

Adopción de prácticas ágiles de desarrollo de software en los planes de estudio de universidades de Costa Rica: revisión de la literatura

Carlos Martín Flores González, Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
Email: martin.flores@ieee.org

Resumen—Aquí va el resumen

Palabras Clave—IEEEtran, journal, L^AT_EX, magnetics, paper, template.

I. 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 [23]:

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 [16] como un conjunto de doce principios.

II. 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.

II-A. 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?

Este documento fue realizado durante el curso de Ingeniería de Software, impartido por el profesor Rodrigo Bogarín. Programa de Maestría en Computación, Instituto Tecnológico de Costa Rica. Segundo Semestre, 2017.

Recibido el 21 de setiembre del 2017.

2. ¿Qué información hay disponible acerca de la enseñanza de prácticas ágiles de desarrollo de software en carreras universitarias de TIC's en Costa Rica como en el extranjero?
3. ¿Cuáles son los beneficios, problemas y retos reportados en la enseñanza de estas prácticas de desarrollo ágil?
4. sobre practicas agiles e industria. Problemas, insercion laboral, etc

II-B. Estrategia de búsqueda

Con el fin de identificar el primer conjunto de artículos relevantes, se hizo una revisión preliminar con Google Scholar¹ porque con este motor de búsqueda se puede abarcar un 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 Online²
- Google Scholar

II-C. 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 enseñanza 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

¹<http://scholar.google.com>

²<https://www.safaribooksonline.com>

- Estudios que proporcionen evidencia sobre la enseñanza de prácticas ágiles de desarrollo de software y el impacto en la industria

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)

II-D. 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.

III. DESARROLLO ÁGIL DE SOFTWARE

Las metodologías de desarrollo ágil de software emergen al final de los 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 *Adaptive Software Development* [25]. Las metodologías ágiles se caracterizan por promover el desarrollo interactivo e incremental y la entrega frecuente de funcionalidad prioritaria para los clientes. Estas metodologías están dirigidas a equipos pequeños y altamente colaborativos. *Scrum* y XP son las metodologías que presentan mayores niveles de adopción [30]: 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 [16] (Apéndice A). 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.

III-A. Prácticas Ágiles de desarrollo de software

De acuerdo con [14], a pesar de la gran cantidad de literatura disponible sobre metodologías ágiles de software, mucha de la misma menciona poco acerca de las prácticas ágiles 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 que estas son una parte muy específica de una metodología que aborda varios aspectos. Algunos ejemplos conocidos son programación en parejas y reuniones diarias. A pesar de no haber una definición común en la literatura de práctica ágil [12], 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 [28].

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 [30], y el cual expone las prácticas ágiles en desarrollo de mayor aceptación. Las prácticas expuestas en este reporte se confrontaron con la literatura acerca de XP [14], [15], [28] 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

III-B. DevOps

DevOps es la práctica que combina desarrollo y operaciones y que desde el 2009 viene siendo adoptada por muchas organizaciones de la industria [9]. 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 [8]. DevOps apareció como una respuesta directa a los retos de la plataformas de software a gran escala que se actualizan rápidamente, tal y como lo indica Anderson [5]: *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 de operaciones”*.

III-C. DevOps vs Metodologías Ágiles

A pesar de no estar considerada como una metodología de desarrollo ágil como tal, autores como Bæbark [8] ven al DevOps como el siguiente paso natural del movimiento 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: nuevas funciones, 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 [22] se estudió la relación de las prácticas de DevOps con otras metodologías de desarrollo y se destaca con respeto a las metodologías ágiles que:

- DevOps extiende las metodologías ágiles: los principios de DevOps puede proporcionar una extensión pragmática 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 [22] 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, entrega y pruebas, el establecimiento de métricas y el intercambio de conocimientos y herramientas.

Esta relación de colaboración e interoperabilidad que se presenta entre DevOps y las metodologías ágiles también se expone en otra literatura consultada, como en [24]

En [22] también se expone que a pesar de las similitudes, DevOps no cumple con todos los principios propuestos en el Manifiesto Ágil [16] y que DevOps elimina la brecha entre desarrolladores y personal de operaciones mientras que las metodologías ágiles están orientadas a alinear requerimientos de negocio con el desarrollo.

De acuerdo con [22], 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 [31] es un ejemplo de la aplicación de las prácticas de DevOps identificadas en [22] adaptadas para desarrollo de aplicaciones de tipo *software como servicio* o SAAS³ por sus siglas en inglés.

IV. ENSEÑANZA DE PRÁCTICAS ÁGILES DE DESARROLLO DE SOFTWARE

De acuerdo con los resultados obtenidos a partir la búsqueda en la sección II-D, se evidencia que la enseñanza de prácticas

ágiles de desarrollo es un tema activo de investigación y desarrollo en universidades extranjeras. En [20] 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 [18] 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 los aspectos básicos del desarrollo ágil.

Aunque las formas y aplicación de la enseñanza de estos temas varían de institución en institución, prevalece el hecho de que la mejor forma de aprender desarrollo ágil es 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 graduación⁴ [13], proyectos/casos de estudio que se desarrollan durante un semestre [29], desarrollando juegos [26], por medio de laboratorios de robótica [27], Wikis [11] o bien como un tema paralelo al principal de un curso. Por ejemplo en [17], se introduce a los estudiantes al versionamiento de código como herramienta de apoyo al desarrollo de una aplicación Web. De acuerdo al estudio en [21] 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 desarrollo ágil pero con un mayor énfasis en la introducción en temas de automatización [8]–[10] de procesos de construcción, pruebas, entrega y computación en la nube. Un caso interesante es el que se expone en [19], “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.

IV-A. 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 proponer y adaptar recomendaciones en los planes de estudio debido al panorama cambiante de la tecnología [1].

En la guía curricular recomendada para los programas de bachillerato para la carrera de ciencias de la computación en [2] se incluyen temas tales como integración continua, automatización, control de versiones, estándares de codificación, desarrollo orientado a pruebas, pruebas unitarias y pruebas de integración como parte del cuerpo de conocimiento de

³Software-as-a-service

⁴Capstone project

ingeniería de software sugerido para esta carrera. Para la guía curricular recomendada para los programas de ingeniería de software [3] se incluyen temas de prácticas ágiles de desarrollo de software en las secciones de Validación y Verificación, y en la sección de Procesos asociados al desarrollo de software. También se pudo constatar la presencia 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 [4] las prácticas ágiles de desarrollo aparecen como parte de las competencias del área de desarrollo y entrega de sistemas.

Los autores de [10] 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 puede responder a las tendencias de desarrollo ágil y DevOps [32]. 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.

V. 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 [7] 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 [6] presentó un plan de estudios para el programa de Especialista en Tecnología de Software, el cual se diferenciaba de los carreras universitarias convencionales 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 [34] 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 [35], [36], sin embargo en este caso, la experiencia reportada no incluye 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 [7], [33] por medio proyectos integradores los cuales aparte de formar habilidades técnicas o fuertes, también pretenden formar en habilidades suaves como comunicación, creatividad, valores éticos, agilidad, educación continua, entre otros.

VI. BENEFICIOS Y RETOS REPORTADOS

VI-A. Beneficios

- Uso de herramientas de software de actualidad y un aprendizaje más alineado con las tendencias de la industria [7].
- 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 [21].
- En [20] reporta una mejora de las habilidades de programación y calidad del código.
- De acuerdo a [26] 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 [19].
- 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 [34].
- La introducción de prácticas ágiles de desarrollo fortaleció el proceso de pruebas, asegurando mayor calidad y evitando el re-trabajo [34].
- Las prácticas ágiles de desarrollo de software, al ser el aspecto más técnico en las metodologías ágiles ha resulta ser el más atractivo para los estudiantes y en donde concentran más sus esfuerzos de aprendizaje [29].
- 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 [35] 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 la forma de abordar el problema, y como consecuencia aumentar su motivación en el proceso de aprendizaje y de asumir mayor responsabilidad.

- Hazzan y Dubinsky [18] 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

VI-B. Retos

- Existen muy pocos artículos publicados acerca de la enseñanza de DevOps [8]
- En [29] se cita:
 - 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 [19].
- 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 [34]
- En [35] 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 [20] 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 orientado a objetos, algoritmos y estructuras de datos.

VI-C. Relevancia de Prácticas ágiles de desarrollo

Los reportes consultados sobre el estado actual de metodologías ágiles [30] y DevOps [37] reflejan una amplia aceptación de ambos enfoques en la industria de tecnología. Por ejemplo, en [30] el 94 % de las organizaciones que tomaron parte del estudio respondieron afirmativamente a la pregunta de si practicaban enfoques ágiles y el 71 % de las mismas reportó que están llevando a cabo iniciativas en DevOps. En [37] 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 practican DevOps tienen que estar concientes del uso de estas prácticas técnicas y no solamente de herramientas específicas.

La encuesta realizada en [20] 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 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 tienen 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 nuevas opciones en los planes de estudio de carreras de tecnología de información de las universidades de ese estado [32]. 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 Rademacher y Walia [38] 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 [38] 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 mencionó 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 avanzados mostraron poca habilidad de uso de herramientas para cobertura 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 respecto al *debugging* de código.

VI-C1. En Costa Rica:

VII. CONCLUSIÓN

La conclusión

APÉNDICE A EL MANIFIESTO ÁGIL

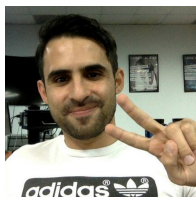
*Individuos e interacciones sobre procesos y herramientas
Software funcionando sobre documentación extensiva
Colaboración con el cliente sobre negociación contractual
Respuesta ante el cambio sobre seguir un plan*

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

REFERENCIAS

- [1] ACM. 2017. *Curricula Recommendations*. Obtenido de: <http://www.acm.org/education/curricula-recommendations>
- [2] 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>
- [3] The Joint Task Force on Computing Curricula. 2015. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. Technical Report. ACM, New York, NY, USA.
- [4] Heikki Topi, Helena Karsten, Sue A. Brown, João Alvaro Carvalho, Brian Donnellan, Jun Shen, Bernard C. Y. Tan, and Mark F. Thoun. 2017. *MSIS 2016: Global Competency Model for Graduate Degree Programs in Information Systems*. Technical Report. ACM, New York, NY, USA.
- [5] Charles Anderson. 2015. *Docker*. IEEE Software 32, 3 (May-June 2015). 102-c3. 4 páginas. DOI: <https://doi.org/10.1109/MS.2015.62>
- [6] 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.
- [7] Cenfotec. 2016. *“Currículos basados en proyectos integradores.pdf” y póster “Proyectos integradores en U Cenfotec.pdf”*. SIIC Simposio Internacional sobre Innovaciones Curriculares, UCR, 2016.
- [8] 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>
- [9] 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>
- [10] 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>
- [11] Marija Cubric. 2008. *Agile learning & teaching with wikis: building a pattern*. In Proceedings of the 4th International Symposium on Wikis (WikiSym '08). ACM, New York, NY, USA, , Article 28 , 2 pages. DOI: <http://ezproxy.itcr.ac.cr:2075/10.1145/1822258.1822296>
- [12] 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>
- [13] 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.
- [14] 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
- [15] Kent Beck, Cynthia Andres. 2004. *Extreme Programming Explained*. Addison-Wesley Professional.
- [16] Kent Beck, Mike Beelde, Arie van Bennekum, Allistair Cockburn, et al. 2001. *Manifiesto for Agile Software Development*. Agile Alliance.
- [17] 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>
- [18] 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>
- [19] 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>
- [20] Martin Kropp, Andreas Meier. 2013. *Teaching Agile Software Development at University Level: Values, Management, and Craftsmanship*. En Software Engineering Education and Training (CSE&T). Páginas 179 - 188. DOI: <https://ezproxy.itcr.ac.cr:2878/10.1109/CSEET.2013.6595249>
- [21] 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>
- [22] 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>
- [23] 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>
- [24] Mitesh Soni. 2016. *DevOps for Web Development*. Packt Publishing
- [25] 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>

- [26] 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>
- [27] 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>
- [28] James Shore y Shane Warden. *The Art of Agile Development*. 2008. O'Reilly Media, Inc.
- [29] 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>
- [30] Version One. *11th Annual State of Agile Report*. 2017. Obtenido de <http://stateofagile.versionone.com/>
- [31] Adam Wiggins. *The Twelve-Factor App*. 2017. Obtenido de <https://12factor.net>
- [32] 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
- [33] Ignacio Trejos, Alvaro Cordero. 2017. *Learn-by-doing-collaboratively across the curriculum: Integrative projects at UCenfotec*. IEEE World Engineering Education Conference – EDUNINE2017. 3.
- [34] 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.
- [35] 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>
- [36] 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>
- [37] Puppet Labs, DevOps Research & Assessment(DORA). 2017. *State of DevOps*. Reporte. Obtenido de <https://puppet.com/resources/whitepaper/state-of-devops-report>
- [38] 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>



Martín Flores es Ingeniero en Informática de la Universidad Nacional. Actualmente, realiza sus estudios de Maestría en Ciencias de la Computación del Tecnológico de Costa Rica. Sus principales intereses son: lenguajes de programación, ingeniería de software y *DevOps*.