software (IS) en las universidades busca preparar a los estudiantes para su futura práctica profesional en la industria de software mediante el desarrollo de las competencias necesarias. El desarrollo de competencias se hace de forma evolutiva mediante la adquisición de los conocimientos, desarrollo de las habilidades y la comprensión de cómo estos pueden ser combinados para ser aplicados en determinado contexto que les permita ejecutar una tarea.

La construcción de software de calidad, las altas exigencias de tiempo de salida al mercado, contar con procesos de desarrollo y profesionales para aplicarlos, es una preocupación de la comunidad del software. Compartimos con Bastarrica et al. (2017), Hsu et al. (2019) y Matthies et al. (2019), entre otros, que, para lograrlo, dicha construcción debe estar en un contexto de proceso de desarrollo ágil —aplicado de forma disciplinada— que se adapte rápidamente a las necesidades cambiantes y cada profesional que participe debe tener las competencias necesarias.

Los procesos de desarrollo que incorporan la reflexión y evaluación en ciclos cortos de construcción (donde se trabaja de forma colaborativa y se entienden cuáles son las actividades y las responsabilidades) genera en los involucrados una disciplina y un sentido de pertenencia que promueve la mejora continua del proceso, las competencias y la construcción de software de calidad.

Para enseñar procesos de desarrollo de software y para contribuir a que los estudiantes desarrollen competencias y habilidades necesarias para la industria

de software, uno de los métodos utilizados son los cursos *capstone* basados en proyectos con un contexto similar al de la industria del software, donde los estudiantes siguen un proceso de desarrollo. Estos cursos promueven el desarrollo de las competencias de forma integral. Este método es el aplicado en el PIS.

Una necesidad que teníamos desde hace unos años en el curso era contar con un proceso de desarrollo ágil especificado y documentado y que diera respuesta a solicitudes y restricciones de la industria.

En el marco de este trabajo realizamos una revisión sistemática de la literatura para conocer las características y experiencias de cursos basados en proyectos de desarrollo de software. A esto se sumaron estudios aportados por expertos y un estudio de revisión de antecedentes con un objetivo similar. De los cursos presentados en los artículos extrajimos los objetivos de aprendizaje, métodos de enseñanza, características de los proyectos y su cercanía con experiencias del «mundo real», procesos de desarrollo aplicados, y los resultados de su aplicación.

Como resultado de la revisión encontramos que existen diversas formas de implementar un curso similar al PIS. Obtuvimos algunas ideas para este trabajo como enfocar la investigación hacia algún aspecto en particular del aprendizaje, por ejemplo, las habilidades blandas. Como dificultad, encontramos que varios de los estudios no reportaban detalles del proceso de desarrollo de software aplicado, ni consideraciones para su creación y se centraban en analizar algún aspecto particular del curso, pero no vinculados al aprendizaje ni a la evaluación del proceso.

Luego de hacer un estudio para evaluar el uso de determinados procesos de desarrollo, concluimos que no había uno que se adaptara de forma completa a nuestro curso, pero sí algunos aspectos. Los evaluados surgieron de: una revisión de literatura que presentaba un resumen de los principales procesos de desarrollo, conocimiento previamente adquirido -en la especialización realizada o en el propio PIS- y de charlas con especialistas. De estos, se captaron aquellos elementos y mejores prácticas que entendimos importantes para incluir desde el punto de vista de las características del PIS y los objetivos de aprendizaje.

Logramos proponer y crear una primera versión de un proceso ágil (PISAgile) para que pueda ser aplicado de forma disciplinada y estandarizada en el PIS. Para su creación tomamos como base Scrum, por ser el marco de trabajo ágil más enfocado en la gestión del desarrollo. Desde nuestra perspectiva fue

necesario complementar Scrum con algunos aspectos no cubiertos: el abordaje inicial del proyecto para conformar el equipo y tener una visión global de este, su cierre para la transferencia definitiva al cliente, y el abordaje de las disciplinas de ingeniería de software no contempladas. Estos aspectos los complementamos con aportes personales, sugerencias de expertos y otros procesos o prácticas, tales como, Agile Unified Process (AUP)[1], Proceso Modularizado Unificado y Medible (MUM), Team Software Process (TSP) (W. S. Humphrey, 2005), eXtreme Programing (XP)[2] y Kanban.

El principal desafío que encontramos fue realizar la construcción de PISAgile paralelamente a la revisión sistemática, al estudio de procesos y a su aplicación en el curso con fechas de entrega de documentación acotados. Entendemos que esto no afectó de forma significativa la calidad de la entrega del proceso a los involucrados (estudiantes, docentes y clientes), ya que previo a ello el proceso era revisado y validado. Consideramos que al no contar con una herramienta adecuada para realizar una documentación de procesos sumado a la entrega de PISAgile en etapas, tuvo un impacto negativo en la comprensión del mismo. Sin embargo, previendo mejoras futuras, se tuvo en cuenta el estándar para documentar procesos de software, SPEM V2.0 —Software Process Engineering Metamodel— (Object Management Group (OMG), 2008).

Con esto logramos cumplir con el primer objetivo general de esta tesis, *crear un proceso de desarrollo de software ágil para ser aplicado en el PIS que sirva para la construcción de software de calidad cumpliendo con las expectativas de los clientes y que por medio de su aplicación los estudiantes puedan incorporar y poner en práctica sus conocimientos en ingeniería de software, apliquen y desarrollen las habilidades (técnicas y blandas) necesarias para trabajar como profesionales en la industria del software.* Es importante resaltar que, a diferencia de otros estudios, el nuestro presenta cómo fuimos construyendo un proceso ágil para un contexto académico y una especificación detallada del mismo, que facilita su aplicación y evolución.

Nuestro trabajo reporta el resultado de evaluar distintos aspectos vinculados al proceso y al aprendizaje de los estudiantes —con foco en las habilidades blandas—, que surgen de la aplicación del proceso PISAgile en el PIS. Si bien encuestamos a todos los actores del proceso, el análisis se basa en la percepción

[1] http://www.ambysoft.com/unifiedprocess/agileUP.html
[2] http://www.extremeprogramming.org/

de los estudiantes.

Al igual que otros estudios, el nuestro reporta el beneficio de contar con un proceso ágil disciplinado y un curso como el PIS para preparar a los estudiantes para su desempeño profesional. Todos los estudiantes reportaron que su participación en el curso los ayudará en su práctica profesional. Se identifican aprendizajes vinculados a habilidades blandas y técnicas, con destacables mejoras en algunas de las habilidades blandas. Esto confirma resultados obtenidos por otros investigadores en estudios similares.

Como principal resultado, logramos que mediante la aplicación de PISAgile en un proyecto de desarrollo de software en un contexto similar al «mundo real», los estudiantes pongan en práctica la mayoría de las habilidades (que identificamos que deben ser aplicadas) y muchos de ellos lograron desarrollarlas durante el curso.

Nosotros creemos que nuestros estudiantes no lograron desarrollar o percibir el desarrollo de algunas habilidades blandas debido a que ello requiere tiempo y práctica que va más allá del curso. Entendemos que se debe a que intervienen procesos cognitivos complejos que pueden implicar mayor tiempo para visualizar y producir cambios.

Si bien, la mayoría de las habilidades fueron aplicadas por casi todos los estudiantes, nos encontramos con habilidades como *liderazgo* que no fue aplicada por un 40 % de estos. Creemos que en este caso (como en el de otras habilidades blandas) su aplicación depende del rol que tenga el estudiante en el proyecto. Por lo que es necesario reevaluar el contexto y si es posible brindar en el ámbito académico (ya sea en este u otro curso) un aprendizaje que promueva el desarrollo sobre determinadas habilidades blandas. Por esto es importante realizar un mapeo entre el desarrollo de habilidades y los roles del proceso como posible trabajo a futuro.

En nuestra opinión, teniendo en cuenta también los resultados del curso, aplicando PISAgile es posible construir productos de calidad y desarrollar el aprendizaje de los estudiantes. Finalmente, entendemos —luego de aplicado el proceso—, que PISAgile puede ser implantado en el PIS -más allá de la experiencia puntal del 2020- tomando como referencia la primera aproximación a su evaluación que nos brinda determinadas pautas respecto a cómo continuar. Uno de los principales desafíos es definir una evaluación holística del proceso para poder seguir mejorándolo.

De acuerdo a resultados obtenidos en nuestra evaluación, PISAgile sirvió

para que los estudiantes pudieran construir software de calidad acorde a lo esperado por el cliente y el curso; llegamos a ello teniendo en cuenta sus percepciones y la evaluación realizada por los clientes al finalizar el curso sobre los productos desarrollados (aunque entendemos que es conveniente poder complementar la visión de cliente con la encuesta que realizaron). Esto coincide con resultados de otros estudios, pero a diferencia de estos, el nuestro cuenta con equipos de estudiantes de mayor tamaño aplicando un proceso ágil. En nuestra opinión, uno de los principales motivos que hace que funcione PISAgile es por ser un proceso con un buen nivel de detalle, que brinda pautas específicas, que define roles para su ejecución y que durante su construcción entendemos que integramos elementos que promueven la mejora continua del proceso. Esto último es un aspecto con el cual no están de acuerdo la mitad de los estudiantes y que debemos explorar más.

Sumado a lo anterior, los estudiantes perciben que los principales aciertos de PISAgile son: contar con fases, iteraciones y roles; incorporar la reflexión y evaluación para la mejora continua, y contar con elementos conocidos de otros procesos. Mientras que las dificultadas en su comprensión son vinculadas a la documentación del mismo. En líneas generales, coincidimos con los estudiantes respecto a los aciertos y dificultades percibidos. Sin embargo, en el caso de la documentación para algunos es un acierto, entonces es necesario indagar más al respecto para su mejora.

Para finalizar, entendemos que contar con un proceso ágil como PISAgile en el PIS es una mejora importante respecto a la situación actual. Esto nos permite contar con un proceso actualizado y tener así dos procesos en el curso, con distintas características que puedan ser aplicados y evolucionados para acompañar el aprendizaje de los estudiantes y utilizarlos según las características del proyecto, dando respuesta a la necesidad planteada desde hace varios años en el curso.

Respecto al aprendizaje, luego de la puesta en práctica de PISAgile en un proyecto de desarrollo (de duración media con un cliente real) trabajando en un grupo de estudiantes en un contexto similar al «mundo real», todos los estudiantes se sienten más preparados para su práctica profesional. Percibieron que durante el curso pusieron en práctica un alto número de habilidades blandas que luego deberán aplicarlas como profesionales, y varios estudiantes incorporaron y mejoraron varias de ellas. O sea, desarrollan su aprendizaje respecto a los objetivos del curso aplicando PISAgile, independientemente de

que tengan (o no) experiencia en proyectos de la industria. Respecto a la construcción del software, los equipos lograron construir software de calidad acorde a las expectativas del cliente y del curso.

7.2. Aportes del trabajo

Los aportes de esta tesis son los siguientes:

- Retomamos la línea de investigación asociada al curso Proyecto de Ingeniería de Software (PIS), que desde hace nueve años estaba inactiva abordando una necesidad identificada por primera vez en 2013 de contar con un proceso ágil para el PIS (incluyendo también una revisión de los procesos utilizados hasta el momento). Esto es de importancia para los investigadores del curso, ya que no solo se reabre la línea de trabajo de mejora de procesos vinculados al curso, sino que también los resultados pueden ser utilizados para definir otras líneas. En particular detectamos que sería conveniente hacer una evaluación de las características, objetivos de aprendizaje y evaluación del curso para mejorarlo. También nos recuerda, como docentes investigadores, la importancia de explotar entornos como el PIS para la mejora de la enseñanza de nuestros estudiantes.
- Estudiamos cuáles eran las necesidades de un proceso ágil para el PIS, lo diseñamos y construimos. Los estudiantes lo pueden aplicar y ajustar acorde a las características de su proyecto para construir software de calidad, y desarrollar sus conocimientos y habilidades. Los docentes del curso pueden utilizar PISAgile, conjuntamente con la guía desarrollada para guiar a los estudiantes en su aplicación y en su aprendizaje y hacer propuestas de mejoras para el proceso y la guía. Mientras que otros investigadores pueden utilizar el proceso para aplicarlo (y evolucionarlo) en sus contextos y con el fin de generar mayor conocimiento en la enseñanza de procesos y el aprendizaje de los estudiantes. También se puede utilizar como base la forma en que creamos este proceso para generar otros.
- Ejecutamos una primera aproximación a la evaluación de PISAgile respecto a su adecuación a los fines para los que se creó aplicándolo en el curso. Particularmente utilizamos para esta evaluación la percepción de los estudiantes. Los investigadores del curso y otros investigadores pue-

den utilizar los resultados obtenidos de la evaluación para contrastarla con sus propios estudios similares o, en el caso de los primeros, complementar la evaluación realizada y la mejora de PISAgile.

- Estudiamos el proceso y los resultados de su aplicación con respecto a otros estudios vinculados a la enseñanza de procesos de desarrollo de software internacionales. Esto es de importancia para los docentes del curso ya que pueden utilizarlos para conocer la alineación de nuestro curso con otros, para dar mayor comprensión a las decisiones tomadas y para identificar oportunidades de mejora.

7.3. Limitaciones del trabajo

La principal limitación de este trabajo responde a restricciones de esfuerzo, alcance y calendario debido a que hubo que ajustar al cronograma de clases y limitar el alcance de la tesis. Esto pudo afectar el rigor del estudio ya que no contamos con algunos instrumentos de forma oportuna, como por ejemplo: conocimiento profundo de algunos aspectos vinculados a la enseñanza de procesos —pocos estudios encontrados con objetivos similares a los nuestros— y especificación de estos; no contar con la encuesta al comienzo del curso para ejecutarla y poder evaluar la brecha en la percepción de los estudiantes respecto a su aprendizaje. También limitamos el alcance de la evaluación de PISAgile: solo contemplamos la perspectiva de los estudiantes sin contrastarla con el resto. Entendemos que esto lo podemos mejorar a futuro, ya que el proceso contempla su mejora continua y proponemos continuar con la evaluación de PISAgile durante su implantación.

También, durante el año 2020 se contó con un contexto inusual en el PIS. A raíz del COVID-19, el curso pasó a modalidad virtual y los proyectos, a una semana menos de ejecución. A su vez, los actores no tenían disponible el proceso a aplicar de forma completa a principios del semestre, mientras que los otros procesos que se aplican en el curso ya estaban acordados y eran conocidos desde un comienzo. Esto afectó principalmente a los docentes del curso que debían guiar la enseñanza utilizando un proceso que iban conociendo conjuntamente con los estudiantes. Entendemos que lo anterior afecta la generalización del estudio ya que fue aplicado en un contexto distinto al usual del PIS. Sin embargo, creemos que en un contexto más favorable podemos obtener mejores resultados. Intentamos mitigar el impacto de la virtualidad por

medio de acciones del *coach* de cada grupo, y la disponibilización paultatina del proceso mediante un apoyo constante y directo de la autora de esta tesis. Para los próximos años el proceso estará disponible al comienzo del semestre.

Para finalizar, otra limitación de rigor, se vincula con que la propuesta inicial de PISAgile, y la extracción y parte del análisis de datos de la encuesta fueron realizadas únicamente por la autora de esta tesis. Esto puede introducir malas interpretaciones de datos por ser participante directa. Es importante notar que la autora de esta tesis tiene cinco años como docente en el PIS, durante el 2020 fue *coach* de un grupo que utilizó PISAgile, creó PISAgile y lideró su aplicación en el curso. Para mitigar el impacto durante su aplicación, se realizaron revisiones de PISAgile con otras personas —director de la tesis, otros docentes y especialistas—, mientras que para su evaluación se revisó la extracción de datos y en análisis con el director de la tesis.

7.4. Trabajo futuro

Para darle un primer cierre a la creación y evaluación de PISAgile, debemos impactar los cambios que surgen de la evaluación realizada. Los principales aspectos a mejorar es la forma documental del proceso, su distribución y la presentación a los actores. Para ello es necesario seleccionar una herramienta para su especificación y, al comienzo del curso, distribuirlo y realizar inducciones a PISAgile para los distintos actores.

Es interesante también definir y ejecutar una estrategia de implantación de PISAgile que contemple la evaluación de todos los actores y por consiguiente su ajuste. Deberíamos considerar la evaluación realizada en el marco de este trabajo evaluando las preguntas que quedaron pendientes de la encuesta de estudiantes, completando con las visiones de los docentes y clientes. Estos resultados se podrían contrastar con datos y métricas generados en los proyectos. Se podría incluir información relevada de forma informal durante su aplicación como ser: retrospectiva final del curso con los estudiantes, comentarios y consultas de los docentes no vertidos en la encuesta como por ejemplo la recabada en el Diario del *coach* o las que surgieron en los encuentros de apoyo a los *coach*.

Nos interesa también establecer una sistematización de la mejora continua para los procesos que tenemos definidos en el curso. Esta debe incluir hacer una revisión de otros métodos de evaluación que permitan enriquecer el trabajo

realizado. Por ejemplo, entrevistas, evaluación de a pares, aplicar las mismas encuestas al principio y al final, etc.

Para enriquecer los resultados obtenidos y los ajustes al proceso definido, sería bueno ampliar la revisión sistemática de la literatura realizada, ya que tuvo como fuente a determinadas conferencias o sugerencias de expertos. Ahora disponemos de mayor información para limitar el objeto de búsqueda.

Un aporte importante podría ser continuar este trabajo abordando un tema que fue tratado de forma tangencial: la gestión del conocimiento y generación de evidencia en el curso. Desde hace algunos años la memoria organizacional del curso está desactualizada y no tenemos sistematizada la recolección de datos de los proyectos. Por ejemplo, se podrían incluir nuevos recursos de soporte que colaboren con la ejecución del proyecto y a la vez que sirvan como herramienta para almacenar y analizar información (y los productos) de los proyectos ejecutados.

Creemos que sería bueno también estudiar alternativas respecto a la evaluación del aprendizaje de los estudiantes, con mediciones más objetivas, pese al desafío que ello puede implicar en este tipo de cursos. Por ejemplo, podría ser con evaluaciones periódicas por parte de los distintos actores que den lugar a identificar posibles mejoras para el estudiante y a su vez que brinde mayores herramientas al docente para ayudarlo.

Finalmente, disponibilizaremos el proceso de forma completa y los datos obtenidos en su evaluación —cuidando la privacidad de los mismos—, para aportar nuestro conocimiento con la comunidad científica y que pueda ser utilizado, evaluado y evolucionado. Entendemos que este conocimiento es distinto a otros casos, debido a que el procesos se creó específicamente para un contexto académico.

Bibliografía

Abad, Z. S. H., Bano, M. y Zowghi, D. (2019). How Much Authenticity can be Achieved in Software Engineering Project Based Courses? *IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 208-219.

Agile unified process (AUP) [URL http://www.ambysoft.com/unifiedprocess/agileUP.html. Último acceso: octubre 2022]. (s.f.).

Bacigalupo, M., Kampylis, P., Punie, Y. y Van den Brande, G. (2016). *EntreComp: The entrepreneurship competence framework* [URL https://ec.europa.eu/social/main.jsp?catId=1317. Último acceso: mayo 2022].

Basili, V. R. (1992). *Software Modeling and Measurement: The Goal/Question/Metric Paradigm* (inf. téc.). University of Maryland at College Park. Estados Unidos.

Bastarrica, M. C., Perovich, D. y Samary, M. M. (2017). What Can Students Get from a Software Engineering Capstone Course? *IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, 137-145.

Bastarrica, M. C., Perovich, D., Gutierrez, F. J. y Marques, M. (2019). A Grading Schema for Reinforcing Teamwork Quality in a Capstone Course. *IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 276-277.

Beck, K. y Andres, C. (2004). *Extreme Programming Explained: Embrace Change* (2.ª ed.). Addison-Wesley Professional.

Bourque, P. y Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge: Version 3.0 (SWEBOK)* (3.ª ed.). IEEE Computer Society.

Burden, H., Steghöfer, J.-P. y Hagvall Svensson, O. (2019). Facilitating Entrepreneurial Experiences through a Software Engineering Project Course. *IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 28-37.

Cha, S., Taylor, R. N. y Kang, K. (2019). *Handbook of Software Engineering* (1.ª ed.). Springer.

dinngo crowdfunding, c. (s.f.). *Design Thinking en español* [URL https://www.designthinking.es/inicio/. Último acceso: octubre 2022].

Dutson, A. J., Todd, R. H., Magleby, S. P. y Sorensen, C. D. (1997). A review of literature on teaching engineering design through project-oriented capstone courses. *Journal of engineering education*, *86*(1), 17-28.

Garousi, V., Giray, G., Tuzun, E., Catal, C. y Felderer, M. (2020). Closing the Gap Between Software Engineering Education and Industrial Needs. *IEEE Software*, *37*(2), 68-77.

Germain, T., Robillard, P. N. y Dulipovici, M. (2002). Process activities in a project based course in software engineering. *Annual Conference on Frontiers in Education*, S3G-S3G.

Heredia, A., Colomo-Palacios, R. y de Amescua, A. (2015). A Systematic Mapping Study on Software Process Education. *International Workshop on Software Process Education, Training and Professionalism (SPETP)*, 7-17.

Hilburn, T. y Humphrey, W. (2002). Teaching teamwork. *IEEE Software*, *19*(5), 72-77.

Hsu, H.-J., Lin, E., Chang, K. y Hsiao, E. (2019). Practicing Scrum in Institute Course. *IEEE Conference on Software Engineering Education and Training (CSEE&T)*, 7770-7778.

Humphrey, W. (2006). *TSP(SM) Coaching Development Teams (SEI Series in Software Engineering)* (1.ª ed.). Addison-Wesley Professional.

Humphrey, W. (2000). *The Personal Software Process (PSP)* (inf. téc.). Software Engineering Institute, Carnegie Mellon University. Pittsburgh, PA.

Humphrey, W. (2005). *PSP: A Self-Improvement Process for Software Engineers* (1.ª ed.). Addison-Wesley Professional.

Humphrey, W. S. (2005). *TSP(SM)-Leading a Development Team (SEI Series in Software Engineering)* (1.ª ed.). Addison-Wesley Professional.

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, 1-84.

InCo. (2017). *Programa de Proyecto de ingeniería de software - Instituto de computación, Facultad de ingeniería, Udelar* [URL https://www.fing.

edu.uy/es/curso/grado/2019/proyecto-de-ingenieria-de-software. Último acceso: mayo 2022].

Joint Task Force on Computing Curricula. (2013). *CS2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science* (1.ª ed.). Association for Computing Machinery (ACM) IEEE Computer Society.

Joint Task Force on Computing Curricula. (2015). *SE 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering* (1.ª ed.). Association for Computing Machinery (ACM) IEEE Computer Society.

Joint Task Force on Computing Curricula 2005. (2005). *Computing Curricula 2005 - The Overview Report covering undergraduate degree programs in Computer Engineering, Computer Science, Information Systems, Information Technology, Software Engineering* (1.ª ed.). Association for Computing Machinery (ACM) IEEE Computer Society Association for Information Systems (AIS).

Kanban [URL https://kanban.university/resources/#official-kanban-guide. Último acceso: octubre 2022]. (s.f.).

Khakurel, J. y Porras, J. (2020). The Effect of Real-World Capstone Project in an Acquisition of Soft Skills among Software Engineering Students. *IEEE Conference on Software Engineering Education and Training (CSEE&T)*, 1-9.

Lethbridge, T. C., Sim, S. E. y Singer, J. (2005). Studying software engineers: Data collection techniques for software field studies. *Empirical Software Engineering*, *10*(3), 311-341.

Marques, M., Ochoa, S. F., Bastarrica, M. C. y Gutierrez, F. J. (2018). Enhancing the Student Learning Experience in Software Engineering Project Courses. *IEEE Transactions on Education*, *61*(1), 63-73.

Matthies, C., Huegle, J., Dürschmid, T. y Teusner, R. (2019). Attitudes, Beliefs, and Development Data Concerning Agile Software Development Practices. *IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 158-169.

Matturro, G., Raschetti, F. y Fontán, C. (2015). Soft Skills in Software Development Teams: A Survey of the Points of View of Team Leaders and Team Members. *International Workshop on Cooperative and Human Aspects of Software Engineering*, 101-104.

Moore, M. y Potts, C. (1994). Learning by doing: Goals and experiences of two software engineering project courses. *Conference on Software Engineering Education (CSEE)*, 151-164.

Object Management Group (OMG). (2008). *Software & Systems Process Engineering Meta-Model Specification V2.0* [URL http://www.omg.org/spec/SPEM/2.0/PDF. Último acceso: agosto 2022].

Ochoa, S. F., Pino, J. A., Guerrero, L. A. y Collazos, C. A. (2006). SSP: A Simple Software Process for Small-Size Software Development Projects. *Advanced Software Engineering: Expanding the Frontiers of Software Technology*, 94-107.

Paasivaara, M., Vanhanen, J. y Lassenius, C. (2019). Collaborating with Industrial Customers in a Capstone Project Course: The Customers' Perspective. *IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 12-22.

Pedrana, M. L. y Bellini, M. C. (2005). Mejora de la Calidad de los Procesos de Ingeniería de Software: Proceso Modularizado Unificado y Medible. Proyecto de Grado de la Carrera Ingeniería en Computación.

Rosnow, R. y Rosenthal, R. (1997). *People studying people: artifacts and ethics in behavioral research* (1.ª ed.). WH Freeman.

Schwaber, K. y Sutherland, J. (2020). *The Definitive Guide to Scrum: The Rules of the Game* [URL https://www.scrum.org/resources/scrum-guide. Último acceso: agosto 2022].

Sommerville, I. (2016). *Software Engineering* (10.ª ed.). Pearson Education.

Tafliovich, A., Estrada, F. y Caswell, T. (2019). Teaching Software Engineering with Free Open Source Software Development: An Experience Report. *IEEE Conference on Software Engineering Education and Training (CSEE&T)*, 7731-7741.

Tuzun, E., Erdogmus, H. y Ozbilgin, I. G. (2018). Are Computer Science and Engineering Graduates Ready for the Software Industry? Experiences from an Industrial Student Training Program. *IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 68-77.

Wells, D. (2013). *Extreme Programming: A gentle introduction* [URL http://www.extremeprogramming.org. Último acceso: octubre 2022].

ANEXOS

Anexo 1

Métricas, indicadores y medidas

Las métricas se dividen en tres tipos: esfuerzo, avance, y calidad del producto y del proceso. Si se desea conocer en mayor detalle las métricas, la definición de los artefactos asociados a cada una o visualizarlas con una recorrida más simple del contenido se puede consultar la especificación completa en repositorio Zenodo[1].

Las métricas de esfuerzo tienen por objetivo evaluar el esfuerzo medido en horas de trabajo (hst), estas son:

Métrica
Brecha del esfuerzo por iteración (y semanal) individual y grupal (medido en horas de trabajo (hst))

Frecuencia de evaluación
Al final de cada iteración y de cada semana.

Medidas asociadas
- Horas de trabajo (hst) individuales registradas en la ventana de tiempo que se está considerando.
- hst planificadas en cada ventana de tiempo considerada (plan)
- hst reales en cada ventana de tiempo considerada (real)

Registros y artefactos asociados
- Tabla de registro de esfuerzo
- Esfuerzo planificado de forma individual para cada semana y para la iteración

Cálculo de la métrica
- resultado = plan (hst) – real (hst)
- calcular a nivel grupal e individual

Análisis de la métrica
El objetivo es evaluar si la dedicación del equipo y a nivel individual es la adecuada. Para ello se evalúa la diferencia entre el esfuerzo que planificó dedicar el equipo durante la iteración en comparación al que dedicó realmente, de forma grupal e individual. En el caso de desvíos evaluar cuáles han sido las causas e identificar posibles ajustes. Para los ajustes se debe considerar la dedicación que debe tener semanalmente cada integrante al proyecto.

[1] https://doi.org/10.5281/zenodo.6730363

Métrica
Brecha del esfuerzo acumulado individual y grupal (medido en horas de trabajo (hst))

Frecuencia de evaluación
Al final de cada iteración.

Medidas asociadas
- Horas de trabajo (hst) individuales registradas en la ventana de tiempo que va desde el comienzo del proyecto hasta su medición
- hst acumuladas planificadas desde el comienzo del proyecto hasta su medición (planacu)
- hst acumuladas reales desde el comienzo del proyecto hasta su medición (realacu)

Registros y artefactos asociados
- Tabla de registro de esfuerzo
- Esfuerzo planificado de forma individual para cada semana y para la iteración

Cálculo de la métrica
- resultado = planacu (hst) – realacu (hst)
- calcular a nivel grupal e individual

Análisis de la métrica
El objetivo es evaluar si la dedicación del equipo es la adecuada. Para ello se evalúa la diferencia entre el esfuerzo que planificó dedicar el equipo desde el comienzo del proyecto a la fecha (planacu) en comparación al que dedicó realmente (realacu), de forma grupal e individual. En caso de desvíos, evaluar cuáles han sido las causas e identificar posibles ajustes. Para los ajustes se debe considerar la dedicación que debe tener semanalmente cada integrante al proyecto.

El avance del proyecto debe analizarse de varias perspectivas, en PISAgile una hace referencia al avance que se logró en determinada ventana temporal para lograr los objetivos del proyecto, y otra es para evaluar qué tan ajustado está estimando el equipo. Para medir el avance se proponen las siguientes métricas:

Métrica
Evolución del avance durante la iteración

Frecuencia de evaluación
Entre reuniones de sincronización.

Medidas asociadas
- Esfuerzo planificado para las tareas de los ítems del backlog de la iteración.
- Completitud de las tareas asociadas a los ítems del backlog de la iteración y de los ítems del backlog de la iteración.

Registros y artefactos asociados
- Ítems del Backlog de la iteración.
- Tablero de seguimiento de la iteración.

Cálculo de la métrica
- Gráfico del trabajo pendiente de la iteración.
- Gráfico de las horas de trabajo pendiente de la iteración.

Análisis de la métrica
En ambos gráficos se puede visualizar el trabajo remanente que se tiene en la iteración, para analizar si es necesario realizar ajustes de forma temprana, y cumplir con el objetivo de la iteración, el alcance definido y la aplicación del proceso.

Métrica
Evaluación del logro del alcance definido

Frecuencia de evaluación
Al final de cada iteración, a partir de que se defina el producto mínimo viable

Medidas asociadas
Trabajo (esfuerzo) completado en la iteración medido en puntos de esfuerzo (pe)

Registros y artefactos asociados
Gráfico de la velocidad del equipo.

Cálculo de la métrica
Gráfico del trabajo realizado para el producto final.

Análisis de la métrica
Permite visualizar el trabajo que resta realizar para lograr el alcance acordado.

Métrica
Tasa de cumplimiento del objetivo de la iteración

Frecuencia de evaluación
Al final de cada iteración.

Medidas asociadas
- Cantidad de ítems del backlog de la iteración comprometidos (#IBIcomp)
- Cantidad de ítems del backlog de la iteración aceptados por el Dueño del producto durante la revisión (#IBIacep)

Registros y artefactos asociados
- ítems del Backlog de la iteración
- Tablero de seguimiento de la iteración

Cálculo de la métrica
resultado = (#IBIacep) / (#IBIcomp)

Análisis de la métrica
El indicador del resultado esperado es 1. Si el resultado de la métrica corresponde con el valor del indicador, implica que se cumplió con el objetivo de la iteración y el alcance acordado con el Dueño del producto. En otro caso, esto no fue así y se deben evaluar las causas que llevaron al no cumplimiento del objetivo y en la aceptación de los ítems por parte del Dueño del producto, para identificar oportunidades de mejora vinculadas al proceso y al producto.

Métrica
Evolución de la Velocidad del equipo

Frecuencia de evaluación
Al final de cada iteración

Medidas asociadas
Sumatoria de los puntos de esfuerzo (pe) de todos los ítems del backlog de la iteración aceptados durante la revisión por el Dueño del producto.

Registros y artefactos asociados
- ítems del backlog de la iteración
- Tablero de seguimiento de la iteración (kanban)

Cálculo de la métrica
Gráfico de la velocidad del equipo

Análisis de la métrica
El objetivo es poder visualizar la evolución de la Velocidad del equipo a lo largo de las iteraciones. Se debe visualizar y analizar de forma conjunta la velocidad estimada y real por cada iteración. Con esta información el equipo podrá estimar al velocidad a la cual se puede comprometer para la siguiente iteración. Es esperable que cuando un equipo es nuevo, en las primeras iteraciones la fluctuación en la velocidad del equipo sea grande, hasta que logra acomodarse. La finalidad de ello es poder ir mejorando el ajuste de la estimación y por consiguiente en la planificación y compromiso con el cliente.

Métrica
Porcentaje de desvío de la estimación (individual) - Métrica opcional

Frecuencia de evaluación
Al final de cada iteración.

Medidas asociadas
- Horas de trabajo (hst)
- Plan (hst): Horas que se estiman dedicar a las tareas asignadas
- Real (hst): Horas dedicadas realmente a las tareas asignadas

Registros y artefactos asociados
- ítems del backlog de la iteración
- Tabla de registro de esfuerzo

Cálculo de la métrica
- totalplan(hst): Sum (Plan ((hst))
- totalreal(hst): Sum (Real (hst))
- resultado = (totalplan(hst) – totalreal(hst)) * 100/ totalplan(hst)

Análisis de la métrica
El objetivo de esta métrica es conocer el desvío que se ha tenido en las estimaciones de las tareas a nivel individual para poder realizar los ajustes necesarios para la próxima iteración. Si el resultado es negativo es porque se planificó un esfuerzo menor al que llevó realmente, si es positivo caso contrario.

Las métricas que se deben considerar para evaluar causas y mejoras en la calidad del producto y del proceso son cuantitativas y cualitativas. Se proponen las siguientes:

Métrica
Evolución en la cantidad de defectos encontrados y removidos por iteración - vista por iteración y agregada

Frecuencia de evaluación
Al final de cada iteración.

Medidas asociadas
- Cantidad de defectos encontrados por iteración (clasificados por tipo y severidad)
- Cantidad de defectos removidos por iteración (clasificados por tipo y severidad)

Registros y artefactos asociados
Registro o reporte de incidentes

Cálculo de la métrica
Tabla que presente la cantidad de defectos inyectados y removidos por iteración clasificados por tipo y severidad

Análisis de la métrica
La cantidad de defectos es una muestra de la calidad del producto y de la eficacia del proceso de pruebas durante el proceso de desarrollo. Permite conocer la cantidad de defectos remanentes identificados en el Incremento. Hay que tener cuidado con la evaluación de esta métrica ya que un aumento en la cantidad de defectos encontrados puede interpretarse como una disminución en la calidad del producto, sin embargo puede surgir de una mejora en el proceso de pruebas (ej: encuentro un mayor número de defectos).

Métrica
Cumplimiento de cobertura de pruebas

Frecuencia de evaluación
Al final de cada iteración

Medidas asociadas
Porcentaje (%) de cobertura en las pruebas. Este porcentaje puede estar definido por tipo de pruebas y/o componentes del incremento

Registros y artefactos asociados
- Cobertura de las pruebas realizadas en la iteración.
- Indicador de cubrimiento aceptado definido por el Dueño del producto.

Cálculo de la métrica
Ejecutar las pruebas y obtener el porcentaje (%) de cubrimiento

Análisis de la métrica
El indicador del resultado esperado debe ser acordado con el cliente. Si el resultado del cálculo de la métrica es mayor o igual al indicador, se cumplió con el cubrimiento acordado. Caso contrario no se cumplió.

Métrica
Termómetro de percepción

Frecuencia de evaluación
Al final de cada iteración

Medidas asociadas
Nivel de percepción de cada integrante de 0 a 10 para la aplicación proceso, otro para la calidad del producto y funcionamiento de equipo

Registros y artefactos asociados
-

Cálculo de la métrica
Cada integrante debe evaluar su percepción sobre ciertos aspectos vinculados al proyecto y luego se promedian para tener una visión global. El termómetro se particiona en tres sub-termómetros: uno para evaluar la aplicación del proceso, otro para la calidad del producto y otra para el funcionamiento de equipo. El rango va de 0 a 10, donde 0 es que la percepción es muy mala y 10 excelente. A los termómetros para representarlos se le asignan rangos de colores: de 0 a 4 (rojo), 5 a 7 (amarillo) y de 8 a 10 (verde).

Análisis de la métrica
El objetivo de esta métrica es poder evaluar las percepción que tiene el equipo en comparación con la evidencia que se tiene de la iteración. El resultado obtenido se comparará con la evidencia y de allí se debe analizar diferencias y similitudes entre los resultados de las métricas.

Métrica
Aplicación del Plan de mejora de la iteración

Frecuencia de evaluación
Al final de cada iteración

Medidas asociadas
- Cantidad de acciones comprometidos (#AccComp)
- Cantidad de acciones ejecutadas (#AccEj)

Registros y artefactos asociados
- Plan de mejora de la retrospectiva.
- Acciones ejecutadas del plan de mejora.

Cálculo de la métrica
- resultado cuantitativo= ((#AccComp – #AccEj) * 100)/ #AccComp
- resultado cualitativo = percepción de las mejora con las acciones.

Análisis de la métrica
El objetivo es evaluar el cumplimiento del Plan de mejora definido en la iteración inmediata anterior (i-1), para determinar si se cumplieron las acciones identificadas y si el efecto fue el esperado. De no haberse cumplido con las acciones identificadas determinar causas.

Anexo 2

Fases y sus artefactos

Se presentan los artefactos destacables de cada una de las fases: asociados de forma directa al producto de software o el propio producto (ver tabla, 2.1), asociados al proceso y el proyecto, y su ejecución (ver tabla, 2.2), y más relevantes para el seguimiento y reporte (ver tabla, 2.3).

Tabla 2.1: Artefactos asociados directamente al producto de software

Artefactos	FI	FC	FT
Visión global del producto	■	■	■
Prototipo	■		
Producto final (software y otros artefactos)			■
Descripción del problema/necesidad	■		
Lista de criterios de aceptación generales	■	■	■
Producto mínimo viable (PMV)		■	
Plan de liberación		■	■
Incremento de la iteración (software y otros artefactos)		■	■
Microincremento		■	■
Artefactos entregables internos al equipo y no entregables		■	■
Conceptualización técnica	■		
Conceptualización del diseño de la interacción	■		
Descripción del producto de software (o sistema)			■
Descripción/diagramas del diseño de la arquitectura y técnico		■	■
Descripción/diagramas del diseño de la interacción de usuario		■	■

(FI: Fase Inicial, FC: Fase Construcción, FT: Fase Transición)

Tabla 2.2: Artefactos asociados al proceso y el proyecto

Artefactos	FI	FC	FT
Propuesta de proyecto	■		
Trade off sliders	■	■	■
Backlog del producto (bp)	■	■	■
Objetivo de la iteración		■	
Backlog de la iteración (bi)		■	■
Definición de listo (del inglés, definition of ready (DoR))	■	■	■
Definición de hecho (del inglés, definition of done (DoD))	■	■	■
Acuerdos de equipo	■	■	■
Plan de acciones y actividades de equipo	■	■	
Acuerdos de trabajo	■	■	■
Estrategia y roadmap de gestión de proyecto	■	■	■
Estrategia y roadmap de calidad del producto y verificación	■	■	■
Estrategia y roadmap de gestión de la configuración	■	■	■
Documento de riesgos	■	■	■
Plan de la iteración		■	
Plan de mejora de la iteración		■	
Plan y cronograma de transferencia			■
Plan de fase de transición			■
Herramientas de soporte	■	■	■

(FI: Fase Inicial, FC: Fase Construcción, FT: Fase Transición)

Tabla 2.3: Artefactos más relevantes para el seguimiento y reporte

Artefactos	FI	FC	FT
Backlog del producto (bp)	■	■	■
Backlog de la iteración (bi)		■	■
Informe semanal	■	■	■
Informe de la iteración		■	
Tabla de registro de esfuerzo	■	■	■
Tablero de seguimiento de la iteración	■	■	■
Métricas		■	■
Reporte de incidencias		■	■
Gráfico del trabajo pendiente de la iteración		■	■
Gráfico de las horas de trabajo pendiente de la iteración		■	■
Gráfico del trabajo realizado para el producto final		■	■
Gráfico de la velocidad del equipo		■	■
Informes finales de todas las estrategias			■
Encuesta de satisfacción al cliente			■
Lecciones aprendidas			■
Presentación final			■

(FI: Fase Inicial, FC: Fase Construcción, FT: Fase Transición)