La Percepción de los Estudiantes sobre Scrum usando un Proyecto en un Curso de Ingeniería de Software

Guillermo Rodríguez, Isabela Gasparini, Avanilde Kemczinski, Alexandre Veloso

CÓMO REFERENCIAR ESTE ARTÍCULO:

G. Rodríguez, I. Gasparini, A. Kemczinski and A. Veloso de Matos, "Students' Perception of Scrum in a Course Project," in IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 16, no. 4, pp. 329-336, Nov. 2021, doi: 10.1109/RITA.2021.3136436.

***Title*— Students' Perception of Scrum in a Course Project**

***Abstract*— Gaining insight into what students experience in a course in software engineering is essential to the continuous development of the course design. The growing use of agile approaches in professional settings has facilitated their introduction into training and undergraduate courses in software engineering. This paper reports a case study of student participation in a course project by using a previously presented training model. We aimed to assess how students felt with the training model when conducting the 5 Scrum events: Sprint Planning, Daily Meeting, Sprint, Sprint Review, and Sprint Retrospective. To achieve our goal, we applied a 22-item survey to 31 undergraduate students to gather students' opinions. Results showed that most students had a positive opinion about team communication, the position of Scrum Master, Sprint goals definition, and guidance using meetings as a means of checkpoints.**

***Index Terms*— Agile software development, Computer science education, Scrum, Course project.**

I. Introducción

LOS proyectos en la educación de Ingeniería de Software tienen como objetivo capacitar a los estudiantes para abordar situaciones y desafíos que enfrentarían en contextos profesionales. En consecuencia, los cursos de pregrado requieren un desarrollo y actualización continuos para satisfacer las demandas de la industria tecnológica y garantizar la calidad académica. Las universidades acercarán la diferencia entre las circunstancias que se enseñan en el aula y las que ocurren dentro de la industria en este sentido. Por lo tanto, Scrum da como resultado una estructura eficaz y versátil para preparar a los estudiantes para incorporar prácticas calificadas de ingeniería de software [1].

La enseñanza de la Ingeniería de Software ha sido un tema ampliamente discutido; Dawson [2] y Regev et al. [3] han manifestado que los proyectos utilizados en la docencia se alejan de la realidad, formando estudiantes con una preparación limitada para su futuro profesional. Según Lethbridge et al. [4], las empresas a veces tienen que complementar los conocimientos de los estudiantes con capacitación para proporcionar las habilidades necesarias para el desarrollo de software que demanda la industria. Según Yamaguti et al. [5], en la actualidad, existe un desafío importante en la enseñanza de la Ingeniería de Software, debido a la demanda de profesionales calificados para desarrollar software de manera eficiente. Según Yamaguti et al., Deberían adoptarse enfoques innovadores para permitir la integración de contenido a través de un enfoque interdisciplinario [5]. Algunas investigaciones han evidenciado la necesidad de enseñar desarrollo de software, simulando un contexto profesional [6, 7].

Existen numerosas explicaciones para seleccionar Scrum como la técnica ágil en los cursos de SE. En primer lugar, ha crecido en los últimos años; en 2019, el 72% de los encuestados utilizó Scrum o un híbrido que incluye Scrum [8]. Además, este enfoque ágil se centra en las técnicas de gestión de proyectos que promueven el seguimiento del progreso del proyecto, los comentarios de las partes interesadas y el compromiso del equipo. En segundo lugar, Scrum respalda el proceso de cumplimiento de requerimientos, porque si bien todos los requerimientos permanecen en el backlog, en Sprint, solo se cumple un conjunto limitado. En tercer lugar, las reuniones retrospectivas y el contacto continuo con el cliente permiten el reconocimiento temprano de impedimentos y refinamientos del plan de trabajo.

Este artículo informa un caso de estudio realizado con estudiantes de pregrado de un curso de Ingeniería de Software para evaluar cómo se sintieron los estudiantes con nuestro modelo de capacitación previamente informado al realizar los eventos Scrum, a saber, Planificación de Sprint (*Sprint Planning*), Sprint, Reuniones diarias (*Daily Meetings*), Revisión de Sprint (*Sprint Review*) y Retrospectiva de Sprint (*Sprint Retrospective*). Se pidió a los estudiantes que

Guillermo Rodríguez, UADE-INTEC, Buenos Aires, Argentina (guirodriguez@uade.edu.ar), https://orcid.org/0000-0003-4125-3998
Isabela Gasparini, UDESC, Santa Catarina, Brasil (isabela.gasparini@udesc.br)
Avanilde Kemczinski, UDESC, Santa Catarina, Brasil (avanilde.kemczinski@udesc.br)
Alexandre Veloso, UDESC, Santa Catarina, Brasil (alexandre.matos@udesc.br).

completaran una encuesta para lograr nuestro objetivo. Los datos recopilados y procesados mostraron que los estudiantes estuvieron de acuerdo en que nuestro modelo los ayudó a reforzar la comunicación del equipo y la comunicación con el Product Owner, comprender mejor el rol del Scrum Master, cómo lograr los objetivos de Sprint y aprovechar la retroalimentación de las reuniones de control, entre otras habilidades no técnicas. Además, los estudiantes estuvieron de acuerdo en que habían mejorado la comprensión de Scrum siguiendo nuestro modelo de formación.

El resto de este artículo está organizado de la siguiente manera. Scrum y sus enseñanzas se enumeran en la Sección II. La Sección III describe el rol del Coach ágil. La Sección IV presenta nuestra metodología de investigación. La sección V ilustra el desarrollo del estudio de caso. La sección VI analiza los resultados de la encuesta. La sección VII establece las direcciones para estudios futuros. Finalmente, este artículo se concluye en la Sección VIII.

II. Enseñanza de Scrum

El advenimiento del Manifiesto de desarrollo de software ágil (https://agilemanifesto.org/) trajo consigo mejoras dentro de la cultura de desarrollo de software. Esencialmente, el Manifiesto esboza una nueva perspectiva sobre el desarrollo de software que se centra en la movilidad, la flexibilidad, las habilidades sociales y el potencial en poco tiempo de ofrecer nuevos productos y servicios de alto valor al mercado [9].

Scrum es un enfoque iterativo e incremental que organiza tareas para hacerlas manejables para equipos pequeños, autoorganizados y multifuncionales, además de permitir a los equipos sistematizar proyectos de desarrollo de software, fomentando buenas prácticas de ingeniería de software al fomentar la cooperación en el trabajo en equipo [12].

Los roles en Scrum son bastante diferentes de los métodos de software tradicionales. Los roles y expectativas claramente definidos permiten a las personas realizar sus tareas de manera eficiente. Hay tres roles: Product Owner, Development Team y Scrum Master. Juntos, estos se conocen como Scrum Team. El Product Owner representa al cliente y controla el Backlog del Proyecto, y está trabajando en estrecha colaboración con el Equipo de Desarrollo para brindar claridad y aceptación a la historia del usuario. Completar las historias de usuario es responsabilidad del equipo de desarrollo. El Scrum Master es responsable de iniciar el proceso, ayudar al propietario del producto a preparar lanzamientos y ayudar al éxito del equipo de desarrollo eliminando impedimentos y poniendo los recursos a disposición.

El uso generalizado de Scrum ha mostrado una disparidad entre la orientación académica y las especificaciones de la industria del software. Más allá de centrarse principalmente en la formación y las certificaciones, la enseñanza de Scrum se ha convertido en una piedra angular de los cursos de pregrado de ingeniería de software [15, 38]. Si bien Scrum ha sido parte de los programas de grado en Ingeniería de Software, no hay suficiente material instructivo que cubra procesos, técnicas y patrones de interacción personal dentro de la industria del software, lo que promovería significativamente la incorporación de los estudiantes a la formación profesional.

Teniendo en cuenta que la adopción industrial de Scrum se está convirtiendo en un estándar [13, 14], es importante destacar por qué el desarrollo ágil de software debe enseñarse a nivel universitario. El desarrollo iterativo e incremental de Scrum, junto con las reuniones diarias y retrospectivas, permite a los estudiantes proporcionar información temprana sobre cómo se realizan las prácticas ágiles; también permiten la identificación oportuna de problemas en la comprensión de las prácticas ágiles y el concepto de medidas correctivas a ser tomadas en sprints posteriores por parte de los estudiantes. A diferencia de los enfoques en cascada, los estudiantes reciben comentarios sin esperar que el proyecto termine. Por otro lado, podría ser que los equipos de estudiantes carezcan de las habilidades suficientes para desarrollar un proyecto, particularmente cuando se presenta Scrum por primera vez.

Tomando las pautas del Currículo de Ingeniería de Software 2004 - *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering* (http://sites.computer.org/ccse/SE2004Volume.pdf), los estudiantes deben ser entrenados en el trabajo en equipo ya que el desarrollo ágil de software se enfoca en el trabajo en equipo. Además, dado que los enfoques ágiles facilitan los procesos de aprendizaje [42], apoyamos la idea de que la enseñanza del desarrollo ágil de software podría reducir la complejidad cognitiva de los procesos de desarrollo de software y fomentar la adquisición de habilidades al hacer que Scrum sea más comprensible y adecuado para estudiantes de pregrado. El uso de proyectos de cursos basados en Scrum se ha introducido como una herramienta para enseñar conceptos básicos de ingeniería de software [15].

Un proyecto de curso generalmente se lleva a cabo en el aula y es supervisado por maestros. Este proyecto tiene como objetivo aumentar la participación y protagonismo de los estudiantes en el proceso de aprendizaje y abordar no solo problemas comunes que se encuentran en el desarrollo de software, sino también otros conceptos sugeridos por el Manifiesto Ágil [16, 17]. En [18], los autores comparan un proyecto de curso desarrollado con un enfoque ágil con sus proyectos anteriores desarrollados con un enfoque no ágil. En [19], los autores describen su experiencia de ocho años enseñando metodologías de desarrollo ágil a diferentes grupos de estudiantes en varias universidades. A partir de la experiencia adquirida en la enseñanza del desarrollo ágil de software, sugirieron cómo resolver posibles problemas.

Scrum ha sido un método ágil ampliamente utilizado, junto con otros enfoques ágiles [20, 21]. El logro de los objetivos de enseñanza y la evaluación del desempeño de los estudiantes en la estimación y planificación de habilidades en proyectos basados en Scrum se tratan en [22]. Además, la percepción de los estudiantes de Scrum por primera vez se estudió en [15]. Una de las ventajas de Scrum es que ayuda a los estudiantes a proporcionar un mejor ambiente para el trabajo en equipo y una mejor comunicación, lo que da como resultado productos de alta calidad [10].

Cada una de las estrategias anteriores comparte el hecho de que existen roles de gestión fuera del control del Scrum Team, como la gestión de recursos financieros, la toma de decisiones corporativas y la gestión medioambiental de la organización [20].

En trabajos anteriores, hemos introducido un modelo de formación original basado en Scrum enriquecido con el rol de Coach Agile para optimizar el rendimiento de los estudiantes. Comparamos dos grupos de equipos de estudiantes (equipos con coaching no ágil y equipos con coaching ágil) en términos de cobertura de las prácticas de ingeniería de software y las percepciones de los estudiantes sobre el uso de Scrum por primera vez. Los resultados han revelado que los equipos entrenados ágilmente superaron a los no entrenados ágilmente. Además, los estudiantes entrenados obtuvieron información valiosa sobre la internalización de Scrum, la resolución de problemas y la orientación a través de reuniones de puntos de control [25]. A diferencia de este trabajo anterior, en la investigación actual, nos enfocamos en medir cómo se sintieron los estudiantes entrenados con nuestro modelo de capacitación al realizar los 5 eventos Scrum: Sprint Planning, Daily Meeting, Sprint, Sprint Review, y Sprint Retrospective.

III. Coaching Ágil

La comunicación y la colaboración son habilidades centrales para el desarrollo ágil. Por lo tanto, afirmamos que aumentar los roles de Scrum con el Coach ágil ayuda a los profesores a capacitar a los estudiantes en la complejidad de los aspectos del desarrollo de software, a profundizar su comprensión de un entorno basado en Scrum y a aprender habilidades de coaching y liderazgo [23]. Integrar el coaching en un proyecto de curso, simulando un contexto profesional, sería útil para ofrecer a los estudiantes la oportunidad de practicar el uso de Scrum en un entorno seguro y controlado. A pesar de los conocidos beneficios del coaching, se ha demostrado que los enfoques eficaces del coaching ágil son difíciles de aplicar incluso en el ámbito académico [25].

El coaching ágil implica tener en cuenta desafíos importantes dentro y fuera de los equipos de estudiantes, eliminando impedimentos y promoviendo la implementación de conceptos ágiles por parte de profesores, líderes de equipo y Scrum Masters [24]. Si bien todos los trabajos académicos mencionados anteriormente han reconocido que el coaching ágil es una consideración importante, el efecto del coaching en el éxito de los estudiantes en el desarrollo de software no se ha analizado adecuadamente. El Coach Ágil tiene como objetivo ayudar y fortalecer el crecimiento de los estudiantes al servir como mentor en el desarrollo de prácticas de ingeniería de software. También vale la pena mencionar que el Coach Ágil no debe estar involucrado en el proyecto y es responsable de mantener un compromiso justo entre la autoorganización y el coaching ágil.

Es importante aclarar que cada equipo de estudiantes realizó reuniones diarias generalmente realizadas por WhatsApp (https://www.whatsapp.com/?lang=es) o Slack (https://slack.com/intl/es-ar/). Cada equipo decidió utilizar la herramienta que mejor se adaptaba a sus necesidades. Además, una vez a la semana, los equipos de estudiantes, el Product Owner y el Coach ágil llevaron a cabo una reunión de control denominada Weekly Meeting.

IV. Metodología

En esta investigación, nuestro objetivo es medir cómo los estudiantes percibieron nuestro modelo de entrenamiento basado en Scrum reportado anteriormente como un vehículo para cumplir efectivamente los objetivos de los eventos Scrum [25].

Objetivo de la Investigación: El objetivo de la investigación es determinar hasta qué punto el modelo de formación es eficaz para ayudar a los estudiantes a cumplir los objetivos de los eventos Scrum. Sobre la base del objetivo de la investigación, se formulan las siguientes preguntas de investigación (PI).

PI1: ¿El modelo de formación enriquecido con el rol de Coach ágil tiene un impacto positivo en el rendimiento de la reunión de planificación de Sprint?

PI2: ¿El modelo de entrenamiento enriquecido con el rol de Coach Ágil promueve una comprensión clara de Scrum a lo largo del Sprint?

PI3: ¿El modelo de formación enriquecido con el rol de Coach ágil aumenta el compromiso del equipo durante las Weekly Meetings?

PI4: ¿El modelo de capacitación enriquecido con el rol de Coach ágil permite a los estudiantes mejorar el desempeño en las reuniones de Sprint Review?

PI5: ¿El modelo de formación enriquecido con el rol de Coach Ágil alienta a los estudiantes a ser autocríticos y a establecer cursos de acción para mejorar en sprints posteriores en reuniones de Sprint Retrospective?

V. Diseño del Caso de Estudio

Hemos presentado un modelo de entrenamiento basado en Scrum en un estudio anterior, enriquecido con el rol del Coach Ágil [25]. Realizamos un proyecto de curso con estudiantes de pregrado para desarrollar nuestro caso de estudio utilizando el modelo de formación en el contexto de un curso de Ingeniería de Software. Los roles de Product Owner y Agile Coach fueron desempeñados por dos profesores diferentes. Dentro de cada equipo de estudiantes, un estudiante desempeñó el papel de Scrum Master.

Realizamos el caso de estudio en el contexto del curso Taller de Ingeniería de Software en el segundo semestre de 2018. Este curso electivo de un semestre se puede tomar en el último año del programa de Ingeniería de Sistemas de la Facultad de Ciencias Exactas (Departamento de Ciencias de la Computación-UNCPBA). El curso tiene como objetivo fortalecer la comprensión de Scrum de los estudiantes y simular un contexto profesional en el que deben desarrollar un proyecto de curso.

El proyecto del curso consistió en diseñar un campus virtual utilizando el motor de juego Unity (https://unity.com/) construido como arquitectura con cliente-servidor de múltiples niveles. El campus virtual permite a los usuarios acceder a las instalaciones de la Universidad, jugar juegos temáticos y asistir a clases virtuales, utilizando canales de networking para chat, correo electrónico y sitio web.

Siempre que los estudiantes planificaran y coordinaran el Backlog del producto con el Product Owner, el Coach ágil trabajó para cultivar las relaciones interpersonales dentro de los equipos para que cada miembro del equipo conociera las fortalezas y debilidades de sus socios. Además, el Coach ágil se centró en la importancia de comprender las historias de los

usuarios y la visión del proyecto utilizando el Product Backlog antes mencionado.

El Coach ágil ayudó a los equipos a abordar épicas (es decir, una gran historia de usuario que no se puede entregar como se define en una sola iteración o es lo suficientemente grande como para dividirse en historias de usuario más pequeñas), incorporar estándares no funcionales y definir los criterios de finalización. Por ejemplo, el Coach ágil puede enfatizar el requisito de disponibilidad no funcional: "Como usuario, quiero que el mundo virtual esté disponible el 99,99% del tiempo y trato de acceder a él para no frustrarme". Cuando finaliza la historia de usuario, el "criterio de done" se confirma; en el ejemplo, la historia del usuario se compara con un usuario que desea iniciar sesión en el mundo virtual sin registrarse. Además, era importante centrarse en el objetivo del sprint, que permitirá a los equipos lograr el sprint. Al trabajar hacia el mismo objetivo, los estudiantes ganaron motivación, apreciaron trabajar al darse cuenta de cómo estaban contribuyendo, tomaron mejores decisiones y asumieron la responsabilidad al visualizar lo que deberían hacer junto con el proyecto del curso.

Las siguientes subsecciones describen la preparación del diseño del estudio de caso (subsección A), la composición de la encuesta (subsección B) y el perfil de los estudiantes que participaron en el experimento (subsección C).

A. *Preparación*

Al final del proyecto del curso en el segundo trimestre de 2018, llevamos a cabo una encuesta para recopilar la percepción de los estudiantes sobre Scrum y cómo se sentían simulando un contexto profesional. La encuesta de 22 ítems en español fue anónima y administrada por un formulario de Google. Para construir la encuesta, usamos una escala Likert de seis puntos [26]: absolutamente de acuerdo (puntaje 6), de acuerdo (puntaje 5), algo de acuerdo (puntaje 4), algo en desacuerdo (puntaje 3), en desacuerdo (puntaje 2) o totalmente en desacuerdo (puntuación 1).

Para preparar mejor a los estudiantes para completar la encuesta, agregamos algunas preguntas generales de "pre-calentamiento" colocadas al comienzo de la encuesta, preguntando, por ejemplo, qué es Scrum y qué tipo de situaciones se benefician realmente de él. Luego, incluimos varios elementos de consulta diseñados para recopilar las opiniones de los estudiantes con respecto a cómo nuestro modelo propuesto [25] puede ayudar efectivamente a los estudiantes a realizar eventos Scrum a lo largo de un proyecto de curso. Siguiendo el enfoque de Likert para construir cuestionarios [26], los ítems no eran preguntas sencillas sino declaraciones con las que los estudiantes podían estar totalmente de acuerdo, de acuerdo, algo de acuerdo, algo en desacuerdo, en desacuerdo o totalmente en desacuerdo. En este sentido, los estudiantes no se sintieron evaluados sino consultados. A diferencia de otros estudios que han utilizado una escala de acuerdo de número impar (por ejemplo, [34, 35]), decidimos emplear una escala de número par para captar mejor las opiniones de los estudiantes (sin punto medio neutral). Además, cada elemento tenía que recibir un argumento textual breve pero completo. También reservamos un área de texto que podría usarse para designar cualquier comentario adicional. Además, la selección de la escala de números pares se debe al hecho de que esta escala elimina la posibilidad de que los encuestados hagan un mal uso del punto medio [37]. Además, la selección de la escala Likert de 6 puntos está respaldada por [36], en la que los autores concluyen una mayor ganancia con escalas que abordan alrededor de seis categorías de respuestas.

B. *Encuesta*

La encuesta consta de 22 preguntas (Q) de la siguiente manera. Las historias de usuarios no se aclararon bien en el momento de la creación de la lista de productos pendientes (Q1). La evolución de la cartera de productos siempre fue clara, rastreable y visible (Q2). Planning Poker de historias de usuarios más pequeñas no mejoró la participación de los estudiantes (Q3). El trabajo en equipo y la comunicación entre los miembros del equipo fueron buenos (Q4). Como miembro del equipo, considero que el Scrum Master no desempeñó bien su papel (Q5). Las respuestas de los estudiantes en Reunión Semanal fueron más precisas en relación a las sugerencias recibidas anteriormente (Q6). Las reuniones semanales fueron cruciales para seguir y monitorear el progreso del proyecto del curso (Q7). La exposición de métricas ayudó a los miembros del equipo a reflexionar sobre lo sucedido (Q8). La comunicación con el Product Owner fue buena (Q9). Las reuniones de Sprint Review no ayudaron a recibir comentarios del cliente y del Product Owner (Q10). Las reuniones retrospectivas de Sprint permitieron identificar mejoras para el próximo Sprint (Q11). El entorno de desarrollo obstaculizó el ciclo de vida de User Stories (Q12). El estado inicial de cada Sprint se acordó satisfactoriamente (Q13). Los goles de cada Sprint no se elevaron de forma satisfactoria (Q14). Las personas involucradas en la determinación del enfoque de control de cambios fueron las correctas (Q15). Los encargados de evaluar el procedimiento y el control de calidad del producto fueron los adecuados (Q16). Los recursos de formación desarrollados fueron ineficaces (Q17). El alcance de los objetivos de cada Sprint se manejó de manera satisfactoria (Q18). La planificación de la implementación al final de cada Sprint fue aceptable (Q19). En el futuro, manejaría a mi equipo de software profesional de manera muy diferente (Q20). Los riesgos e impedimentos no se identificaron, gestionaron, resolvieron ni mitigaron satisfactoriamente de forma eficaz (Q21). Las reuniones previas a la demostración de Sprint fueron útiles para preparar el producto y entrenar el discurso necesario para la presentación (Q22).

Para evitar que los estudiantes marquen la misma puntuación Likert cuando completan la encuesta [27], invertimos la dirección de los ítems de consulta negativos (Q1, Q3, Q5, Q10, Q12, Q14, Q17, Q20 y Q21).

La Tabla 1 muestra la agrupación de elementos organizados según los 5 eventos de Scrum: Sprint Planning, Daily Scrum Meeting (y Weekly Meeting), Sprint, Sprint Retrospective y Sprint Review. La encuesta fue diseñada en términos de los eventos de Scrum, en los que los profesores, ya sea Product Owner o Agile Coach, se involucraron con los equipos de estudiantes. Aunque Daily Scrum Meeting y Sprint Retrospective son eventos en los que solo participan miembros

del equipo, se informó a los estudiantes sobre cómo se deben realizar estas reuniones y qué artefactos se deben generar después de las reuniones.

TABLA I
AGRUPACIÓN DE LAS PREGUNTAS SEGÚN LOS EVENTOS DE SCRUM.

Evemto de Scrum	Preguntas
Sprint Planning	Q1, Q2, Q3, Q13
Daily Meeting	Q6, Q7, Q12, Q17
Sprint	Q4, Q5, Q9, Q15, Q16
Sprint Review	Q10, Q19, Q22
Sprint Retrospective	Q8, Q11, Q14, Q18, Q20, Q21

C. Perfil de los estudiantes

Los 31 estudiantes que participaron en el experimento eran 23 hombres y 8 mujeres y tenían un promedio de alrededor de 25 años. Los estudiantes ya habían superado los siguientes cursos en este nivel, a saber, diseño de sistemas de software, programación orientada a objetos, sistemas operativos, metodologías ágiles y gestión de bases de datos. Sus habilidades técnicas incluían C #, Java, UNIX, Scrum y SQL.

Los estudiantes se agruparon aleatoriamente en equipos de seis o cinco miembros, seis equipos en total, para replicar un escenario profesional en términos de tamaño y diversidad del equipo, con el fin de llevar a cabo el proyecto del curso. Permitimos a los estudiantes tomar sus propias decisiones y aprender de sus errores, lo cual es un beneficio invaluable de la experiencia de un proyecto de curso.

VI. RESULTADOS Y DISCUSIÓN

Las siguientes subsecciones proporcionan un análisis cualitativo (subsección A) y cuantitativo de las opiniones de los estudiantes (subsección B).

A. Análisis Cualitativo de las opiniones de los estudiantes

Considerando las preguntas con los niveles más altos de acuerdo, resumimos que con respecto a la reunión de Sprint Planning, la mayoría de los estudiantes no están de acuerdo o algo en desacuerdo con las ideas expresadas en Q1, Q2, Q3 y Q13. Como Q1 y Q3 se formularon con un sentido negativo, podemos afirmar que la mayoría de los estudiantes están de acuerdo con la idea de que las User Stories se aclararon bien en el momento de la creación de Product Backlog (Q1, 27,07%) y Planning Poker de historias de usuarios más pequeñas mejoran participación de los estudiantes (Q3, 57,04%). En cuanto a las Weekly Meetings, el 29,03% coincide en que “Las respuestas de los alumnos en las Weekly Meetings fueron más precisas respecto a las sugerencias recibidas anteriormente (Q6)”.

En cuanto a Sprint, el 61,3% de los estudiantes está de acuerdo en “El trabajo en equipo y la comunicación entre los miembros del equipo fue buena (Q4)”, el 74,2% está de acuerdo en “Como miembro del equipo, considero que el Scrum Master no cumplió bien su papel (Q5) ”, Y el 51,61% está de acuerdo en“ La comunicación con el Product Owner fue buena (Q9) ”.

En cuanto a la reunión de Revisión de Sprint, el 32,25% está de acuerdo en “La planificación de la implementación al final de cada Sprint fue aceptable (Q19)” y, finalmente, el 67,74% está de acuerdo en “Las reuniones antes de la demostración de Sprint fueron útiles para preparar el producto y entrenar el discurso necesario para la presentación (Q22)”.

Finalmente, en cuanto a la reunión de Sprint Retrospective, el 48,38% está de acuerdo en "Las reuniones de Sprint retrospectiva permitieron identificar mejoras para el próximo Sprint (Q11)", el 45,16% está de acuerdo en "Los objetivos de cada Sprint no se plantearon de manera satisfactoria (Q14)", 64,51 % coincide en “El alcance de los objetivos de cada Sprint se manejó satisfactoriamente (Q18)”.

B. Análisis Cuantitativo de las opiniones de los estudiantes

El análisis antes mencionado se apoya estadísticamente en esta subsección. Para cuantificar las opiniones de todos los participantes, se determinaron las puntuaciones Likert de los estudiantes por ítem de pregunta. A continuación, probamos la precisión interna de la muestra utilizando el coeficiente alfa ($\alpha = 0,67$) de Cronbach [28].

Para responder a la pregunta de investigación descrita anteriormente, formulamos las siguientes hipótesis nulas (H1 a H5):

H1: El modelo de formación enriquecido con el rol del Coach ágil no afecta el rendimiento de la reunión de planificación de Sprint (la puntuación de los estudiantes es menor o igual que 3)

H2: El modelo de entrenamiento falla al no promover una comprensión clara de Scrum a lo largo del Sprint (la puntuación de los estudiantes es menor o igual a 3),

H3: No existe correlación entre la aplicación del modelo de formación y el compromiso del equipo durante la Weekly Meeting (puntuación de los alumnos menor o igual a 3),

H4: el modelo de formación no permite que los estudiantes mejoren el rendimiento en las reuniones de Sprint Review (la puntuación de los estudiantes es menor o igual que 3),

H5: No existe correlación entre la aplicación del modelo de formación y la capacidad del alumno para ser autocrítico y establecer cursos de acción para mejorar en sprints posteriores (puntuación de los alumnos menor o igual a 3).

Antes de realizar la prueba t de Student, se verificó tanto el supuesto de normalidad en las puntuaciones de los estudiantes como el supuesto de homogeneidad de las varianzas. El supuesto de normalidad se verificó mediante la prueba de normalidad de Shapiro-Wilk. El supuesto de homogeneidad de las varianzas se verificó mediante la prueba de Levene, así como los resultados de la prueba t de Student. Este último muestra que el valor p es menor que 0,0001. Los valores en las puntuaciones de los estudiantes responden a una distribución normal (Fig. 1). Los valores t muestran que 9 (en negrita en la Tabla 2) de 22 hipótesis fueron rechazadas al nivel de significancia de $p < 0,0001$.

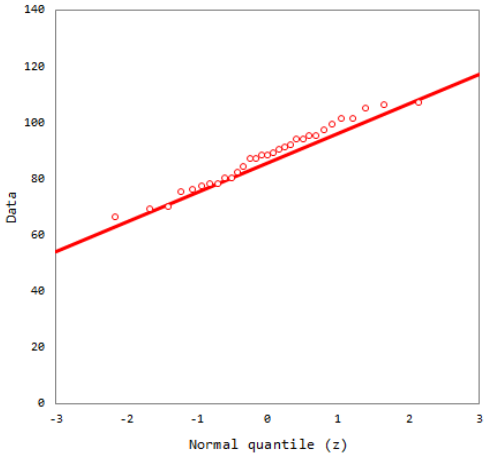

Fig. 1 Distribución normal de las puntuaciones de los estudiantes

TABLA II
ANÁLISIS DE LOS PUNTAJES OBTENIDOS EN LA ENCUESTA (P-VALOR < 0,0001)

Pregunta	Mínimo	Máximo	Media	Mediana	Mode	Desv. Std.	Student t	p-valor
Q1	4	6	3,61	5	4	1,14	2,97	0,002
Q2	3	6	3,58	4	4	1,14	2,8157	0,004
Q3	4	5	3,16	3	3	1,67	0,5361	0,297
Q4	**2**	**6**	**4,32**	**5**	**4**	**1,70**	**4,3293**	**<,0001**
Q5	**2**	**6**	**4,93**	**4**	**5**	**1,18**	**9,1211**	**<,0001**
Q6	**4**	**6**	**4**	**5**	**5**	**1,095**	**5,0839**	**<,0001**
Q7	3	6	4,12	5	4	1,66	3,7684	0,0003595
Q8	4	5	2,80	3	4	1,74	-0,6192	0,2702295
Q9	**3**	**6**	**4,48**	**5**	**4**	**1,2**	**6,8414**	**<,0001**
Q10	2	6	4,25	5	5	1,67	4,1881	0,0001135
Q11	**4**	**6**	**4,29**	**4**	**4**	**1,46**	**4,9042**	**<,0001**
Q12	3	6	3,58	4	4	1,64	1,9608	0,0296225
Q13	4	6	4,09	5	4	1,46	4,1577	0,0001235
Q14	**2**	**6**	**4,19**	**5**	**4**	**1,42**	**4,6658**	**<,0001**
Q15	2	6	4	4	4	1,34	4,1494	0,0001265
Q16	4	6	3,54	4	4	1,67	1,828	0,038755
Q17	3	6	3,83	4	5	1,46	3,1926	0,0016505
Q18	**4**	**6**	**4,80**	**5**	**4**	**1,16**	**8,6188**	**<,0001**
Q19	**2**	**6**	**4,09**	**4**	**5**	**1,04**	**5,8496**	**<,0001**
Q20	2	6	3,54	4	4	1,31	2,3267	0,013461
Q21	4	6	3,54	4	5	1,2	2,5307	0,008432
Q22	**3**	**6**	**4,93**	**5**	**5**	**1,34**	**8,0411**	**<,0001**

Debido a que las calificaciones de los estudiantes fueron sustancialmente superiores a 3, podemos aceptar hipótesis alternativas, lo que significa que las impresiones de nuestro modelo de capacitación en relación con los eventos de Scrum por parte de los estudiantes son optimistas. La mayoría de los estudiantes estuvieron de acuerdo en que la experiencia de aprendizaje les permitió mejorar la comunicación del equipo, comprender el rol de Scrum Mater, comprender y lograr los objetivos de Sprint, y monitorear el proceso y el proyecto mediante reuniones de puntos de control, como Daily Meetings (y Weekly Meetings), Sprint Review, y Sprint Retrospective.

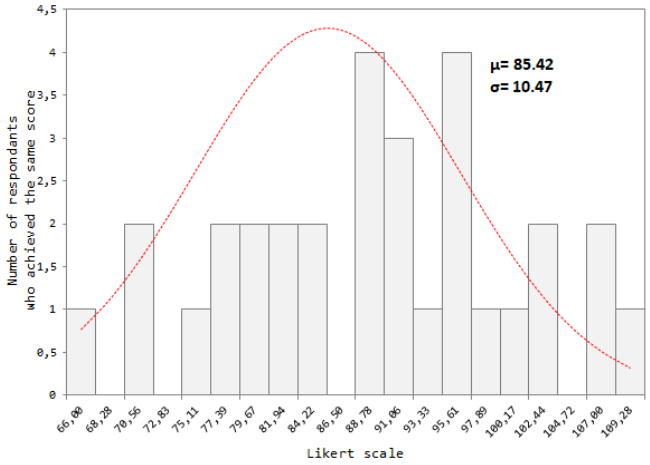

Fig. 2 Escala Likert: frecuencia de puntuaciones de las preguntas de la consulta.

Medimos la puntuación Likert por alumno teniendo en cuenta las 22 preguntas de la encuesta. Por tanto, nuestra escala Likert se situó en el rango de 22-132, con 22 muy en desacuerdo con todos los ítems de la pregunta y 132 muy de acuerdo con todos ellos. La Figura 2 muestra el histograma de puntuación, en el que cada barra contiene el número de estudiantes que obtuvieron la misma puntuación. Además, el histograma de puntuación se suavizó utilizando curvas de Bézier [29] como se muestra en la curva roja de la Figura 2. Esto muestra que los resultados también tendieron hacia una distribución normal con un promedio μ = 85,42 y una desviación estándar σ = 10,47. Luego, se aplicó la prueba de distribución normal (Figura 1), por lo que llegamos a la conclusión de que el 64,82% de los estudiantes puntuaron entre [μ- σ, μ + σ]. Así, se incluyeron 20 estudiantes en el rango de [73,95; 95,89], lo que manifiesta una percepción satisfactoria de nuestro modelo de entrenamiento basado en Scrum para la realización de los eventos Scrum.

VII. Discusión

A lo largo de este estudio, los resultados han revelado la percepción de los estudiantes sobre Scrum en términos de cómo nuestro modelo de capacitación propuesto anteriormente les ayudó a alcanzar los objetivos del evento Scrum. Como se informó anteriormente, nuestro modelo de capacitación incorpora el rol de Agile Coach para brindar tutoría y capacitación en Scrum para estudiantes de nivel de pregrado. En esta investigación, en particular, nos enfocamos en medir cómo se sintieron los estudiantes con el modelo de capacitación al realizar la Sprint Planning, Sprint, Daily Meeting (y Weekly Meeting), Sprint Review, y Sprint Retrospective.

En términos de Sprint Planning, observamos que, durante esta reunión al comienzo de cada Sprint, los estudiantes sintieron que las Historias de Usuario estaban bien aclaradas en el momento de la construcción del Product Backlog y su refinamiento a lo largo del proyecto. Además, el uso de Planning Poker en historias de usuarios refinadas (también conocidas como historias secundarias) mejoró la participación de los estudiantes. Sin embargo, hay dos cuestiones en las que debemos centrarnos para mejorar la percepción de los estudiantes sobre Scrum. En primer lugar, la evolución del Product Backlog no siempre fue clara, rastreable y visible, y en segundo lugar, se acordó satisfactoriamente el estado inicial de cada Sprint (es decir, la definición de la meta del Sprint).

En términos de Sprint, los estudiantes han adoptado la idea de que el trabajo en equipo y la comunicación entre los miembros del equipo era bueno, y la comunicación con el Product Owner fue positiva para refinar y completar el Product Backlog. Sin embargo, vale la pena señalar que la mayoría de los estudiantes sintieron que su Scrum Master no pudo desempeñar bien su papel. Para abordar este problema en el futuro, planeamos ajustar nuestro modelo de capacitación y dejar el rol de Scrum Master para que lo desempeñe un profesor [15].

En cuanto a las Weekly Meetings, observamos positivamente que las respuestas de los estudiantes en las Weekly Meetings fueron más precisas respecto a las sugerencias recibidas previamente por el Coach Ágil. Además, los estudiantes no estaban de acuerdo con la idea de que el entorno de desarrollo obstaculizaba el ciclo de vida de las historias de usuario y los recursos de capacitación desarrollados eran ineficaces. Esto podría significar que se sintieron cómodos con las herramientas y la capacitación recibidas antes del desarrollo del proyecto. Sin embargo, hay algunos problemas que deben abordarse. Por ejemplo, debemos trabajar para mejorar la percepción de las Weekly Meetings ya que deben ser cruciales para seguir y monitorear el progreso del proyecto del curso.

En términos de Sprint Review, los estudiantes consideraron positivamente los consejos del Agile Coach para planificar la implementación al final de cada Sprint. Además, las reuniones previas a la demostración de Sprint fueron útiles para preparar el producto y entrenar el discurso necesario para la presentación. No obstante, deberíamos trabajar para que esta reunión sea más fructífera para ayudar a los estudiantes a recibir comentarios del cliente y del Product Owner.

En términos de Sprint Retrospective, estas reuniones permitieron identificar mejoras para el próximo Sprint, elevar las metas para cada Sprint de manera satisfactoria y manejar el alcance de las metas de Sprint de manera satisfactoria. Sorprendentemente, los estudiantes estuvieron de acuerdo en trabajar de manera similar en su contexto profesional. Sin embargo, debemos reforzar el concepto de que exponer métricas es saludable para que los miembros del equipo reflexionen sobre lo sucedido. Además, los estudiantes sintieron que los riesgos e impedimentos no se identificaron, manejaron, resolvieron y mitigaron satisfactoriamente de manera efectiva.

Un aspecto interesante a destacar en esta investigación es el diagnóstico que realiza el Agile Coach a lo largo del proyecto del curso. Al finalizar el proyecto del curso, el Coach Agile organizó sus anotaciones cronológicamente y redactó un breve diagnóstico del equipo y proyecto durante

todo el período, y entregó estos diagnósticos y los comentarios al profesor, quien indicó que los resultados de nuestro modelo de formación le ayudaron. Determinar debilidades y oportunidades para mejorar aspectos que, de acuerdo con los comentarios de los estudiantes, podrían facilitar la ejecución de los eventos Scrum. Por ejemplo, Agile Coach señaló en las reuniones de revisión de sprints que los estudiantes tardaron demasiado en tomar el control del proyecto. Por lo tanto, el Agile Coach al comienzo del Sprint ayudó a mitigar esta limitación al motivar a los estudiantes a obtener más información del cliente para mejorar la construcción de historias de usuarios. Los resultados indican que nuestro modelo de formación no solo ayuda a los estudiantes a mejorar su experiencia de aprendizaje, sino que también ayuda a los profesores a mejorar las estrategias de enseñanza.

VIII. Conclusiones

En este artículo, nuestro objetivo fue medir cómo se sentían los estudiantes con nuestro modelo de capacitación informado anteriormente al realizar los eventos de Scrum, a saber, Sprint Planning, Sprint, Daily Meeting, Sprint Review, y Sprint Retrospective. Como profesores, nos hemos centrado tanto en mejorar el aprendizaje de la experiencia del software técnico como en retener la eficiencia del proceso de software. Descubrimos que la enseñanza del desarrollo de software Scrum es exitosa cuando los estudiantes están interesados en el desarrollo de un proyecto en lugar de las clases tradicionales. Además, la resolución de problemas de ingeniería de software en un entorno regulado proporcionó a los estudiantes las habilidades necesarias para trabajar en entornos profesionales. Descubrimos que este enfoque de enseñanza podría promover la incorporación de los estudiantes a la industria del software.

Este estudio presentó los hallazgos luego de implementar un modelo de entrenamiento basado en Scrum mejorado con coaching ágil. Los resultados prometedores mostraron que el modelo de formación ayudó a los estudiantes a mejorar sus habilidades técnicas y no técnicas. La mayoría de los estudiantes comentó que la estrategia propuesta les permitió mejorar la comprensión de Scrum.

La encuesta indicó que el curso ofrecía habilidades no técnicas útiles para los estudiantes, como fomentar la internalización de Scrum, una resolución más rápida de problemas y el estímulo a través de reuniones de control. Los hallazgos también mostraron que los estudiantes se sentían cómodos con el trabajo en equipo, la comunicación y los problemas de gestión.

Algunas limitaciones podrían comprometer la validez de nuestro estudio de caso. Primero, los resultados obtenidos podrían impactar a profesores con diferentes antecedentes y experiencia. En segundo lugar, los profesores con experiencia limitada en coaching pueden tener dificultades para implementar sin afectar la autoorganización del equipo y pueden encontrarse diciéndoles a los estudiantes el curso de acción correcto, en lugar de alentarlos a aprender de sus propios errores. En tercer lugar, vale la pena señalar que los estudiantes han abordado muchas dificultades, como la incapacidad de hacer estimaciones razonables de los requisitos de software, la falta de capacitación en pruebas de software y la incapacidad de evaluar la calidad del código fuente; por esta razón, los estudiantes con una experiencia mínima en el desarrollo de software pueden tener dificultades para comprender el enfoque de formación propuesto. Por ello, sugerimos ofrecer nuestro modelo de formación hacia el final de la carrera de Ciencias de la Computación e Ingeniería de Software.

Como trabajo futuro, planeamos replicar este curso en otros contextos educativos y profesionales de diferentes países para abordar cuestiones culturales que podrían afectar los resultados.

Agradecimientos

Queremos agradecer en primer lugar a la revista VAEP RITA (http://rita.det.uvigo.es/VAEPRITA/) y a su editor en jefe, el Dr. Martín Llamas Nistal, por el apoyo recibido para la organización de la sección especial sobre pensamiento computacional.

Como no podría ser de otra forma también nuestro agradecimiento a los autores por haber contribuido con sus trabajos a esta sección especial y, por último, a los revisores por ser los garantes de la calidad de los trabajos aceptados.

Referencias

[1] M Paasivaara, D Vodă, VT Heikkilä, J Vanhanen and C Lassenius. How does participating in a course project with industrial customers affect student attitudes?. In Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training (pp. 49-57). ACM. 2018

[2] R. Dawson. Twenty dirty tricks to train software engineers. In Proceedings of the 22nd international conference on Software engineering. ACM, 2000, pp. 209–218.

[3] G. Regev, D. C. Gause, and A. Wegmann. Experiential learning approach for requirements engineering education. Requirements engineering, vol. 14, no. 4, p. 269, 2009.

[4] T. C. Lethbridge, J. Diaz-Herrera, J. Richard Jr, J. B. Thompson. Improving software practice through education: Challenges and future trends. In Future of Software Engineering (FOSE'07). IEEE, 2007, pp. 12–28.

[5] M. H. Yamaguti, F. M. de Oliveira, C. A. Trindade, and A. Dutra. Ages: An interdisciplinary space based on projects for software engineering learning. In Proceedings of the 31st Brazilian Symposium on Software Engineering. ACM, 2017, pp. 368–373.

[6] B. Bruegge, S. Krusche, and L. Alperowitz. Software engineering project courses with industrial clients. ACM Transactions on Computing Education (TOCE), vol. 15, no. 4, p. 17, 2015.

[7] M. Marques, S. F. Ochoa, M. C. Bastarrica, and F. J. Gutierrez. Enhancing the student learning experience in software engineering project courses. IEEE Transactions on Education, vol. 61, no. 1, pp. 63–73, 2017.

[8] VersionOne. http://stateofagile.versionone.com/. 2020.

[9] J. Highsmith. Agile project management: creating innovative products. Pearson education, 2009.

[10] J. Diaz, J. Garbajosa, J. A. Calvo-Manzano. Mapping CMMI Level 2 to Scrum Practices: An experience report. Software Process Improvements, 42, 93-104, 2009, Springer.

[11] P. Maher. Weaving agile software development techniques into a traditional computer science curriculum. In Proceeding on 6th Conference on Information Technology: New Generations, 2009.

[12] K Schwaber and M Beedle. Agile software development with scrum. Upper Saddle River, NJ: Prentice Hall. 2002.

[13] L. Mathiassen, J. Pries-Heje. Business agility and diffusion of information technology. European Journal of Information Systems, 116-119. 2006

[14] S. Nerur, V. Balijepally. Theoretical reflections on agile development methodologies. Communications of the ACM, 50, 79-83. 2007

[15] V Mahnic. A course course on agile software development using scrum. IEEE Transactions on Education, 5(2), 99–106, 2012.

[16] MD Cano. Students' involvement in continuous assessment methodologies: A case study for a distributed information systems course. IEEE Transactions on Education, 54(3), 442–451. 2011

[17] P Maher (2009). Weaving agile software development techniques into a traditional computer science curriculum. Proceeding on 6th Conference on Information Technology: New Generations. Academic Press. 2009

[18] C Coupal and K Boechler (2005). Introducing agile into a software development course project. Proceedings of Agile Conference (pp. 289–297). Academic Press. 2005

[19] V Devedzic and SR Milenkovic. Teaching agile software development: A case study. IEEE Transactions on Education, 54. 2011
[20]H Kniber. The manager's role in scrum. Retrieved from http://www.scrumalliance.org/ resources/293. 2008
[21] V. Mahnic. Teaching Scrum through Team-Project work: Students' perceptions and Teachers' observations. The International Journal of Engineering Education. 2010
[22] V Mahnic. A case study on agile estimating and planning using scrum. Electronics and Electrical Engineering, 5. 2011
[23] G. Hedin, L. Bendix, B. Magnusson. Teaching extreme programming to large groups of students. Journal of Systems and Software, 74 (2), 133-146. 2005
[24] Y. Dubinsky, O. Hazzan. Extreme programming as a framework for student-project coaching in computer science course courses. In Proceedings of IEEE International Conference on Software: Science, Technology and Engineering (pp. 53-59). 2003
[25] G Rodríguez, A Soria and M Campo (2016). Measuring the Impact of Agile Coaching on Students' Performance. IEEE Transactions on Education, 59(3), 202-209. 2016
[26] R Likert. A technique for the measurement of attitudes. Archives of Psychology, 22(140). 1932
[27] M Solís Salazar. The dilemma of combining positive and negative items in scales. Psicothema, 27(2). 2015
[28] JF Hair, WC Black, BJ Babin, RE Anderson and RL Tatham (2009). Análise multivariada de dados. Bookman Editora, Porto Alegre. 2009
[29] G Rodriguez, A Soria and M Campo. Virtual Scrum: A teaching aid to introduce undergraduate software engineering students to Scrum. Computer Applications in Engineering Education, 23(1), 147-156. 2015
[30] M. Ally. Foundations of educational theory for online learning. In (Ed. Anderson, T. & Elloumi, F.) Theory and practice of online learning (pp. 3-31). Athabasca University, Canada's Open University. 2004
[31] C. Antunes. Anticipating Student's Failure as soon as possible (pp. 353-363). In C. Romero et al. (Ed.) Handbook of Educational Data Mining. CRC Press. 2010
[32] W. O. Dick, L. Carey, J.O. Carey. The systematic design of instruction. Pearson/Allyn & Bacon. 2005
[33] E. Scott, G. Rodríguez, Á. Soria, M. Campo. Towards better Scrum learning using learning styles. Journal of Systems and Software, 111, 242-253. 2016
[34] A. Savoy, R. Proctor, and G. Salvendy, Information retention from PowerPointTM and traditional lectures, Comput Educ, 52, 858–867. 2009
[35] W. Woody and C. Baker, E-books or textbooks: Students prefer textbooks, Comput Educ, 55, 945–948. 2009
[36] P. E. Green, V. R. Rao. Rating scales and information recovery—How many scales and response categories to use? Journal of Marketing, 34, 33-39. 1970.
[37] S. Y. Chyung, K. Roberts, I. Swanson, A. Hankinson. Evidence-based survey design: The use of a midpoint on the Likert scale. Performance Improvement, 56(10), 15-23. 2017
[38] T. Linden. Scrum-based learning environment: Fostering self-regulated learning. Journal of Information Systems Education, 29(2), 3. 2019

Guillermo Rodríguez es actualmente miembro del ISISTAN Research Institute (CONICET-UNICEN), Investigador Adjunto de CONICET y profesor de UNCPBA. Se graduó de Ingeniero de Sistemas en 2010 (UNCPBA), y Doctor en ciencias de la Computación en 2014 (UNCPBA). Sus principales áreas de investigación son Desarrollo de Software Orientado a Servicios y Razonamiento Basado en Casos.

Isabela Gasparini es Profesora Asociada del Departamento de Ciencias de la Computación de la Universidad del Estado de Santa Catarina - UDESC. Tiene un doctorado (2013) en la Universidad Federal de Rio Grande do Sul y una maestría (2003) en la Universidad Federal de Rio de Janeiro. Coordina el Programa de Postgrado en Enseñanza de Ciencias, Matemáticas y Tecnología (PPGECMT) y trabaja en el Programa de Postgrado en Computación Aplicada (PPGCA) y en los cursos de pregrado Licenciatura en Ciencias de la Computación (BCC) y Tecnología en Análisis y Desarrollo de Sistemas (TADS). Interesada Human-Computer Interaction y Gamification y Educación en Ciencias de la Computación.

Avanilde Kemczinski es Profesora Asociada del Departamento de Ciencias de la Computación de la Universidad del Estado de Santa Catarina - UDESC. Tiene un doctorado (2005) y una maestría (2000) en Ingeniería de Producción y Sistemas de la Universidad Federal de Santa Catarina (UFSC), una licenciatura en Terapia Ocupacional de la Asociación Docente de Santa Catarina - Facultad de Ciencias de la Salud de Joinville (1992) y un postdoctorado (2016) en Educación en la Universidad de Málaga / España. Coordina el Programa de Postgrado en Enseñanza de Ciencias, Matemáticas y Tecnología (PPGECMT) y trabaja en el Programa de Postgrado en Computación Aplicada (PPGCA) y en los cursos de pregrado Licenciatura en Ciencias de la Computación (BCC) y Tecnología en Análisis y Desarrollo de Sistemas (TADS). Es líder del Grupo de Investigación en Informática en Educación - GPIE por CNPQ / UDESC. Interesado en áreas relacionadas con la informática en la educación, en particular la tecnología educativa, objetos de aprendizaje, sistema colaborativo / aprendizaje (CSCL), interfaz hombre-ordenador, calidad en el proceso y producto de software educativo, juegos y metodologías educativos y / o modelos de enseñanza-aprendizaje.

Alexandre Veloso de Matos es Profesor Adjunto del Departamento de Ciencias de la Computación de la Universidad del Estado de Santa Catarina - UDESC. Tiene un doctorado (2020) en la Universidad Federal de Paraná y una maestría (2000) en la Univerisdad Federal de Santa Catarina. Interesado en áreas relacionadas con Desarrollo de Software, Sistemas Distribuidos y Metodologías Ágiles.