CACiC 2023
CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN

Propuesta para la Enseñanza de Desarrollo de Software Guiado por Pruebas

Nicolás Paez[1], Alejandra Zangara[2], Diego Fontevila[1], and Alejandro Oliveros[1]

[1] Departamento de Ciencia y Tecnología, Universidad Nacional de Tres de Febrero, Sáenz Peña, Argentina

[2] Facultad de Informática, Universidad Nacional de La Plata, La Plata, Argentina

{nicopaez@computer.org, azangara@info.unlp.edu.ar, dfontdevila,aoliveros}@untref.du.ar

Abstract. El Desarrollo Guiado por Pruebas es una técnica de desarrollo ágil de software muy conocida pero de poco uso en la industria. Algunos autores sugieren que esto puede ser consecuencia de una falta de capacitación o tal vez de una enseñanza muy superficial y poco efectiva. A partir de esta problemática el presente artículo presenta los avances realizados de una tesis de maestría en Tecnología Informática Aplicada en Educación cuyo objetivo es generar una propuesta para la enseñanza del Desarrollo de Software Guiado por Pruebas. Este artículo presenta la motivación del trabajo de tesis, una breve descripción de sus aportes, el grado de avance logrado hasta el momento y los trabajos futuros de cara a completar la tesis.

Keywords: Desarrollo Guiado por Pruebas, Ingeniería de Software, TDD, Educación

1. Motivación

El Desarrollo Guiado por Pruebas, habitualmente conocido como TDD (del inglés Test-Driven Development) es una técnica de desarrollo de software creada por Kent Beck durante los años 90 y formalizada en su libro Test-Driven Development by Example de 2003 [1].

Luego de la publicación del mencionado libro la técnica fue ganando cada vez más popularidad en términos de cantidad de gente que la conoce y dando origen a diversas variantes, algunas de ellas con beneficios adicionales pero todas ellas sin mayor grado de adopción. Entre estas variantes cabe destacar BDD (del inglés Behaviour-Driven Development) [2] y SBE (del inglés Specification by Example)

[3].

TDD ha logrado una gran popularidad en términos de conocimiento tanto a nivel industria como academia. Se enseña en las universidades [4] y se han realizado diversas investigaciones formales sobre sus beneficios y propiedades [5–7]. Pero curiosamente dicha popularidad no se ha traducido en adopción, sino que es en cierto modo ”solo fama”. Dicho de otro modo: mucha gente conoce la técnica, sabe de su existencia, pero son muy pocos los que la utilizan en el desarrollo de software a nivel industrial[8] . Los estudios más optimistas reportan que TDD es utilizado por menos del 30 % de los programadores. Pueden pensarse diversas explicaciones para este fenómeno, algunas de ellas podrían ser: que es una técnica de muy difícil aplicación o que resulta muy

difícil aprenderla. He aquí la motivación del trabajo de investigación descripto en el presente artículo.

Dicho trabajo pertenece a una tesis para la Maestría en Tecnología Informática Aplicada en Educación de la Facultad de Informática de la Universidad Nacional de La Plata. El mismo es llevado a cabo por el tesista Ingeniero Nicolás Paez bajo la dirección de la Dra. Alejandra Zangara, la co-dirección del Dr. Diego Fontdevila y el asesoramiento técnico del Lic. Alejandro Oliveros. El trabajo de investigación fue iniciado en el año 2021 y se espera sea completado en el transcurso de 2023.

El objetivo del trabajo es desarrollar una Propuesta para la Enseñanza del Desarrollo Guiado por Pruebas.

A continuación, se describen los puntos principales del plan de trabajo:

1. Realizar una revisión formal de bibliografía para establecer el estado del arte en lo referente a la enseñanza de TDD
2. Relevar con especialistas en la materia desafíos y recomendaciones para la enseñanza de TDD
3. Elaborar una propuesta de enseñanza de TDD
4. Implementar la propuesta en al menos dos cursos para validarla

En la siguiente sección se detalla el estado actual de cada uno de los items del plan de trabajo.

2. Aporte del trabajo

En primera instancia, acorde a lo planificado, se realizó una revisión sistemática de literatura que será publicada como un artículo independiente en el transcurso del corriente año. Esta revisión sistemática representa un primer aporte del trabajo de investigación: conocer el estado del arte en la enseñanza de TDD.

En segunda instancia se llevó a cabo un relevamiento de desafíos y recomendaciones para la enseñanza de TDD. Este segundo relevamiento se hizo a partir de entrevistas individuales semi-estructuradas con 12 expertos en la enseñanza de TDD. Estas entrevistas, o más precisamente las información extraída y sintetizada de las mismas representa un segundo aporte de la tesis: un conjunto de desafíos y recomendaciones para la enseñanza de TDD. Este aporte también está planificado como una publicación independiente.

A partir de conocer el estado del arte, los desafíos y recomendaciones de los expertos se elaboró una propuesta para la enseñanza de TDD. En la elaboración de dicha propuesta también se ha tenido en cuenta la experiencia personal del tesista en la enseñanza de TDD. La parte conceptual de la propuesta de enseñanza ya se encuentra desarrollada y se está trabajando en la generación de los materiales didácticos que permitirán su implementación. Al mismo tiempo parte de la propuesta ya se está aplicando en las materias que actualmente dicta el tesista.

En términos resumidos, la propuesta para la enseñanza de TDD está estructurada en los siguientes pilares referentes al recorrido del contenido:

- Comenzar la enseñanza de TDD como un enfoque de programación resolviendo problemas de modelado y algoritmia utilizando el estilo ”Clásico de TDD” (también

denominado estilo Chicago) [9]

- Luego en segunda instancia, trabajar sobre problemas más complejos, incluyendo la problemática de manejo de entrada/salida y utilizando el estilo ”mockist” (conocido también como estilo London) [10]
- Adicionalmente, en esta segunda instancia, introducir otras prácticas de la metodología Extreme Programming (XP) que es la que dio origen a TDD. Concretamente las prácticas a incorporar son Programación de a Pares (Pair Programming) e Integración Continua (Continuous Integration) que son dos prácticas íntimamente relacionadas a TDD [11]

Esta parte de la propuesta ya ha sido presentada en un artículo que fue aceptado para publicación en el Simposio Argentino de Educación en Informática* que se llevará a cabo en el mes de Septiembre de 2023.

Por otro lado, en lo referente a la estrategia didáctica la propuesta plantea los siguientes pilares:

- Aula invertida: esta técnica propone sacar de la clase sincrónica el contenido teórico entregándolo a los alumnos en forma de videos y/o lecturas para que sea estudiado fuera del aula. Al mismo tiempo esto permite utilizar el tiempo de clase sincrónica para realizar actividades de aplicación utilizando estrategias de educación activa como las planteadas por Bowman [12]
- Aula extendida: esta técnica propone extender el espacio de intercambio más allá del aula de clase sincrónica a partir del uso de un aula virtual y demás herramientas digitales permitiendo un intercambio más fluido y continuo entre docentes y alumnos.
- Evaluación continua: este enfoque apunta a realizar evaluaciones de forma regular a lo largo de toda la duración del curso. Como estas evaluaciones son periódicas las mismas suelen ser más acotadas que las tradicionales evaluaciones parciales y al mismo tiempo suelen estar más enfocada en temas puntuales.
- Aprendizaje basado en proyectos: esta estrategia propone que los alumnos adquieran los conocimientos y competencias mediante la realización de proyectos que los enfrenten con situaciones del mundo real [13].

El uso de las técnicas enumeradas en la enseñanza de temas de Ingeniería de Software no es una novedad. Lo que sí resulta innovador en un punto es su uso combinado en un mismo curso. Parte de la propuesta para integrar estas estrategias ya ha sido descripta en una publicación anterior del tesista [14] y en términos resumidos consiste en dividir el dictado de la materia en dos partes de la siguiente forma:

- Una primera parte centrada en que los alumnos individualmente adquieran ciertos conocimientos y habilidades que les permitan desempeñarse de forma armónica y eficiente al trabajar en grupo. Esta primera parte se dicta en una modalidad de aula invertida y extendida y realizando evaluaciones continuas.
- Una segunda parte donde los alumnos trabajan en grupo en el desarrollo de un proyecto que les permita poner en práctica los conocimientos y técnicas estudiados en la primera parte de la materia. Todos los grupos deben presentar avances semanales que posibilitan una evaluación continua del desempeño de los grupos y

* https://52jaiio.sadio.org.ar/simposios/SAEI

sus integrantes.

3. Trabajos futuros

Una vez completos los materiales didácticos para la implementación de la propuesta la misma será implementada dos cursos de Ingeniería de Software de cara a validar su utilidad y efectividad. Esos cursos son: la asignatura Ingeniería de Software perteneciente a la carrera de Ingeniería de Computación de la UNTreF y la asignatura Métodos y Modelos de Ingeniería 2 perteneciente a la carrera de Licenciatura en Análisis de Sistemas de la UBA. Estas dos implementaciones de la propuesta serán utilizadas para su validación. Respecto del mecanismo de evaluación se utilizarán dos estrategias. En primer lugar, las distintas actividades de evaluación del curso permitirán verificar el grado de conocimiento de los temas logrado por los alumnos. Pero solamente esto podría no resulta suficiente pues podría haber alumnos que ya contarán con los conocimientos desde antes de tomar el curso (esto no sería raro ya que cierto porcentaje de los alumnos llegan al curso con experiencia en el ámbito laboral donde bien podrían haber aprendido los temas en cuestión). Es por esto que adicionalmente se realizará un cuasi-experimento. Se les pedirá a los alumnos resolver dos ejercicios: uno ejercicio al comienzo del curso y otro al final del mismo. Si el alumno no tenia conocimientos sobre los temas de la materia y los adquirió durante la cursada, entonces deberían apreciarse diferencias significativas en las soluciones del primer y segundo ejercicio.

Una completado el trabajo de tesis, podrían surgir algunos otros temas de investigación como ser la factibilidad de utilizar la estrategia didáctica planteada para enseñanza de otros temas de Ingeniería de Software.

References

1. Beck, K.: Test-Driven Development by Example. Addison-Wesley (2003)
2. Rose, S.,Wynne,M.,Hellesoy,A.The Cucumber for Java Book: Behaviour-Driven Development for Testers and Developers.United States:Pragmatic Bookshelf (2015)
3. Adzic, G.: Specification by Example: How Successful Teams Deliver the Right Software. Manning Publications (2011)
4. Paez, N., Oliveros, A., Fontdevila, D. Initial Assessment of Agile Development in the Undergraduate Curricula. In: Meirelles, P., Nelson, M., Rocha, C. (eds) Agile Methods. WBMA 2019. Communications in Computer and Information Science, vol 1106. Springer, Cham. https://doi.org/10.1007/978-3-030-36701-5_6
5. Buchan, J., Li L. and MacDonell, S. G., Causal Factors, Benefits and Challengesof Test-Driven Development: Practitioner Perceptions, 2011 18th Asia-Pacific Software Engineering Conference, Ho Chi Minh City, Vietnam, 2011, pp. 405-413, doi: 10.1109/APSEC.2011.44.
6. Crispin, L.. Driving Software Quality: How Test-Driven Development Impacts Software Quality, in IEEE Software, vol. 23, no. 6, pp. 70-71, Nov.-Dec. 2006, doi: 10.1109/MS.2006.157.
7. A. Causevic, D. Sundmark and S. Punnekkat, “Test case quality in test driven development: A study design and a pilot experiment,” 16th International Conference on Evaluation &

Assessment in Software Engineering (EASE 2012), Ciudad Real, 2012, pp. 223-227, doi: 10.1049/ic.2012.0029.

8. Paez, N., Fontdevila, D., Gainey, F., Oliveros, A. (2018). Technical and Organizational Agile Practices: A Latin-American Survey. In: Garbajosa, J., Wang, X., Aguiar, A. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2018. Lecture Notes in Business Information Processing, vol 314. Springer,
9. Cham. https://doi.org/10.1007/978-3-319-91602-6_10
10. Martin, B. Comparative Case Study: London vs. Chicago, https://bit.ly/ 3W1GtRl. Last accessed 8 Ago 2023
11. Freeman, S. Pryce, N.: Growing Object-Oriented Software, Guided by Tests. Pearson Education (1999)
12. Beck,K., Andres,C.. Extreme Programming Explained: Embrace Change.UnitedKingdom:Pearson Education (2004)
13. Bowman, S. L.. Training from the back of the room!:65 ways to Step aside and letthem learn. John Wiley & Sons, (2008)
14. Pérez, B. and Rubio, A.. A Project-Based Learning Approach for Enhancing´ Learning Skills and Motivation in Software Engineering. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20). Association for Computing Machinery, New York, NY, USA, 309–315. https://doi.org/10.1145/3328778.3366891
15. Paez, N. and De la Fuente, H., Software Engineering Education Meets DevOps: an Experience Report, 2022 IEEE Biennial Congress of Argentina (ARGENCON), San Juan, Argentina, 2022, pp. 1-7, doi: 10.1109/ARGENCON55245.2022.9940102.