Adopción de prácticas ágiles de desarrollo de software en los planes de estudio de universidades de Costa Rica

Carlos Martín Flores González

Carné: 2015183528

Profesor: Rodrigo Bogarin

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Maestría en Computación

Ingeniería de Software

16 de noviembre del 2017

Índice general

Índice de cuadros

Glosario

Cenfotec Universidad Cenfotec. 12

Invenio Universidad Invenio. 12

TEC Instituto Tecnológico de Costa Rica. 12

UACA Universidad Autónoma de Centroamérica. 12

UAM Universidad Americana. 12

UCEM Universidad de Ciencias Empresariales. 12

UCR Universidad de Costa Rica. 12

UIA Universidad Internacional de las Américas. 12

UISIL Universidad Internacional San Isidro Labrados. 12

ULACIT Universidad Latinoamerica de Ciencia y Tecnología. 12

UMCA Universidad Metropolitana Castro Carazo. 12

UNA Universidad Nacional. 12

UNADECA Universidad Adventista de Centro América. 12

UNED Universidad Estatal a Distancia. 12

UTN Universidad Técnica Nacional. 12

Resumen

A pesar de la popularidad de las metodologías ágiles de desarrollo de software en la industria, expresada en varias encuestas y estudios recientes, muchas organizaciones siguen reportando en estas mismas encuestas que se cuenta con poco personal capacitado con las habilidades necesarias y que los planes de estudio de carreras de tecnología de información muestran rezagos en estos temas. Con el fin de brindar una formación que no solamente incluya la enseñanza de prácticas ágiles sino también el desarrollo de otras actitudes como el trabajo en equipo y el empredurismo, las universidades han venido introduciendo paulativamente estas prácticas a sus planes de estudio principalmente por medio de enfoques de *aprender-haciendo*. Universidades en Costa Rica también han reportado la inclusión de estos temas en sus planes de estudio como respuesta a las tendencias mundiales y para apoyar la creciente industria de desarrollo de software del país.

Abstract

Despite the popularity of the agile engineering practices in the software industry reported in recent surveys and studies, many organizations keep reporting in those same surveys which they have no enough personal with the desired skill set, and also that they think that IT curricula is behind in these new software development trends. In order to provide an education which involves agile engineering practices but also other skills such as teamwork and entrepreneurship, universities have been introducing these practices in their curricula, mainly by the use of *learning-by-doing* approaches. Universities in Costa Rica have reported the introduction of these topics in their curricula as a response to their increasing popularity and to support the growing software industry in the country.

Palabras Clave / Keywords

agile software development, agile teaching, agile engieering practices, software engineering education, costa rica software education

Introducción

En las últimas décadas se ha hecho un esfuerzo significativo en identificar buenas prácticas, modelos y métodos que conduzcan a desarrollar software de forma más eficiente. Los desarrolladores de software tienden a clasificar las metodologías de desarrollo en dos categorías [1]:

1. Metodologías de desarrollo clásicas: requieren definición de requerimientos por adelantado, documentación y planes detallados. Dos ejemplos relevantes de son el modelo de cascada y espiral.

2. Metodologías ágiles: a menudo se llama "livianas". Esta categoría incluye *eXtreme Programming* (XP) y *Scrum*.

El movimiento de desarrollo ágil de software nace como una alternativa a los métodos tradicionales con los que se hace software, en los cuales los ciclos de desarrollo y entrega tienden a ser muy prolongados. El desarrollo ágil de software se define en el Manifiesto Ágil [2] como un conjunto de doce principios.

Reportes sobre el uso de metodologías ágiles en desarrollo de software muestran un sostenido crecimiento a través de los años [3] así como la literatura de investigación con respecto al impacto de estos enfoques en varios aspectos del ciclo de vida de desarrollo como lo puede ser la calidad, la entrega, gestión de requerimientos, entre otros.

Otra práctica que ha resultado tener gran impacto en el sector del desarrollo de software en los años muy reciente es DevOps. Esta práctica promueve retomar viejos paradigmas de trabajo en donde el ingeniero/programador/investigador, estaba a cargo de todo el proceso de desarrollo: desde el diseño y la programación hasta la generación de pruebas, artefactos de software e instalación final. DevOps ha ganado mucho terreno gracias a la computación en la nube en donde en lugar de estar a cargo de recursos físicos, se está a cargo de recursos virtuales por lo que los roles del programador que solo programa, el ingeniero de pruebas que solo prueba software y finalmente el administrador de servidores

que solamente se dedica a configurar hardware, se motivan a ser re-pensados. En ambientes como el de la computación en la nube, la infraestrutura está en el código lo que hace que al trabajar con este tipo de aplicaciones se tiene que estar consciente de muchas más cosas. De esta forma se puede decir que DevOps es la combinación entre desarrollo y operaciones.

A pesar de ser un enfoque de desarrollo aún más reciente que las metodologías ágiles, DevOps ha sido influenciado por muchos de los principios de estas, principalmente del *eXtreme Programming*. DevOps es visto por muchos como una extensión natural de las prácticas ágiles [4] y actualmente se puede encontrar mucha literatura disponible sobre la aplicación de metodologías ágiles junto con DevOps.

Sin embargo la rápida penetración y auge de estas prácticas viene con un precio a pagar. En los primeros años de adopción de las metodologías ágiles era muy difícil encontrar personal con las habilidades necesarias, esto forzaba tanto a empleados como a empleadores a buscar formas alternas de formación. Por otro lado, los graduados de carreras de tecnologías de información del momento tampoco contaban con tales conocimiento debido a que los planes de estudio abarcaban poco o nada de estas prácticas. En los círculos académicos no había mucho interés en desarrollar estos temas debido a que se consideraba que desarrollo ágil no tenía bases teóricas sino que más bien ha sido desarrollada a partir de la práctica y la experiencia [5]. Esto ha venido cambiando de forma substancial y ahora temas sobre metodologías ágiles se han incorporado a los planes de estudio de carreras de tecnologías de la información de formas diversas. Aún se acusan muchos retos asociados a la enseñanza de estas prácticas, tal y como se indica en la Sección 2.5.3.

Con el paso de los años se ha constatado que estas prácticas han dejado de ser una moda o una tendencia del momento y que paralelo a fomentar mejoras en aspectos propios de ingeniería, también promueven el trabajo en equipo y colaborativo, habilidades de gestión, normas éticas y emprendedurismo [5, 6].

Para apoyar la mejora de habilidades técnicas y la inserción laboral, varios enfoques de enseñanza de ingeniería de software utilizando prácticas ágiles han sido reportados, [7–13]. Estas iniciativas persiguen exponer a los estudiantes a ambientes de trabajo similares a los que se encontraran en su práctica profesional y a la mejora de sus habilidades en desarrollo de software. Cobra aún mayor importancia la exposición de estos temas en la universidad si se toma en cuenta el gran auge de la computación en la nube. Baker [14] señala que *"Los estudian-*

tes de hoy en día puede ser que nunca se enfrenten a un servidor físico, un switch de red o un dispositivo de almacenamiento durante su carrera profesional futura pero aún así van a estar construyendo más infraestructura de IT que todos sus predecesores combinados. Se necesita enseñar a los estudiantes en cómo crear infraestructuas usando códido y cómo trabajar juntos de forma colaborativa para gestionar aplicaciones distribuidas complejas".

En Costa Rica, la industria de desarrollo de software ha tenido un gran crecimiento en los últimos 20 años [15, 16]. A pesar del crecimiento, las organizaciones reportan deficiencias en la formación de la fuerza laboral y en los planes de estudio que, según señalan, no están actualizados con las necesidades del sector [16]. Con respecto a la enseñanza de prácticas ágiles, las iniciativas en [17–20] dan a conocer enfoques en cómo se están introduciendo a los estudiantes en estos temas. Se destacan los enfoques de enseñanza orientados a resolución de problemas y proyectos.

Capítulo 1

Generalidades de la Investigación

1.1. Marco de Referencia Contextual

En el nivel más amplio, esta propuesta se basa en la interacción de dos áreas:

- planes de estudio: que abarcan la formalización de conocimiento
- prácticas ágiles de desarrollo de software: transmisición y entrenamiento de conocimiento de la industria

1.1.1. Planes de Estudio

Existen varias organizaciones profesionales preocupadas de la formación académica en carreras de tecnología de la información:

- *The Association for Computing Machinery* (ACM)
- *The Institute for Electrical and Electronic Engineers*, en particular la Sociedad de Computación (*Computer Society* – IEEE-CS)
- *The Association for Information Systems* (AIS)

El ACM cumple el rol más amplio, es la organización presente en una gran variedad de guías para educadores que va desde el kinder hasta universidad. La ACM señala en [21] *"A partir de la década de los 1960's, la ACM juntos con sociedades profesionales y científicas líderes en computación, se han dado a la tarea de reunir una serie de recomendaciones para el panorama rápidamente cambiante de la tecnología en computación"*.

En el 2005, un grupo de trabajo conjuntado por la ACM, AIS y el IEEE-CS establecieron los límites entre varios campos de la tecnología en un reporte titulado: *Computing Curricula 2005: The Overview Report* [22]. Las disciplinas identificadas en el reporte de 2005 han sido usado como base para la organización de mucho planes de estudio en muchos países:

1. Ciencias de la Computación (*Computer Science*)
2. Ingeniería de Computadores (*Computer Engineering*)
3. Sistemas de Información (*Information Systems*)
4. Tecnología de la Información (*Information Technology*)
5. Ingeniería de Software (*Software Engineering*)

De las anteriores, las disciplinas 1-3 se conocen desde hace muchas décadas atrás, mientras que 4-5 son más recientes. En el reporte del 2005 no se afirma que estas fueran las únicas, en el mismo de preveía la aparación de nuevas disciplinas en computación.

Las disciplinas se describen en detalle en el reporte y a continuación se extraen algunas definiciones:

Ingeniería de Computadores *"abarca el diseño y construcción de computadoras y sistemas basados en computadoras. Involucra el estudio de hardware, software, comunicaciones y la interacción entre ellos... su estudio puede hacer mayor énfasis en hardware que en el software o podría existir un énfasis balanceado."* (página 13)

Ciencias de la Computación *"abarca un amplio rango que va desde bases teóricas y algorítmicas a desarrollos de vanguardia en robótica, visión computacional, sistemas inteligentes, bioinformática... Mientras otras disciplinas puede producir graduados con habilidades más relevantes para el trabajo, ciencias de la computación ofrece una base detallada que permit a los graduados adaptarse a nuevas tecnologías e ideas"* (páginas 13-14)

Ingeniería de Software *"es la disciplina del desarrollo y mantenimiento de sistemas de software que se comportan de forma confiable y eficientemente, que tienen costos razonables de desarrollo y mantenimiento y satisfacen todos los requerimiento que los clientes han definido... ha evolucionado como respuesta a*

factores tales como el crecimiento e impacto de grandes sistemas de software en una amplia variedad de situaciones y la creciente importancia de aplicaciones de software críticas. La ingeniería de software es diferente de otras disciplinas de ingeniería debido a ambos, la naturaleza intangible del software y la naturalize discontinua de la operación del software…" (página 15)

Sistemas de Información *"se concentra en la integración de soluciones de tecnología de información y procesos de negocio para cumplir las necesidades del negocio y otras empresas…La mayoría de los programas en sistemas de información se encuentran en escualas de negocios. Todos los grados de sistemas de información combinan negocios y un conjunto de cursos en computación. Existe una variedad de programas en sistemas de información que tienen una mayor concentración sistemas de información computacionales, mientras que otros programas tienen una mayor énfasis en administración de sistemas de información. Hoy en día los nombres de los programas no son siempre consistentes"* (página 14)

Tecnología de la Información es vista como la inversa de sistemas de información: *"su énfasis es en la tecnología como tal más que en la información que tiene y transmite…las organizaciones de cualquier tipo son dependientes de tecnología de información. Necesitan tener los sistemas apropiados trabajando. Esos sistemas deben trabajar apropiadamente, ser seguros, actualizados, mantenidos, reemplazados. Los empleados en las organizaciones requieren soporte del personal de IT para entender sistemas de computadora y sus software, y están comprometidos a resolver cualquier problema relacionado con computadoras. Los graduados de tecnología de información son los que abarcan estas necesidades"* (página 14)

Planes de estudio en consideración

De las cinco grandes disciplinas mencionadas en la sección 1.1.1, esta propuesta se enfocará en:

- Ciencias de la Computación
- Ingeniería de Software
- Sistemas de Información

- Tecnología de la Información

y estará menos enfocado en ingeniería de computadores, el cual estará fuera del alcance del resto de este documento. En la sección 1.5 se brindan los planes de estudio a considerar para el caso de la universidades de Costa Rica.

1.1.2. Desarrollo de Software Ágil

Este movimiento se inicia a principios de la década del 2000 como alternativa a métodos de desarrollo tradicionales a los cuales se les atribuyen la realización de muchos procesos los cuales impiden con la implementación pronta de los requirimientos.

Este nuevo estilo se basa en desarrollo basado en pruebas incremental e iterativo y refactorización constante para prevenir la deuda técnica. Requiere una integración continua de código, automatización de la infraestructura, capacidad para escalar de forma dinámica y sistemas débilmente acoplados.

La segregación de funciones especializada es reemplazada por colaboración multifuncional, llevada a cabo por equipos de no más de 8 personas centrados en la implementación de un producto.

En estos esquemas de trabajo, los empleadores buscan lo que se le conoce como los profesionales tipo "T", que son más flexibles en los roles que desempeñan y esto permite que los equipos se muevan más rápido. *El principio de los recursos tipo T: desarrollar personas que tienen conocimiento profundo de un área y otro más amplio en otras.* La importancia de equipos que tienen altos niveles de confianza está ganando mucha atención debido al impacto cuantificable que esto tiene en la entrega de un producto. [23]

DevOps es otra tendencia, altamente ligada con los principios del desarrollo ágil, que ha ganado mucha popularidad en los últimos años. Informes como [24] muestran que las prácticas de desarrollo ágil pueden ser parte de esquemas de entrega continua, ya que tienen una relación significativa con el éxito de la puesta en producción, rendimientos y tasas de error.

Prácticas ágiles de desarrollo en consideración

El conjunto de prácticas a tomar en consideración en esta propuesta abarcan:

- Pruebas

- Integración continua
- Estándares de codificación
- Refactorización de código
- Puesta en producción continua
- Propiedad compartida del código
- Diseño emergente

Estas se detallan en la sección 2.2.2 y se obtuvieron como parte de la revisión de las prácticas más populares reportadas por profesionales de la industria.

Aunado a lo anterior, para efectos de esta propuesta se va a considerar DevOps como una práctica de desarrollo ágil a evaluar, esto porque como se menciona en la sección 2.2.4 representa el siguiente paso natural en el movimiento de desarrolo ágil.

1.2. Justificación

Diversas organizaciones de profesionales y académicos se han dado a la tarea de revisar los planes de estudio de las universidades con el fin de evaluar su relevancia y pertinencia con tendencias en la industria. En el reporte de [22] se señala que *"algunos programas en TI tienen baja calidad y fallan servir tanto a sus estudiantes como a sus comunidades en una forma responsable. Existen programas en sistemas de información y tecnología de la información que solamente buscan aumentar las matriculas y/o proporcionar la imagen de ser receptivo a las necesidades locales mediante la creación de un programa de TI que es poco más que el reempaquetado de los cursos existentes ofrecidos en otras disciplinas"*. En otros estudios como [14, 25, 26] evidencian carencias en los profesionales recién graduados en cuanto al conocimiento e implementación de prácticas ágiles de desarrollo de software. Aunado a estas carencias, se evidencia de igual forma descontento de los empleadores con las universidades porque se considera que estas no están brindando conocimiento alineado con las tendencias del sector.

El desarrollo de software es un tarea desafiante, usualmente requiere de un conocimiento que va más allá de un lenguaje de programación y de la programación misma y que tiene que ver más procesos de ingeniería asociados al aseguramiento de un producto de calidad. En este sentido, el conocimiento

de prácticas ágiles de desarrollo de software contribuyen con esta visión, son prácticas que se han desarrallado a lo largo del tiempo y que han probado ser efectivas pero que requiere de tiempo y madurez para que se aprendan y apliquen de manera efectiva.

Costa Rica no es ajeno a este panorama. En informes tales como [16] y [27] se señala también el descontento tanto de las habilidades de personal de ingeniería/desarrollo de software como de la formación de las universidades del país con respecto a la enseñanza de temas más importantes para la industria. Igual de importante es saber que actualmente no se cuenta tampoco con ningún tipo de informe sobre la adopción de prácticas ágiles de desarrollo de software o sobre adopción desarrollo ágil en general en el país que pueda servir como referencia para académicos y empleadores sobre en qué poner mayores esfuerzos.

Un estudio exploratorio sobre la adopción de prácticas ágiles de desarrollo de software en los planes de estudio de universidades de Costa Rica podría servir como punto de partida para evaluar la relevancia que se le dan a la enseñanza de estos temas para que de esta forma se pueda comparar su pertinencia con el estado reportado en informes internacionales y realizar esfuerzos en pos de la mejora de su adopción.

1.3. Planteamiento del Problema

Las prácticas ágiles de desarrollo de software están cambiando las formas tradicionales de hacer y entregar software. Estas prácticas gozan de una gran aceptación y auge dentro la industria del software desde hace más de una década atrás. En Costa Rica, a pesar de contar con una creciente industria de software, encuestas y estudios revelan que los empleadores señalan deficiencias en las habilidades de desarrollo ágil como programación, pruebas, integración y entrega. Aunado a lo anterior se acusa un rezago en los planes de estudio de las carreras de tecnología de la información con respecto a las necesidades de la industria.

1.4. Objetivos

1.4.1. Objetivo General

Explorar la penetración de prácticas ágiles de desarrollo de software en los planes de estudio de las carreras en tecnología de la información en Costa Rica.

1.4.2. Objetivo Específicos

1. Dar a conocer las prácticas ágiles de desarrollo de software de mayor adopción en la industria

2. Identificar la aplicación de prácticas ágiles de desarrollo de software como parte de la enseñanza de carreras de tecnología de información en Costa Rica.

3. Exponer causas por medio de las cuales la enseñanza de prácticas ágiles de desarrollo de software se ve impactada tanto de forma positiva como negativa en los centros de estudio en donde se imparte carreras de tecnología de la información en Costa Rica.

1.4.3. Preguntas de Investigación

Preguntas de investigación asociadas a los objetivos específicos:

- ¿Qué son prácticas ágiles de desarrollo de software? (Objetivo específico 1)
- ¿Cuáles son las prácticas ágiles de desarrollo de software que gozan de mayor aceptación en la industria? (Objetivo específico 1)
- ¿Cuáles prácticas ágiles de desarrollo de software se enseñan durante la carrera? (Objetivo específico 2)
- ¿Cómo se implementan estas prácticas de desarrollo ágil en los planes de estudio? (Objetivo específico 3)
- ¿Qué beneficia o perjudica la enseñanza de nuevas habilidades en programación, pruebas, integración, entrega y qué impacto genera esta enseñanza en aspectos tales como colaboración, generación de mejores hábitos de

programación, emprendedurismo e inserción laboral? (Objetivo específico 3)

1.5. Alcance y Limitaciones

1.5.1. Alcance

Para la realización de este estudio se pretende analizar los planes de estudio de las carreras en Ingeniería de Sistemas, Computación e Informática aprobadas por el Consejo Superior de Educación (CONESUP) de Costa Rica. En el Cuadro 1.1 se listan las carreras aprobadas por el CONESUP en estas áreas, al momento de escribir esta propuesta.

En grado mínimo a considerar es el de Bachillerato. De acuerdo con el listado del Cuadro 1.1 existen dos carreras cuyo título mínimo de salida es Licenciatura, los cuales pertenence a las carreras de Tecnologías de la Información y Comunicación Empresarial de la Universidad INVENIO y Administracion de Tecnología de Información del TEC. Se propone inicialmente poner una mayor atención a aquellas carreras que están certificadas por el Sisteman Nacional en Acreditación de la Educación Superior (SINAES) – 8 en total al momento de escribir esta propuesta – puesto que esta certificación es evidencia de compromiso con altos estándares de calidad. Las carreras acreditadas por el SINAES [28] se listan en el Cuadro 1.2

1.5.2. Limitaciones

- Dificultad en la evaluación de la aplicación de las prácticas ágiles de desarrollo en universidades que impartan la carrera en varias sedes. Si bien a las universidades acreditadas en SINAES se les exige impartir la carrera de la misma forma en todas las sedes acreditadas, existen otras universidades que importan la carrera en otras sedes y no se encuentra acreditadas. Esto podría hacer que tanto los planes como su implementación varíen en cada sede aún estando en la misma universidad. Un ejemplo de esto es el caso de la Universidad Latina en donde la carrera de Ingeniería en Sistemas Computacionales, que se brinda en la sede de Heredia, se encuentra acreditada por SINAES, pero la carrera de Ingeniería en Sistemas Informáticas del resto de las sedes no se encuentra acreditada.

Universidad	Carrera
UNADECA	Bachillerato en Ingeniería de Sistemas
UAM	Bachillerato en Ingeniería de Sistemas
UACA	Bachillerato en Ingeniería de Sistemas
Católica de Costa Rica	Bachillerato en Ingeniería de Sistemas
Cenfotec	Bachillerato en Ingeniería del Software
Cenfotec	Bachillerato en Ingeniería en tecnologías de Información y Comunicación
Central	Bachillerato en Ingeniería Informática
UCEM	Bachillerato en Ingeniería de Sistemas
Federada de Costa Rica	Bachillerato en Sistemas de Computación
Fidélitas	Bachillerato en Ingeniería de Sistemas de Computación
Hispanoamericana	Bachillerato en Ingeniería Informática
UIA	Bachillerato en Ingeniería de Software
UIA	Bachillerato en Ingeniería en Informática
UIA	Bachillerato en Ingeniería en Sistemas de Información
UISIL	Bachillerato en Ingeniería de Sistemas
Invenio	Licenciatura en Tecnologías de la Información y Comunicación Empresarial
Latina de Costa Rica	Bachillerato en Ingeniería de Sistemas Informáticos
Latina de Costa Rica	Bachillerato en Ingeniería de Sistemas Computacionales
Latina de Costa Rica	Bachillerato en Ingeniería del Software
ULACIT	Bachillerato en Ingeniería Informática
Magister	Bachillerato en Ingeniería de Sistemas
UMCA	Bachillerato en Ingeniería Informática
Panamericana	Bachillerato en Sistemas de Computación
Politécnica Internacional	Bachillerato en Ingeniería Informática
Tecnológica Costarricense	Bachillerato en Ingeniería en Sistemas Computacionales
TEC	Bachillerato en Ingeniería en Computación
TEC	Licenciatura en Administracion de Tecnología de Información
UCR	Bachillerato en Ingeniería en Computación e Informática
UCR	Bachillerato en Informática Empresarial
UNA	Bachillerato en Ingeniería en Sistemas de Información
UNED	Bachillerato en Ingeniería informática
UTN	Bachillerato en Ingeniería del Software

Cuadro 1.1: Listado de carreras tomadas en cuenta para la propuesta. Fuente: CONESUP

- Resistencia por parte de las autoridades de las universidades consultadas: debido que este propuesta pretende evaluar el grado de penetración de prácticas ágiles de desarrollo en sus planes de estudio, pueden existir escenarios en donde el personal consultado no desee compartir o cooperar con lo que se le pide porque a pesar de tener nombres tales como “Tecnología de la Información”

Universidad	Carrera
Fidélitas	Bachillerato en Ingeniería de Sistemas de Computación
Latina de Costa Rica	Bachillerato en Ingeniería de Sistemas Computacionales
ULACIT	Bachillerato en Ingeniería Informática
TEC	Bachillerato en Ingeniería en Computación
TEC	Licenciatura en Administracion de Tecnología de Información
UCR	Bachillerato en Ingeniería en Computación e Informática
UNA	Bachillerato en Ingeniería en Sistemas de Información
UNED	Bachillerato en Ingeniería informática

Cuadro 1.2: Listado de carreras acreditadas por el SINAES. Fuente: SINAES

- Poco interés por parte de entidades académicas o de la industria en llevar a cabo un estudio de este tipo. Falta de patrocinio y recursos en general para llevar el estudio a cabo.

- A pesar del interés por llevar a cabo el estudio, varias carreras de "Tecnología de Información" e "Sistemas de Información" van a quedar fuera del ámbito de esta propuesta: esto se da porque como se destaca en la sección 1.1.1, existen carreras que incluyen en su nombre estas palabras pero que sus planes de estudio están más vinculadas con la integración de la tecnología en procesos de negocio.

1.6. Beneficiarios de la Investigación

- **Universidades:** porque les puede brindar una referencia de lo que están impartiendo con respecto a lo que se hace en la industria. Las prácticas ágiles de desarrollo de software han mostrado ser algo más que una "moda" y hoy gozan de gran aceptación dentro de la comunidad profesional de software. Su enseñanza y divulgación puede no solamente motivar a los estudiantes a adquirir nuevas habilidades técnicas, sino que estas también introducen el desarrollo de otras habilidades personales como lo puede ser el trabajo en equipo y emprendedurismo.

- **Industria de desarrollo de Software:** porque a partir de los resultados de un estudio de este tipo podrían tanto identificar las universidades que formen profesionales con los conocimientos que estos requieren y al mismo tiempo contribuir con aquellas en las que se haya identificado que brindan menor formación en estos temas. El contar con profesionales mejor

formados en temas de desarrollo de software mejorará la calidad de sus proyectos y aumentará la competitividad del sector tanto a nivel nacional como internacional.

- **Aspirantes y estudiantes:** van a poder evaluar qué carreras les brinda la oportunidad de explotar más las habilidades en prácticas ágiles. También los estudiantes activos de las carreras van a poder saber la situación de su carrera con respecto de las demás y de acuerdo con esto proponer posibles ajustes junto con sus profesores o bien, estudiar sobre la posibilidad de desarrollo que les podría dar estas habilidades para sus actividades académicas como para incorporarse al mercado laboral por su cuenta.

Capítulo 2

Marco Teórico

2.1. Revisión de la Literatura

2.1.1. Estrategia

Esta sección presenta la estrategia emprendida para cubrir el cuerpo de conocimiento relacionado con enseñanza de prácticas ágiles de desarrollo de software en universidades. La estrategia general se basó en un proceso iterativo de identificación y lectura de artículos, luego identificar y leer artículos relevantes a partir de referencias y citas bibliográficas.

2.1.2. Identificación de Preguntas de Investigación

Las preguntas de investigación seleccionadas para conducir la revisión de la literatura fueron:

1. ¿Cuáles son las prácticas de desarrollo ágil con mayor aceptación y qué relevancia tienen en la industria?

2. ¿Se podría considerar a DevOps como una práctica de desarrollo ágil o representa más bien un enfoque diferente?

3. ¿Qué información hay disponible acerca de de la enseñanza de prácticas ágiles de desarrollo de software carreras universitarias de TIC's en Costa Rica como en el extranjero?

4. ¿Cuáles son los beneficios y retos reportados en la enseñanza de estas prácticas de desarrollo ágil?

2.1.3. Estrategia de búsqueda

Con el fin de identificar el primer conjunto de artículos relevantes, se hizo una revisión preliminar con Google Scholar[1] porque con este motor de búsqueda se puede abarcar una amplio número de artículos y actas académicas de diferentes fuentes. El criterio de búsqueda se basó en búsquedas de palabras derivadas del tema de investigación y las preguntas de investigación. Se incluyeron palabras como “agile development”, “agile engineering practices”, “agile teaching”, “devops education”, “computer science education” y “software engineering education”.

La búsqueda de la literatura se realizó en Setiembre del 2017 usando las siguientes bases de datos electrónicas:

- ACM *Digital Library*
- IEEE *Explore Digital Library*
- Safari Books Online[2]
- Google Scholar

2.1.4. Criterio de selección de artículos

Se aplicó el siguiente criterio de inclusión de artículos para esta revisión:

- Estudios que den a conocer prácticas ágiles de desarrollo de software y
- Estudios sobre la relación de prácticas ágiles de desarrollo de software y DevOps
- Estudios que proporcionen algún tipo de solución, guía o marco de trabajo relacionado con la enseñana de prácticas ágiles de desarrollo de software en las universidades
- Estudios que reporten sobre el éxito, fracaso y retos de la experiencia de la enseñanza de prácticas ágiles de desarrollo de software
- Estudios que proporcionen evidencia sobre la enseñanza de prácticas ágiles de desarrollo de software y el impacto en la industria

[1] http://scholar.google.com
[2] https://www.safaribooksonline.com

El criterio de exclusión de artículos fue el siguiente:

- Estudios relacionados en dar a conocer aspectos sobre procesos operativos y de negocios asociados con metodologías de desarrollo ágil.
- Estudios relacionados en la enseñanza de metodologías ágiles pero que no cubren o cubren muy poco aspectos sobre prácticas ágiles de desarrollo de software.
- Estudios relacionados con casos de estudio de la aplicación y/o enseñanza de prácticas ágiles de desarrollo de software fuera de la universidad.
- Artículos publicados hace más de 10 años atrás (rango aceptable 2007-2017).

2.1.5. Resultado de la revisión

Luego de obtener los artículos, literatura y recursos relevantes, se identificó que los mismos podían clasificarse en tres grupos: desarrollo ágil de software, enseñanza de prácticas ágiles de desarrollo de software en universidades extranjeras y enseñanza de prácticas ágiles de desarrollo de software en Costa Rica.

2.2. Desarrollo Ágil de Software

Las metodologías de desarrollo ágil de software emergen al final de de la década de los noventa. El término "ágil" se utiliza para agrupar una serie de métodos tales como *Scrum*, *eXtreme Programming*, Crystal, *Feature Driven Development* (FDD), *Dynamic Software Development Method* (DSDM) y *Adaptative Software Development* [29]. Las metodologías ágiles se caracterizan por promover el desarrollo interativo e incremental y la entrega frecuente de funcionalidad prioritaria para los clientes. Estas metodologías estan dirigidas a equipos pequeños y altamente colaborativos. *Scrum* y XP son las metodologías que presentan mayores niveles de adopción [3]: XP se centra en prácticas de desarrollo mientras que *Scrum* cubre principalmente la administración del proyecto.

Las metodologías ágiles son adecuadas para proyectos con requerimientos altamente cambiantes, se motiva aceptar y responder ante los cambios. En 2001,

los desarrolladores de varios de estas metodologías ágiles escribieron el Manifiesto Ágil [2] (Apéndice 2.2.1). Los principios detrás del Manifiesto Ágil incluyen entrega rápida, frecuente consistente y continua de software funcional, respuesta a requerimientos cambiantes, comunicación efectiva y equipos motivados con capacidad de auto organizarse.

2.2.1. El Manifiesto Ágil

Individuos e interacciones sobre procesos y herramientas Software funcionando sobre documentación extensiva Colaboración con el cliente sobre negociación contractual Respuesta ante el cambio sobre seguir un plan

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.

2.2.2. Prácticas Ágiles de desarrollo de software

De acuerdo con [30], a pesar de la gran cantidad de literatura disponible sobre metodologías ágiles de software, mucha de la misma menciona poco acerca de las prácticas agiles de desarrollo de software. La mayoría se centran principalmente en gestión, procesos y estimación, y no tanto en la parte relacionada con la ingeniería de software.

En contraste con las metodologías, las prácticas ágiles están un nivel por debajo debido a estas ya que son una parte muy específica de una metodología, la cual aborda varios aspectos. Algunos ejemplos conocidos son programación en parejas, desarrollo orientado en pruebas y revisiones de código. A pesar de no haber una definición común en la literatura de práctica ágil [31], se puede tomar XP en consideración para tener un mejor entendimiento por medio del estudio de la colección de prácticas de desarrollo que se promueven en esta metodología. A diferencia de otras metodologías como *Scrum*, XP se dedica tanto a la gestión del proyecto como también en la forma en cómo los equipos construyen código [32].

Para efectos de esta revisión, se definen un conjunto de prácticas. Esto porque las metodologías ágiles llaman a sus prácticas de forma diferente. Para obtener un conjunto de prácticas de referencia, se tomó como punto de partida el reporte anual del estado de la metodologías ágiles de Version One [3], y el cual expone las prácticas ágiles en desarrollo de mayor aceptación. Las prácticas ex-

puestas en este reporte se confrontaron con la literatura acerca de XP [30,32,33] con el fin de validar si se hace referencia de las mismas. El resultado es la siguiente lista de prácticas ágiles en desarrollo de software:

- Pruebas
  - Pruebas unitarias
  - Desarrollo orientado a pruebas (TDD, por sus siglas en inglés)
  - Desarrollo orientado al comportamiento (BDD, por sus siglas en inglés)
  - Desarrollo orientado a pruebas de aceptación (ATDD, por sus siglas en inglés)
- Integración continua
- Estándares de codificación
- Refactorización de código
- Puesta en producción continua
- Propiedad compartida del código
- Diseño emergente

Pruebas unitarias las pruebas unitarias se concentran en una clase o método. Se ejecutan enteramente en memoria lo que las hace muy rápidas [32].

Desarrollo orientado a pruebas (TDD) es un ciclo rápido de prueba, codificación y refactorización [32]. La idea es qeu cuando se desarrolla una nueva pieza de funcionalidad o se arregla un error, los desarrolladores primero crean una prueba que actúa como una especificación ejecutable del comportamiento esperado del código que se va a escribir. Estas pruebas no solamente orientan el diseño de la aplicación sino que también sirven tanto como prueba de regresión y como documentación del código y del comportamiento esperado de la aplicación. [34]

Desarrollo orientado al comportamiento (BDD) toma la posición de que puede convertir una idea de un requisito en código implementado, probado y listo para la producción de manera simple y eficaz, siempre y cuando el requisito sea lo suficientemente específico para que todos los involucrados sepan lo que está pasando. BDD utiliza una historia como la unidad básica de funcionalidad, y por lo tanto de entrega. Los criterios de aceptación son una parte intrínseca de la historia. [35].

Desarrollo orientado a pruebas de aceptación (ATDD) es una forma avanzada de TDD en el cual los criterios de aceptación automatizados, definidos en colaboración con los usuarios, orientan y enfocan el proceso de desarrollo. Esto ayuda a asegurar que todo el mundo entienda qué características están en desarrollo.

Integración continua la idea que hay detrás es que si la integración regular del códiga es algo bueno integrar, ¿por qué no hacerlo todo el tiempo? En el contexto de la integración "todo el tiempo" significa cada vez que alguien promueva cualquier cambio al sistema de control de versiones [34].

Estándares de codificación *eXtreme Programming* recomienda la creacio4n de estándares y guías de codificación con las que los desarrolladores se puedan poner de acuerdo en adherise cuando programan [32]. Esto incluye: hábitos de programación, ambientes de desarrollo integrados, herramientos, configuración de archivos, convenciones de construcción y diseño y manejo de errores.

Puesta en producción continua técnica que alienta a poner en producción cada cambio que haya pasado a través proceso de pruebas automatizadas. Aunque también se puede hacer entregar de forma continua software funcional en ambientes de pruebas o pre-producción, el punto crucial con esta técnica que entregar software funcional directamente en ambientes de producción [34].

Propiedad compartida del código extiende la responsabilidad de mantenimiento del código a todos los programadores. Todos comparten la responsabilidad de la calidad del código. Ninguna persona reclama la propiedad de alguna parte del sistema y cualquiera puede hacer un cambio en cualquier lugar del código [32].

Diseño emergente *eXtreme programming* hace demandas desafiantes a los programadores: cada semana deben finalizar de 4 a 10 historias de valor para los clientes. Cada semana, los clientes pueden revisar el plan actual e introducir nuevas historias. En otras palabras, como programador se tiene que estar en la capacidad de producir valor al cliente de forma constante. Afortunadamente, *eXtreme programming* provee una solución a esto: diseño emergente e incremental, que permite a construir nueva infraestructura técnica incrementalmente, en pequeñas piezas, de acuerdo a como se entregan las historias [32].

El código como infraestructura es la práctica en donde técnicas, procesos y herramientas utilizadas en el desarrollo de software son utilizadas para administrar la entrega y configuración de aplicaciones. Una organización que carece de código como infraestructura puede sufrir de retrasos en la entrega, por lo general debido a que los cambios en la infraestructura crean diferencias entre los ambientes de desarrollo y lo que administran el personal de operaciones [34].

Administración de la configuración se refiere al proceso mediante el cual todos los artefactos relevantes a un projecto y las relaciones entre ellos son almacenados, obtenidos, identificados de forma única y modificados. La estrategia de administracio4n de la configuración determinará cómo se gestionan todos los cambios que pasan dentro de un proyecto. Por lo tanto, se registra la evolución de los sistemas y aplicaciones [34].

Pruebas automatizadas El reto detrás de las pruebas de software es que una tarea que es sumamente importane pero que a la vez puede llegar a consumir mucho tiempo. La respuesta a esto es aplicar un conjunto de pruebas automatizadas tales como pruebas unitarias, de integración, de carga y de aceptación las cuales se ejecutan automáticamente luego de que se efectúa algún cambio en la aplicación. Una prueba automatizada añade valor al encontrar defectos antes que el código sea utilizado por los usuarios finales [34].

Monitoreo del rendimiento lo que se pretende es brindar visibilidad del estado actual de la aplicación. Esto se puede poner a disposición por medio de interfaces gráficas las cuales muestran métricas, alertas y bitácoras. Al analizar estos resultados se puede ir hasta una línea de código y determinar si existe un problema en la misma o en la infraestructura [34].

2.2.3. DevOps

DevOps es la práctica que combina desarrollo y operaciones y que desde el 2009 viene siendo adoptada por muchas organizaciones de la industria [36]. Se puede resumir como una práctica que anima a establecer un ambiente en donde la construcción, pruebas y la entrega de software puede realizarse de forma rápida, frecuente y más confiable [4]. DevOps apareció como una respuesta directa a los retos de las plataformas de software a gran escala que se actualizan rápidamente, tal y como lo indica Anderson [37]: *"El movimiento DevOps emergió a partir de uno de los clásicos obstáculos en muchas organizaciones. Los desarrolladores construyen código y aplicaciones, y las envían al personal de operaciones solo para descubrir que el código y las aplicaciones no corren en producción. Es el clásico problema de "se ejecutaba en mi máquina y funcionaba, ahora es problema del personal de operaciones" "*.

2.2.4. DevOps vs Metodologías Ágiles

A pesar de no estar considerada como una metología de desarrollo ágil como tal, autores como Bæebark [4] ven al DevOps como el siguiente paso natural del movimento de desarrollo ágil el cual hace énfasis en software funcional, colaboración, velocidad y respuesta al cambio. Sin embargo, DevOps tiene mayor énfasis en las operaciones e introduce elementos nuevos: el código nuevo, arreglo de errores e incrementos son desarrollados, probados, integrados y entregados a los usuarios finales en cuestión de horas, y el equipo de desarrollo es responsable total de los requerimientos, desarrollo, pruebas, entrega y monitoreo. En [38] se estudió la relación de las prácticas de DevOps con otras metodologías de desarrollo y se destaca con respeto a DevOps y a las metodologías ágiles que:

- DevOps extiende las metodologías ágiles: los principios de DevOps puede proporcionar una extensión prágmatica a las prácticas ágiles. Se pueden lograr los objetivos de las metodologías ágiles al extender los principios de estas metodologías a través de una tubería de entrega de software.
- Las metodologías ágiles son facilitadores para DevOps: en [38] se menciona que las metodologías ágiles pueden ser consideradas como facilitadores para adoptar los principios detrás de DevOps.
- Las metodologías ágiles apoyan DevOps: al motivar la colaboración entre los miembros de los equipos, la automatización de la construcción, entre-

ga y pruebas, el establecimiento de métricas y el intercambio de conocimientos y herramientas.

De igual forma, para validar esta relación e interoperatibilidad entre ambos enfoques, la búsqueda de literatura realizada en 2.1.3 en Safari *Books Online* retornó 329 títulos los cuales incluían libros y videos. Esto sugiere que hay mucha aceptación en el uso de ambos enfoques en conjunto.

En [38] también se expone que a pesar de las similitudes, DevOps no cumple con todos los principios propuestos en el Manifiesto Ágil [2] y que DevOps elimina la brecha entre desarrolladores y personal de operaciones mientras que las metodologías ágiles están orientadas a alinear requirimientos de negocio con el desarrollo.

De acuerdo con [38], las principales prácticas en desarrollo de software de DevOps son:

- El código como infraestructura
- Administración de la configuración
- Integración continua
- Entrega continua
- Pruebas automatizadas
- Monitoreo de rendimiento

La guía en [39] es un ejemplo de la aplicación de las prácticas de DevOps identificadas en [38] adaptadas para desarrollo de aplicaciones de tipo software como servicio o SAAS[3] por sus siglas en inglés.

Al igual que en la sección 2.2.2, en el apéndice **??** se proveen definiciones de estas prácticas.

2.3. Relevancia de Prácticas ágiles de desarrollo

Los reportes consultados sobre el estado actual de metodologías ágiles [3] y DevOps [24] reflejan una amplia aceptación de ambos enfoques en la industria de tecnología. Por ejemplo, en [3] el 94 % de las organizaciones que tomaron

[3] *Software-as-a-service*

parte del estudio respondieron afirmativamente a la pregunta de si practicaban enfoques ágiles y el 71 % de las mismas reportó que estan llevando a cabo iniciativas en DevOps. En [24] se señala que para lograr que las personas logren el máximo provecho de DevOps, las prácticas técnicas tales como automatización de pruebas, puesta en producción automática, integración continua y un manejo de versiones detallado son claves. Los que práctican DevOps tienen que estar concientes del uso de estas prácticas técnicas y no solamente de herramientas específicas.

La encuesta realizada en [25] donde 160 compañías suizas y casi 200 profesionales en tecnología de información participaron, también muestra resultados claros. Las compañías y los profesionales que siguen metodologías ágiles están más satisfechos con los enfoques que estas promueven que con metodologías basadas en planeamiento. Otro resultado importante del estudio es que los principales objetivos de introducir desarrollo ágil son: la gestión de prioridades cambiantes y la mejora del proceso de desarrollo en general. La encuesta muestra que no hay tantos ingenieros con habilidades para desarrollo ágil y esto le sugiere a los autores que los estudiantes no están siendo educados con las habilidades necesarias. El 70 % de los participantes piensan que los bachilleres en computación tiene muy poco conocimiento de metodologías ágiles; la mayoría piensa lo mismo para los graduados de maestría. Cuando se consultó sobre si las metodologías ágiles deberían ser parte integral del plan de estudios de carreras de computación, la gran mayoría recomendó que el desarrollo de software ágil debería de ser parte integral de estos planes de estudio.

Un estudio similar se llevó a cabo en el estado de Minnesota, Estados Unidos, con el fin de evaluar una nuevas opciones en los planes de estudio de carreras de tecnología de información de las universidades de ese estado [14]. El estudio señala que las habilidades en desarrollo ágil son cada vez más importantes en las decisiones de contratación, el 62 % lo considera un factor relevante y el 41 % comenzaron a hacerlo en los últimos 3 años. El 59 % dice que su desarrollo y entrega de software están basadas en prácticas ágiles/DevOps/*Cloud Computing*. Hay preocupación sobre la disposición de la mano de obra de Minnesota para promover una *transformación digital* y de la capacidad del sistema educacional del estado en producir esta fuerza de trabajo: solamente el 32 % está de acuerdo que la fuerza de trabajo está bien preparada en términos de competencias y el 49 % no están de acuerdo en que el sistema educativo está produciendo una fuerza laboral digital suficientemente grande.

Los resultados de la revisión sistemática de la literatura realizada por Rade-

marcher y Walia [26] para determinar qué áreas de los estudiantes graduados con más frecuencia no alcanzan las expectativas de la industria o la academia indican que los estudiantes recién graduados presentan carencias en áreas técnicas, personales y profesionales. Dentro de las carencias en las áreas técnicas, el estudio de [26] señala las siguientes:

- Carencias en herramientas de software: una categoría que reporta muchas deficiencias y que muestra varios ejemplos específicos. El ejemplo más comunmente reportado es la gestión de la administración de la configuración. Los estudios consultados en este artículo encontraron que los desarrolladores de software recién contratados carecían de experiencia en herramientas de gestión de la configuración. Así mismo se mencion que existe una brecha grande entre las habilidades de los profesionales de software con respecto a su educación.

- Pruebas: es una área que presenta deficiencias tanto en la industria como en la academia. Estudiantes avanzandos mostraron poca habilidad de uso de herramientas para cobertura de de pruebas y en escribir pruebas relevantes. Los desarrolladores de software recién contratados muestran carencias en la habilidad de llevar a cabo pruebas de código exhaustivas. También se reportaron problemas con respeto al *debugging* de código.

En Costa Rica

El sector de tecnología de la información en Costa Rica ha experimentado el mismo crecimiento explosivo con respecto a la oferta y demanda de personal calificado que se ha llegado a reportar en otros países. De 1997 al 2000 las compañías de desarrollo de software y otros emprendimientos crecieron a una tasa del 40 % al 60 % en la planilla [15].

De acuerdo con [16] con respecto a la fuerza laboral total del país, el empleo en el sector de tecnologías de la información representó en 2006 un $2,32\,\%$ y en 2007, $2,47\,\%$, mostrando un leve aumento de $0{,}15\,\%$ para el último año. El aumento de 4886 trabajadores en el sector de tecnologías de la información representa un crecimiento del $10{,}7\,\%$ en relación con la fuerta laboral todal del sector. Por otro lado, el comportamiento del empleo en el área de producción de bienes y servicios de las tecnologías de la información, entre 2006 y 2007 respecto de la fuerza laboral, fue inverso: el porcentaje de trabajadores dedicados a la producción disminuyó un $19{,}8\,\%$, mientras que el porcentaje de empleados

dedicados a servicios aumentó un $19{,}6\,\%$.

En [27] se hizo un mapeo de las habilidades consideradas las más importantes a encontrar en profesionales en tecnologías de la información. Una de las motivaciones de este mapeo es dar a conocer las áreas en las que las empresas multinacionales están más interesadas y de esta forma motivar la creación de nuevos programas educativos para estimular esas habilidades. Las categorías que reportaron mayor interés en las multinacionales fueron las relacionadas con "Desarrollo e Implementación" y "Entrega y Operación". Esto refleja que las hay un interés marcado en contar con profesionales que puedan tener habilidades que cubran todas las necesidades que se presenten durante el ciclo de desarrollo de sistemas.

El estudio llevado a cabo por el Programa Sociendad de la Información y el Conocimiento [16](Prosic) de la Universidad de Costa Rica señala que hay una preocupación en el país por la escasez de mano de obra calificada para el sector de las tecnologías de la información, particularmente en la industria del software. Se añade que *"en el país no se han tomado las medidas necesarias para que este sector se consolide dentro de una economía del conocimiento. Esta consolidación requiere, entre otras cosas, mayores inversiones en investigación y desarrollo y adecuar los sistemas educativos existentes en el país para que preparen profesionales en adecuada cantidad, así como con buena calidad para satisfacer los requerimientos del mercado laboral."*

Dentro del mismo estudio, se aplicó una encuesta a 160 organizaciones de tecnologías de la información y se identificó que el principal problema de para el desarrollo del sector es la poca disponibilidad de recurso humano calificado. En este apartado la palabra "calificado" es motivo de diferenciación porque aunque se cuenta con fuerza laboral, en muchos casos se ha logrado experimentar insuficiencia en la búsqueda de los conocimientos que se desea que posean. Se acusa de un rezago educativo y recomiendan:

- Fortalecimiento de la educación técnica
- Ajuste de los programas: actualización de formación y curricula
- Actualización de docentes
- Diseño de carreras efectivamente articuladas
- Integración de oferta y demanda

- Integración entre empresas y sector educativo para la definición de perfiles de formación

2.4. Enseñanza de Prácticas Ágiles de Desarrollo de Software

De acuerdo con los resultados obtenidos a partir la búsqueda en la sección 2.1.5, se evidencia que la enseñanza de prácticas ágiles de desarrollo es un tema activo de investigación y desarrollo en universidades extranjeras. En [25] se señala que a pesar que el desarrollo de software ágil ha existido por más de una década, incluso antes del Manifiesto Ágil, la enseñanza de desarrollo de software ágil sólo ha llamado la atención en conferencias educativas y de investigación en años recientes. Una razón para esto puede ser que el desarrollo ágil no tiene bases teóricas sino que más bien ha sido desarrollada a partir de la práctica. En [5] se discute las razones por las cuáles los programas de ingeniería de software deberían de enseñar desarrollo ágil de software. Enfatizan que los ingenieros de software no solamente necesitan ejercitar habilidades técnicas sino también sociales y éticas, que son la aspectos básicos del desarrollo ágil.

Aunque las formas y aplicación de la enseñanza de estos temas varían de institución en institución, en la literatura prevalece la opinión de que la mejor forma de aprender desarrollo ágil es mediante el enfoque de aprender-haciendo (*learn by doing*). Se encontraron ejemplos en donde se introduce al aprendizaje en prácticas de desarrollo ágil por medio de la implementación de algún trabajo final de graduación[4] [8], proyectos/casos de estudio que se desarrollan durante un semestre [13], desarrollando juegos [11], por medio de laboratorios de robótica [12], Wikis [7] o bien como un tema paralelo al principal de un curso. Por ejemplo en [9], se introduce a los estudiantes al versionamiento de código como herramienta de apoyo al desarrollo de una aplicación Web. De acuerdo al estudio en [10] el dominio de habilidades técnicas y prácticas de ingeniería representan la base para el entendimiento de la aplicación de otros temas sobre metodologías ágiles y para ir desarrollando una mentalidad dirigida al desarrollo de software de calidad.

Con respecto a la enseñanza de DevOps en universidades, se siguen un enfoques similares a los anteriormente mencionados para las prácticas de desa-

[4] *Capstone project*

rrollo ágil pero con un mayor énfasis en la introducción en temas de automatización [4, 36, 40] de procesos de construcción, pruebas, entrega y computación en la nube. Un caso interesante es el que se expone en [6], "un campo de entrenamiento para emprendedores", el cual es básicamente un curso de verano dedicado a ejercitar al máximo las habilidades de programación de los estudiantes y por medio de esto ejercitar también prácticas ágiles de desarrollo y de DevOps.

2.4.1. Prácticas Ágiles de Desarrollo en Planes de Estudio de Referencia

Desde la década de los sesenta, la *American Computing Machinery* (ACM) junto con sociedades profesionales y científicas en computación se han dado a la tarea de propoponer y adaptar recomendaciones en los planes de estudio debido al panorama cambiante de la tecnología [21].

En la guía curricular recomendada para los programas de bachillerato para la carrera de ciencias de la computación en [41] se incluyen temas tales como integración contínua, automatización, control de versiones, estándares de codificación, desarrollo orientado a pruebas, pruebas unitarias y pruebas de integración como parte del cuerpo de conocimiento de ingeniería de software sugerido para esta carrera. Para la guía curricular recomendada para los programas de ingeniería de software [42] se incluyen temas de prácticas ágiles de desarrollo de software en las secciones de Validación y Verificacion, y en la sección de Procesos asociados al desarrollo de software. También se pudo constatar la prescencia de prácticas ágiles de desarrollo en cursos de referencia sobre programación, ingeniería de requerimientos, proyecto finales, entre otros, que han desarrollado universidades en el extranjero. En [43] las prácticas ágiles de desarrollo aparecen como parte de las competencias del área de desarrollo y entrega de sistemas.

Los autores de [40] junto con un grupo de trabajo formado por investigadores de universidades del estado de Minnesota en Estados Unidos, proponen un plan de estudios para carreras de sistemas de información y de tecnologías de la información que responde a las tendencias de desarrollo ágil y DevOps [14] de la actualidad. La propuesta que presentan fue la que se pudo identificar como la más radical hacia en la enseñanza y aprendizaje de prácticas ágiles de desarrollo.

2.5. Enseñanza de Prácticas Ágiles en Costa Rica

La evidencia recolectada apunta a que los primeras iniciativas dirigidas a la introducción y enseñanza de prácticas ágiles en las universidades de Costa Rica se dio a inicios del 2000. El siguiente fragmento tomado de [15] reseña muy bien la situación de ese momento y da a conocer el porqué las universidades en Costa Rica empezaron a revisar sus planes de estudio para que fueran más acordes con la tendencias de la industria: *"A fines de la década de 1990, la demanda de profesionales y técnicos en Informática tenía un crecimiento cercano al 90 % interanual Norteamérica, Europa Occidental y Japón. En Costa Rica, la industria costarricense de software crecía, en personal, entre un 40 % y 60 % anualmente, gracias a su éxito en mercados internacionales. La sostenibilidad de esa industria radica no en su costo, sino en la calidad de sus productos, la cual depende fundamentalmente del talento humano y de los sistemas de trabajo (procesos) aplicados. El cambio en la tecnología de software y el aumento en la demanda de aplicaciones informáticas – particularmente para la Web y los dispositivos móviles – han exigido innovaciones en la forma de educar profesionales en desarrollo de software preparados para producir aplicaciones y componentes de software de calidad mundial. Además, desde el año 2003, Costa Rica participa como proveedor de servicios tecnológicos y empresariales habilitados por tecnologías digitales. Esto ha planteado la necesidad de preparar profesionales con perfiles distintos de los que se graduaban de las universidades costarricenses."*

En sus inicios, la hoy Universidad Cenfotec [44] presentó un plan de estudios para el programa de Especialista en Tecnología de Software, el cual se diferenciaba de los carreras universitarias convenciales al sacrificar la generalidad en aras de la especialidad y permitir al graduado incorporarse al mercado laboral para realizar desarrollo de software de alta calidad. La intención detrás de este programa era ejercitar de forma intensiva planificación, diseño y programación de software a partir de la ejecución de proyectos y del enfoque de aprender-haciendo.

La Universidad de Costa Rica ha reportado la introducción de *Scrum* y programación extrema como parte de sus cursos en ingeniería de software. En [18] se reporta el uso de estas prácticas junto con RUP. El profesor a cargo establece un proyecto que se va desarrollando durante todo el semestre. La Universidad Nacional reporta la ejecución de iniciativas para impartir cursos de programación bajo modelos pedagógicos orientados a proyectos y a resolución de problemas [19, 20], sin embargo en este caso, la experiencia reportada no in-

cluye prácticas ágiles de desarrollo sino más bien lo que se intenta es propiciar el aprendizaje colaborativo, la participación y la autonomía. Cenfotec también promueve este enfoque de aprendizaje colaborativo y orientado a resolución de problemas [15,17] por medio proyectos integradores los cuales aparte de formar habilidates técnicas o fuertes, también pretenden formar en habilidades suaves como comunicación, creatividad, valores éticos, agilidad, educación continua, entre otros.

2.5.1. Sobre la experiencia en la enseñanza de prácticas ágiles de desarrollo

2.5.2. Beneficios

- Uso de herramientas de software de actualidad y un aprendizaje más alineado con las tendencias de la industria [15].
- Representa para muchos estudiantes la primera experiencia en donde se codifica, prueba, integra y se pone en producción un sistema, recibiendo en cada paso retroalimentación inmediata [10].
- En [25] se reporta una mejora de las habilidades de programación y calidad del código.
- De acuerdo a [11] los estudiantes sentían que se rompía la rutina y que se daba mayor paso a la espontaneidad y la innovación. Esto también propiciaba que los estudiantes tuvieran mayor compromiso en los proyectos en los que trabajaron.
- La actividad en cuanto a los emprendimientos llevados a cabo por los estudiantes aumenta considerablemente luego de ser introducidos a estos temas [6].
- Repetir la misma metodología y herramientas en cada *sprint* mejoró la productividad de desarrollo y esto revela una mejoría de la curva de aprendizaje. Conforme fueron avanzando el desarrollo de los *sprints*, las horas requeridas de esfuerzo para desarrollar una funcionalidad fue disminuyendo, por lo tanto la productividad mejoró [18].
- La introducción de prácticas ágiles de desarrollo fortaleció el proceso de pruebas, asegurando mayor calidad y evitando el re-trabajo [18]

- Las prácticas ágiles de desarrollo de software, al ser el aspecto más técnico en las metodologías ágiles resulta ser también el más atractivo para los estudiantes y en donde concentran más sus esfuerzos de aprendizaje [13]

- La mayoría de la evidencia señala que la aplicación de la enseñanza de prácticas ágiles se hace a través de enfoques orientados a resolución de problemas y proyectos. Los resultados obtenidos en [19] resaltan la importancia de estos enfoques porque ofrecen a los estudiantes mayores posibilidades de tomar decisiones cuando se trabaja con la solución de problemas, y en la forma de abordarlo y, como consecuencia, aumentar su motivación en el proceso de aprendizaje y de asumir mayor responsabilidad.

- Hazzan y Dubinsky [5] señalan otros beneficios asociados a la enseñanza de prácticas ágiles de desarrollo tales como:

  - Su aplicabilidad en la industria
  - Trabajo en equipo
  - Apoya otros procesos de aprendizaje
  - Lidia con aspectos humanos, promueve la diversidad, habilidades de gestión y normas éticas
  - Promueve hábitos mentales
  - Brinda una imagen detallada de ingeniería de software
  - Las metodologías ágiles proporcionan un marco de trabajo que se puede enseñar

2.5.3. Retos

- Las instituciones educativas son lentas en crear e innovar en los planes de estudio para preparar a la futura fuerza laborar y desarrollar programas para re-entrenar aquellos que se ya se encuentran trabajando [15].

- Existen muy pocos artículos publicados acerca de la enseñanza de DevOps [4].

- En [13] se señala que:

- Cuando se enseñan procesos de desarrollo ágil debe de haber retroalimentación rápida y directa por parte de los profesores. La retroalimentación permite la corrección temprana a prácticas que se aplican mal.
- La planificación de cursos en estos temas requiere de un gran esfuerzo de adquisición de conocimiento previo, maxime si los potenciales instructores han tenido poca exposición a los mismos.
- La enseñanza de metodologías ágiles necesita tener un elemento práctico, se necesita aplicar para que se pueda saber. Cursos en donde se exponen solamente aspectos teóricos resultan menos atractivos para los estudiantes.

- Si la formación en tecnologías ágiles y DevOps están por fuera de los planes regulares y se brinda en la modalidad de curso de verano o electivo, esto puede llegar a necesitar de mucho esfuerzo por parte de unidad académica a cargo y reportar pocas ganancias [6].

- No es conveniente que se aprendan y apliquen metodologías ágiles como único método de enseñanza en cursos de ingeniería de software, porque generalmente estos modelos de desarrollo promueven un proceso informal, con escasa documentación y con un mínimo de productos de trabajo de ingeniería de software que atentan con la calidad del software y otras prácticas recomendadas [18]

- En [19] se recalca la importancia de contar con personal motivado y comprometido en adoptar nuevos cambios en los cursos de la carrera: *"Además, aunque la mayoría de los docentes manifiestan estar abiertos a hacer cambios que mejoren el aprendizaje y motiven más a los estudiantes, es evidente que estos procesos implican más trabajo y un cambio de cultura de los actores involucrados. Además es claro que se requiere de una línea de gestión clara y de una estructura de gestión curricular flexible que permita modificar las estructuras de evaluación tradicionales"*.

- Kropp y Meier [25] opinan que los cursos sobre desarrollo ágil de software no pueden ser enseñados en cursos aislados de ingeniería de software. Un reto es la integración de prácticas de desarrollo ágil en otros cursos como programación o análisis y diseño orietado a objetos, algoritmos y estruturas de datos.

Capítulo 3

Marco Metodológico

3.1. Tipo de Investigación

Debido a que en esta propuesta de investigación se contempla la elaboración de encuestas y entrevistas para la recolección y análisis de datos, se va a hacer uso de un enfoque cualitativo.

3.2. Fuentes y Sujetos de Investigación

3.2.1. Fuentes

Los planes de estudio de las universidades.

De forma más específica se pretende evaluar los programas de los cursos que involucren desarrollo y/o análisis de software de alguna forma. Preliminarmente, cursos tales como:

- Programación (cursos introductorios y avanzados)
- Desarrollo Web
- Aseguramiento de la Calidad
- Análisis/Diseño de Sistemas
- Ingeniería de Software
- Ingeniería de Requerimientos
- Bases de Datos

se muestran como candidatos iniciales para el estudio. Se espera que durante la ejecución de la investigación los cursos a evaluar puedan variar de una universidad a otra, pero esto es esperado debido a que cada institución maneja énfasis de enseñana diferentes. El trabajo conjunto con las autoridades de la universidad hará que se logre identificar de forma más precisa cursos a evaluar dependiendo de la estructura del plan de estudios.

Este estudio no tomará en consideración cursos que no tenga relación alguna con software, como lo podrían ser cursos de matemática, física, arquitectura y redes de computadores, idiomas y de estudios generales.

Personal académico y administrativo de las universidades.

Con ellos se pretende trabajar de forma conjunta para valorar la penetración de prácticas ágiles de desarrollo de software en los cursos relacionados con desarrollo y/o análisis de software. Mediante la aplicación de una encuesta y entrevistas presenciales es que se desea estimar la importancia que se le dan a estos temas dentro de sus planes de estudio así como el intercambio experiencias con los mismos, casos de éxito/fracaso y limitaciones que hayan encontrado.

Informes del estado actual de prácticas ágiles de desarrollo de software

El trabajo llevado a cabo en el capítulo 2, así como los informes de referencia de la industria [3,24] brindan un punto de partida para saber qué es lo relevante actualmente y compararlo con lo hallado en los planes de estudio en cuestión.

Población

Los planes de estudio de carreras en computación, informática y sistemas de información listadas en el Cuadro 1.1, así como personal académico y admistrativo a cargo de los mismos.

Muestra

Los planes de estudio de carreras en computación, informática y sistemas de información acreditadas por el SINAES, listadas en el Cuadro 1.2, así como personal académico y admistrativo a cargo de los mismos.

3.2.2. Sujeto de la investigación

Los planes de estudio de las universidades serán el principal sujeto de investigación de esta propuesta, específicamente en cómo incorporan prácticas ágiles de desarrollo en ellos.

3.3. Técnicas de Investigación

Para esta propuesta se propone una investigación de enfoque cualitativo. De acuerdo con Creswell [45] *"la investigación cualitativa es un enfoque para explorar y entender el significado de individuos o grupos envueltos en un problema humano. El proceso de investigación involucra preguntas emergentes y procediientos, datos recolectados típicamente desde los participantes, análisis de datos de forma inductiva de temas particulares a temas generales y en donde el investigados hace interpretaciones del significado de los datos. El reporte final se tiene una estructura flexible. Aquellos que participan de esta forma de investigación apoyan una forma de ver la investigación que honra el estilo inductivo, un enfoque de significado invidual y la importancia de representar la complejidad de una situación"*.

De acuerdo a lo que ha venido exponiendo a lo largo de esta propuesta, este enfoque es el más adecuado debido a la naturaleza exploratoria y de interacción humana. Para la realización de esta investigación se proponen las siguientes técnicas de investigación:

- Encuesta: para evaluar el nivel de adopción e importancia que se le da a las prácticas ágiles de desarrollo de software tanto en los planes de estudio como también desde el punto de vista del personal académico y administrativo de la universidad.
- Entrevista: para discutir tanto los resultados de la encuesta con el personal de la universidad y también como puntos de vista, experiencias y expectativas con respecto al tema.

3.3.1. Encuesta

Debido a lo que se planea en primera instancia es explorar el nivel de penetración de las prácticas, se ha confeccionado una encuesta con 5 preguntas para obtener apreciaciones iniciales del personal a cargo de la universidad.

El detalle de la encuenta se encuentra en el apéndice 1.

3.3.2. Entrevista

Como preparación para la entrevista, se solicitará de antemano una copia del plan de estudios de la universidad y los programas de los cursos que para el investigador tenga relación directa con desarrollo de software. De esta forma se podrán analizar estos documentos y contrastar la información contenida en ellos con los entrevistados.

1. ¿Cuál es el perfil profesional del estudiante graduado de esta carrera?
2. ¿Qué objetivos buscaría la universidad con la enseñanza de estas prácticas?
3. ¿Qué otros efectos se considera que podría causar la adopción activa de estas prácticas?
4. Debido a que mucho del uso que se hace de estas prácticas se da en ambientes en la nube, ¿se expone a los estudiantes a tecnólogías en la nube durante su paso por la carrera?
5. Si se tiene planeado adoptar la enseñanza de estas prácticas, ¿cuáles prácticas se tienen en consideración?
6. ¿Los estudiantes han externalizado su interés en la formación estas prácticas?
7. ¿La universidad tiene algún tipo convenio con empresas con el fin de dar a conocer y/o entrenar a estudiantes en estos temas?

3.4. Procesamiento y Análisis de Datos

Se propone seguir el enfoque de análisis cualitativo de [46] para esta tarea, de acuerdo con este enfoque *"La idea es identificar generalizaciones en los datos términos de patrones, sequencias, relaciones, entre otros. El análisis se lleva a cabo de forma iterativa, esto significa que se para a través del material varias veces y cada vez se identifica o ajusta códigos, categorías y similares. Es importante ser*

sistemático y documentar cada paso de la investigacio4n. Esto hace posible describir una "cadena de evidencia" para el lector in el cual no hay forma de que el lector entienda que las conclusiones son confiables".

3.4.1. Pasos en el análisis

1. Estudiar el material recolectado en detalle, incluso si ya fue estudiado durante las entrevistas.

2. Formular un conjunto de códigos de interés. Es usualmente beneficioso hacer esta formulación como una actividad de grupo. Códigos candidatos:

- PV: punto de vista
- EXP: experiencia en la docencia
- BEN: beneficios y objetivos
- LIM: limitación
- FUT: planes futuros

3. Leer todos los textos y marcar en donde los códigos encajen con los contenidos.
3.1. Transcribir las entrevistas a archivos de texto y marcar secciones de interés
3.2. Asignar códigos a los textos marcados
3.3. Tabular los resultados de la encuesta.
3.4. Aplicar técnicas de estadística descriptiva a los resultados de la encuesta.

4. Utilizar el material codificado para llegar a conclusiones. Comparar el texto para diferentes códigos. Por ejemplo, si hay un tipo de tipo de enunciado que es común para todos los códigos de un tipo pero no para otro, incluso si podría esperarse también para el otro tipo, entonces eso se puede investigar más de cerca.

5. Búsqueda de patrones. Con este enfoque un patrón empírico es observado y comparado con un patrón predicado. Cuando hay coincidencia, esto apoya a que el predicado sea verdadero. La búsqueda de patrones se puede basar en explicaciones rivales. Si tiene varios casos con el mismo resultado, es posible

formular un conjunto de explicaciones rivales para este resultado. Entonces es posible investigar los casos y determinar cuál de las explicaciones que describen los casos de la mejor manera.

5.1. Construcción de una explicación. Los patrones son identificados basados en una relación causa-efecto y se buscan explicaciones subyacentes.

5.2. Estudio de casos similares. Este tipo de análisis puede ser conducido si se tienen otros casos de referencia. En este caso se puede utilizar el material recolectado en la Sección 2 para realizar comparaciones.

7. Validación de pares

6. Generar un reporte el que se de a conocer lo encontrado.

Capítulo 4

Resultados Esperados

4.1. Situación Actual y Esperada

El desarrollo de software siguiendo metodologías ágiles goza de mucha aceptación dentro de la comunidad de profesionales. Sus orígenes datan de inicios de la década de los 2000 por lo que no se puede argumentar que es una nuevo paradigma, pero como se ha destacado en esta propuesta la adopción de estos principios ha calado de forma lenta dentro de los planes de estudio de universidades internacionales y de Costa Rica.

Estudios de este tipo se han hecho principalmente en Estados Unidos y en Europa, los cuales han motivado que las universidades hayan empezado a ofrecer a los estudiantes alternativas para enseñar sobre desarrollo de software ágil y prácticas ágiles de desarrollo de software.

En Costa Rica, de acuerdo con los artículos y reportes recopilados se prevee que el grado de adopción de estas prácticas sea de medio a bajo. Informes de la industria como [16, 27] señalan un descontento por parte de los empleadores los cuales argumentan que el personal que contratan tienen carencias para incorporarse de forma efectiva al trabajo y les toma tiempo poder conceptualizar nuevas habiliidades. También, en una observación inicial que se hizo a cursos relacionados con desarrollo de software en los planes de estudio de carreras de computación, informática y sistemas de información de 3 universidades se pudo notar que no se dedica mucho tiempo al desarrollo de estas habilidades paralelas a la programación.

Dentro del análisis inicial, la Universidad Cenfotec se destaca en exponer a sus estudiantes a muchas horas en el estudio de casos de estudio, problemas y al desarrollo de habilidades relacionadas en desarrollo de software. Este se con-

sidera un elemento diferenciador con respecto a otras carreras de Ingeniería de Software que se ofrecen en el país las cuales siguen esquemas tradicionales. Dejando por fuera al Cenfotec, no se pudo encontrar una universidad que brindara un curso el cual llevara a los estudiantes desde la creación y gestión del proyecto de software hasta su puesta en producción, pasando en el camino por temas de buenas prácticas, pruebas e integración

Aunque se espera que la penetración de las prácticas ágiles de desarrollo de software no sea alta, debido a los énfasis y enfoques propios de cada carrera, también sería de agrado saber que en la universidades se reconoce el valor de su enseñanza, como medio para formar profesionales con mejores armas para enfrentar sus retos profesionales futuros, y que se tiene planes para llevar acciones al respecto.

4.2. Propuesta de la Solución

Basado en los casos de estudio de universidades extranjeras se proponen las siguientes soluciones para brindar enseñanza de prácticas ágiles de desarrollo de software en los planes de estudio de carreras de computación, informática y sistemas de información en Costa Rica:

4.2.1. Creación de un curso de laboratorio/taller/práctica en desarrollo de software

Este curso pretende sintetizar el conocimiento adquirido en desarrollo de software por los estudiantes durante la carrera con el fin de aplicarlo en la implementación de una aplicación. En la implementación se pretende cubrir los principales temas relacionados con prácticas ágiles de desarrollo y además se le puede dar enseñanza de primera mano al estudiante de cómo se configura un proyecto de software desde su inicio y cuáles son las estrategias adecuadas para ponerlo en producción.

Muchos cursos de programación están enfocados a resolver problemas muy específicos y en ocasiones no hay espacio para desarrollar habilidades paralelas en los estudiantes. Este curso serviría como complemento para dar a conocer a los estudiantes otras posibilidades que pueden explotar por medio del software.

Los temas de este curso serían:

1. Construcción y administración de proyectos de software

    - Conformación de equipos de trabajo
    - Selección del tipo de aplicación a desarrollar y sus requerimientos
    - Selección de lenguaje de programación - herramientas de construcción
    - Configuración de repositorio de código

2. Herramientas de desarrollo

    - Buenas prácticas de codificación / refactorización
    - Herramientas propias del lenguaje de programación seleccionado
    - Integración continua

3. Introducción a pruebas

    - Pruebas unitarias / Integración
    - Desarrollo orientado a pruebas (TDD)
    - Pruebas automatizadas

4. Puesta en producción

    - Configuración de infraestructura
    - Introducción a ambientes de producción en la nube
    - Introducción a enfoques de entrega continua

5. Otros temas

    - Escalabilidad
    - Rendimiento
    - Seguridad

Los temas anteriores se irán desarrollando y evaluando por etapas y al final se debería contar con una aplicación funcional que podrá ser accesada por medio de Internet o alguna otra plataforma como lo podría ser una aplicación móvil.

4.2.2. Integración en cursos existentes

En esta modalidad se propone que el personal de la universidad evalue cuáles prácticas de desarrollo se acoplan de mejor manera en los temas de cursos ya existentes. Por ejemplo, se podrían utilizar cursos introductorios a la programación para presentar a los estudiantes a temas de configuración y gestión de proyectos de software, los cuales sin duda alguna les será de utilidad para cursos futuros.

Luego, en cursos de temas avanzados, se puede introducir a temas de buenas prácticas de programación e integración. En cursos sobre aseguramiento de la calidad o ingenieria de requerimientos, los temas de pruebas podrían ser incluidos. Se recomienda que, de adoptarse este esquema, los temas que se incluyan sobre prácticas ágiles de desarrollo de software, no se traten de forma tangencial sino que se pueda ejercitar su enseñanza por medio de actividades que motiven a los estudiantes a desarrollar algún caso de estudio.

4.2.3. Talleres/Charlas impartidos por terceros

Las universidades pueden planear la ejecución de talleres y charlas sobre estos temas con empresas y profesionales. Esta es también una opción interesante porque le permite a los estudiantes ponerse en contacto con las tendencias más actuales de la industria y evaluar posibles temas en los que quisieran investigar o especializarse en un futuro.

Capítulo 5

Plan de Trabajo

5.1. Descripción de Puntos de Control

Para la ejecución de este propuesta se propone dos etapas: evaluación de las carreras acreditadas por SINAES primero y luego las carreras no acreditadas.

5.1.1. Evaluación de las carreras acreditadas por SINAES

Se encontraron 8 carreras acreditadas por SINAES. Al contar con esta acreditación las universidades se ha comprometido con el cumplimiento de los criterios de calidad del SINAES. En la primera etapa del estudio, se evaluarán a estas 8 carreras primero. Con esto lo que se busca es:

- Evaluar el nivel de penetración de las prácticas ágiles de desarrollo en las carreras acreditadas por SINAES

- Obtener experiencia en la ejecución del estudio, lo cual permitiría afinar herramientas de recolección y análisis de datos

- Utilizar los resultados de estas universidades como la primer referencia para comparar con las universidades no acreditadas

Una vez que se haya concluido con la evaluación de estas 8 carreras, el equipo de investigación podrá hacer un alto en el camino y examinar la experiencia y resultados obtenido.

5.1.2. Evaluación de carreras no acreditadas

Se procederá con la evaluación de las carreras no acreditadas y se aprovechará la experiencia y el trabajo realizado del punto anterior. Para evaluar estas carreras no se propone un enfoque en particular, lo que hace que quede a jucio del grupo de investigación.

5.2. Cronograma

5.2.1. Para la evaluación de carreras acreditadas por SINAES

- Recolección de datos (7 días)
- Observaciones preliminares (2 días)
- Preparación de entrevistas y encuestas (2 días)
- Ejecución de entrevista y encuestas (7 días, +/- 2 entrevistas por día)
- Análisis e Interpretación de datos (7 días)
- Total: 25 días

5.2.2. Para la evaluación de carreras no acreditadas por SINAES

- Recolección de datos (15 días)
- Observaciones preliminares (7 días)
- Preparación de entrevistas y encuestas (2 días)
- Ejecución de entrevista y encuestas (30 días)
- Análisis e Interpretación de datos (15 días)
- Total: 69 días

Total General: 94 días.

Inicialmente este cronograma está planeado de forma “pesimista” principalmente porque las tareas asociadas a entrevistas se suponen de forma presencial, lo cual implica traslados constantes. En el caso en que se puedan planear entrevistas de forma remota el calendario podría verse acortado.

Bibliografía

[1] Li Jian y Armin Eberlein. 2009. *An analysis of the history of classical software development and agile development.* Proceedings of the 2009 International Conferencen on Systems, Man, and Cybernetics. San Antonio, TX, USA. Octubre 2009. DOI: https://ezproxy.itcr.ac.cr:2878/10.1109/ICSMC.2009.5346888

[2] Kent Beck, Mike Beelde, Arie van Bennekum, Allistair Cockburn, et al. 2001. *Manifesto for Agile Software Development.* Agile Alliance.

[3] Version One. 2017. *11th Annual State of Agile Report.* Obtenido de http://stateofagile.versionone.com/

[4] Henrik Bærbak Christensen. 2016. *Teaching DevOps and Cloud Computing using a Cognitive Apprenticeship and Story-Telling Approach.* In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '16). ACM, New York, NY, USA, 174-179. DOI: https://ezproxy.itcr.ac.cr:2878/10.1145/2899415.2899426

[5] Orit Hazzan and Yael Dubinsky. 2007. *Why software engineering programs should teach agile software development.* SIGSOFT Softw. Eng. Notes 32, 2 (March 2007), 1-3. DOI: http://ezproxy.itcr.ac.cr:2075/10.1145/1234741.1234758

[6] Timothy J. Hickey and Pito Salas. 2013. *The entrepreneur's bootcamp: a new model for teaching web/mobile development and software entrepreneurship.* In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). ACM, New York, NY, USA, 549-554. DOI:http://ezproxy.itcr.ac.cr:2075/10.1145/2445196.2445361

[7] Marija Cubric. 2008. *Agile learning & teaching with wikis: building a pattern.* In Proceedings of the 4th International Symposium on Wikis (Wi-

kiSym '08). ACM, New York, NY, USA, , Article 28 , 2 pages. DOI: http://ezproxy.itcr.ac.cr:2075/10.1145/1822258.1822296

[8] Dabin Ding, Mahmoud Yousef, and Xiaodong Yue. 2017. *A case study for teaching students agile and scrum in Capstone course*. J. Comput. Sci. Coll. 32, 5 (May 2017), 95-101.

[9] Lassi Haaranen and Teemu Lehtinen. 2015. *Teaching Git on the Side: Version Control System as a Course Platform*. In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '15). ACM, New York, NY, USA, 87-92. DOI: http://ezproxy.itcr.ac.cr:2075/10.1145/2729094.2742608

[10] Martin Kropp, Andreas Meier. 2014. *New sustainable teaching approaches in software engineering education*. IEEE Global Engineering Education Conference (EDUCON). IEEE, 2014, pp. 1019–1022. DOI: https://ezproxy.itcr.ac.cr:2878/10.1109/EDUCON.2014.6826229

[11] Bruce A. Scharlau. 2013. *Games for teaching software development*. In Proceedings of the 18th ACM conference on Innovation and technology in computer science education (ITiCSE '13). ACM, New York, NY, USA, 303-308. DOI: http://ezproxy.itcr.ac.cr:2075/10.1145/2462476.2462494

[12] Andreas Schroeder, Annabelle Klarl, Philip Mayer, Christian Kroiß. 2012. *Teaching agile software development through lab courses*. Global Engineering Education Conference (EDUCON), IEEE. DOI: https://ezproxy.itcr.ac.cr:2878/10.1109/EDUCON.2012.6201194

[13] Jan-Philipp Steghöfer, Eric Knauss, Emil Alégroth, Imed Hammouda, Håkan Burden, and Morgan Ericsson. 2016. *Teaching Agile: addressing the conflict between project delivery and application of Agile methods*. In Proceedings of the 38th International Conference on Software Engineering Companion (ICSE '16). ACM, New York, NY, USA, 303-312. DOI: https://ezproxy.itcr.ac.cr:2878/10.1145/2889160.2889181

[14] Advance IT Minnesota. 2016.*Renewing the IT Curriculum: Responding to Agile, DevOps, and Digital Transformation*. Rep. (November 1, 2016). St. Paul, MN. Obtenido de www.DynamicIT.education

[15] Cenfotec. 2016. *"Currículos basados en proyectos integradores.pdf" y póster "Proyectos integradores en U Cenfotec.pdf"*. SIIC Simposio Internacional sobre Innovaciones Curriculares, UCR, 2016.

[16] Claudio Pinto, Rafael Herrera, Francisco Mata, Rosaura Matarrita y otros. 2009. *Formación de capital humano en el sector de TIC en Costa Rica.* Facultad Latinoamericana de Ciencias Sociales. Programa de Investigación sobre Economía del Conocimiento en América Latina y el Caribe. Obtenido de http://www.prosic.ucr.ac.cr/sites/default/files/documentos/caphum.pdf

[17] Ignacio Trejos, Alvaro Cordero. 2017. *Learn-by-doing-collaboratively across the curriculum: Integrative projects at UCenfotec.* IEEE World Engineering Education Conference – EDUNINE2017. 3.

[18] Gabriela Salazar. 2012 *Desafíos del curso de ingeniería de software.* Revista de Educación en Ingeniería. Enero a Junio de 2012, Vol 7, Num 13. Páginas 32-43.

[19] Sonia Mora, Mayela Coto, Georges Alfaro. 2014. *A proposal for implementing PBL in programming courses.* Computing Conference (CLEI), 2014 XL Latin American. Páginas 1-11. DOI: https://ezproxy.itcr.ac.cr:2878/10.1109/CLEI.2014.6965195

[20] Sonia Mora, Mayela Coto, Georges Alfaro. 2014. *Giving more autonomy to computer engineering students: Are we ready?.* 2013 IEEE Global Engineering Education Conference (EDUCON). Páginas 618-626. DOI: https://ezproxy.itcr.ac.cr:2878/10.1109/EduCon.2013.6530170

[21] ACM. 2017. *Curricula Recommendations.* Obtenido de: http://www.acm.org/education/curricula-recommendations

[22] Russell Shackelford, Andrew McGettrick, Robert Sloan, Heikki Topi, Gordon Davies, Reza Kamali, James Cross, John Impagliazzo, Richard LeBlanc, and Barry Lunt. 2006. *Computing Curricula 2005: The Overview Report.* In Proceedings of the 37th SIGCSE technical symposium on Computer science education (SIGCSE '06). ACM, New York, NY, USA, 456-457. DOI=http://dx.doi.org/10.1145/1121341.1121482

[23] Julia Rozovsky. *The five keys to a successful Google team. re:Work.* 2015. Obtenido de https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/

[24] Puppet Labs, DevOps Research & Assessment(DORA). 2017. *State of DevOps.* Reporte. Obtenido de https://puppet.com/resources/whitepaper/state-of-devops-report

[25] Martin Kropp, Andreas Meier. 2013. *Teaching Agile Software Development at University Level: Values, Management, and Craftsmanship*. En Software Engineering Education and Training (CSEE&T). Páginas 179 - 188. DOI: https://ezproxy.itcr.ac.cr:2878/10.1109/CSEET.2013.6595249

[26] Alex Radermacher and Gursimran Walia. 2013. *Gaps between industry expectations and the abilities of graduates*. In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). ACM, New York, NY, USA, 525-530. DOI: http://ezproxy.itcr.ac.cr:2075/10.1145/2445196.2445351

[27] Jorge Murillo, Ignacio Trejos. 2017. *IT Skills Mapping Engine*. In World Engineering Education Conference (EDUNINE), IEEE. Vol 2, March 29-22, 2017. Páginas 49-50

[28] Sistema Nacional de Acreditación de la Educación Superior. *Listado de Carreras Acreditadas*. Obtenido de http://www.sinaes.ac.cr/index.php?option=com_content&view=article&id=13&Itemid=115

[29] Rashina Hoda, Philippe Kruchten, James Noble y Stuart Marshall. 2010. *Agility in context*. SIGPLAN Not. 45, 10 (Octubre 2010), 74-88. DOI: https://ezproxy.itcr.ac.cr:2878/10.1145/1932682.1869467

[30] Neil Ford. 2011 *Agile Engineering Practices*. Video. O'Reilly Media Inc. Obtenido de https://www.safaribooksonline.com/library/view/neal-ford-on/9781449314439/. ISBN: 9781449314439

[31] Philipp Diebold y Marc Dahlem. 2014. *Agile practices in practice: a mapping study*. En Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14). ACM, New York, NY, USA, , Artículo 30 , 10 páginas. DOI: http://ezproxy.itcr.ac.cr:2075/10.1145/2601248.2601254

[32] James Shore y Shane Warden. *The Art of Agile Development*. 2008. O'Reilly Media, Inc.

[33] Kent Beck, Cynthia Andres. 2004. *Extreme Programming Explained*. Addison-Wesley Professional.

[34] Jez Humble, David Farley. 2011. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, Inc.

[35] Michael Hüttermann. 2012. *DevOps for Developers*. Apress.

[36] Soon K. Bang, Sam Chung, Young Choh, y Marc Dupuis. 2013. *A grounded theory analysis of modern web applications: knowledge, skills, and abilities for DevOps*. En Proceedings of the 2nd annual conference on Research in information technology (RIIT '13). ACM, New York, NY, USA, 61-62. DOI= http://ezproxy.itcr.ac.cr:2075/10.1145/2512209.2512229

[37] Charles Anderson. 2015. *Docker*. IEEE Software 32, 3 (May-June 2015). 102-c3. 4 páginas. DOI: https://doi.org/10.1109/MS.2015.62.

[38] Ramtin Jabbari, Nauman bin Ali, Kai Petersen y Binish Tanveer. 2016. *What is DevOps?: A Systematic Mapping Study on Definitions and Practices*. En Proceedings of the Scientific Workshop Proceedings of XP2016 (XP '16 Workshops). ACM, New York, NY, USA, Artículo 12 , 11 páginas. DOI: http://ezproxy.itcr.ac.cr:2075/10.1145/2962695.2962707

[39] Adam Wiggins. *The Twelve-Factor App*. 2017. Obtenido de https://12factor.net

[40] Charles Betz, Amos O. Olagunju, and Patrick Paulson. 2016. *The Impacts of Digital Transformation, Agile, and DevOps on Future IT curricula*. In Proceedings of the 17th Annual Conference on Information Technology Education (SIGITE '16). ACM, New York, NY, USA, 106-106. DOI: http://ezproxy.itcr.ac.cr:2075/10.1145/2978192.2978205

[41] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2534860

[42] The Joint Task Force on Computing Curricula. 2015. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. Technical Report. ACM, New York, NY, USA.

[43] Heikki Topi, Helena Karsten, Sue A. Brown, João Alvaro Carvalho, Brian Donnellan, Jun Shen, Bernard C. Y. Tan, and Mark F. Thouin. 2017. *MSIS 2016: Global Competency Model for Graduate Degree Programs in Information Systems*. Technical Report. ACM, New York, NY, USA.

[44] Cenfotec 2011. *Educación de calidad para la industria de software*. ALTEC 2001, IX Seminario Latino-Iberoamericano de Gestión Tecnológica. Innovación Tecnológica en la Economía del Conocimiento, Costa Rica, 10. 2001.

[45] John Creswell. 2014. *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*. Cuarta Edición. SAGE Publications, Inc.

[46] Per Runeson, Martin Höst, Austen Rainer, Björn Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. 2012. Wiley.

Capítulo 6

Apéndice

Encuesta Inicial Propuesta

¿Se enseñan las siguientes Prácticas Agiles de Desarrollo en plan de estudios actual?			
Práctica	Sí	En Planes	No
Pruebas unitarias			
TDD			
BDD			
ATDD			
Integración Continua			
Estándares de codificación			
Puesta en producción continua			
Refactorización de código			
Propiedad compartida del código			
Diseño emergente			
DevOps			

Cuadro 6.1: Encuesta: Pregunta 1

¿En caso que se enseñen, cómo se introduce al estudiante en estos temas?	
Mediante un curso dedicado	
Como material de apoyo de otros cursos	
Como trabajo de investigación/proyectos	
Mediante talleres extra clase	
Mediante charlas técnicas extra clase	
Otro	

Cuadro 6.2: Encuesta: Pregunta 2

¿Tiene la universidad convenios con la industria para impartir charlas/talleres sobre estos temas?	
Sí	
No	
En Planes	

Cuadro 6.3: Encuesta: Pregunta 3

¿Qué importancia le da usted como colaborador a la a la enseñanza de las siguientes prácticas ? 1 - Ninguna, 2 - Poca, 3 - Regular, 4 - Importante, 5 - Muy Importante					
Práctica	1	2	3	4	5
Pruebas unitarias					
TDD					
BDD					
ATDD					
Integración Continua					
Estándares de codificación					
Puesta en producción continua					
Refactorización de código					
Propiedad compartida del código					
Diseño emergente					
DevOps					

Cuadro 6.4: Encuesta: Pregunta 4

¿Qué nivel de experiencia tiene de las siguientes prácticas? 1 - Ninguna, 2 - Menor a un año, 3 - 1 a 2 años, 4 - 3 a 5 años, 5 - Más de 5 años					
Práctica	1	2	3	4	5
Pruebas unitarias					
TDD					
BDD					
ATDD					
Integración Continua					
Estándares de codificación					
Puesta en producción continua					
Refactorización de código					
Propiedad compartida del código					
Diseño emergente					
DevOps					

Cuadro 6.5: Encuenta: pregunta 5

¿Cuáles son las principales limitantes a la hora de enseñar estas prácticas? 1 - Menos Importante, 4 - Más Importante	
Presupuesto	
Personal no entrenado	
Resistencia al cambio	
Poca literatura	

Cuadro 6.6: Encuesta: Pregunta 6

¿Se tiene planes de adoptar la enseñanza de estas prácticas en el futuro?	
Dentro de un año	
Dentro de dos años	
No se ha planeado	

Cuadro 6.7: Encuesta: Pregunta 7