

Universidad Tecnológica de La Habana

“José Antonio Echeverría”



**Facultad de Ingeniería Informática**

**Diseño e implementación de pruebas automatizadas al módulo Carga y Descarga del sistema de planificación de recursos de Habana Club International.**

*Informe prácticas profesionales 1*

**Autores:** Pedro Pablo Chevalier Pérez ([pchevalier@ceis.cujae.edu.cu](mailto:pchevalier@ceis.cujae.edu.cu))

Christian Peña González ([cpena@ceis.cujae.edu.cu](mailto:cpena@ceis.cujae.edu.cu))

**Tutor:** Dr. C. Raisa Socorro Llanes (raisa@ceis.cujae.edu.cu)

La Habana, Cuba

Abril, 2025

Resumen

Abstract

Índice General

[Introducción 1](#_Toc193542447)

[Referencias Bibliográficas 4](#_Toc193542448)

# Introducción

En las últimas décadas, el software se ha consolidado como un pilar fundamental de la sociedad moderna, transformando radicalmente sectores como la salud, la educación, las finanzas y la comunicación[1]. Su influencia se extiende desde los dispositivos móviles que se utilizan diariamente hasta los complejos sistemas que controlan infraestructuras críticas, redefiniendo cómo las personas interactúan con la tecnología y entre sí mismas. Esta transformación digital ha generado un ecosistema donde el software ya no es una simple herramienta auxiliar, sino un componente esencial para el funcionamiento de la civilización actual.

La dependencia del software ha puesto de manifiesto la vital importancia de garantizar su calidad. Un error en un sistema bancario puede paralizar transacciones financieras en todo el mundo; una falla en equipos médicos podría comprometer vidas humanas; un defecto en sistemas de control industrial podría desencadenar catástrofes ambientales[2]. Las implicaciones económicas de estas fallas son igualmente alarmantes, con estimaciones que sitúan las pérdidas anuales en decenas de miles de millones de dólares[3].

Más allá del impacto financiero inmediato, el software defectuoso genera una cascada de consecuencias negativas: ciclos de desarrollo que se alargan indefinidamente por la necesidad constante de corregir errores; la erosión gradual pero implacable de la confianza de los usuarios; la frustración de los clientes traducida en abandono; y en casos extremos, disputas legales que pueden amenazar la supervivencia misma de las organizaciones[4, 5].

El ciclo de vida del desarrollo de software(SDLC por sus siglas en inglés) ha evolucionado para adaptarse a las necesidades actuales de la industria, priorizando la agilidad, la integración continua y la entrega rápida de software de calidad. En la actualidad, metodologías como DevOps, Agile y CI/CD(continuous integration and continuous delivery) han reemplazado en gran medida los enfoques tradicionales en cascada, promoviendo iteraciones frecuentes, retroalimentación constante y pruebas automatizadas en cada fase del desarrollo[6, 7]. Desde la planificación y el diseño hasta la implementación y el mantenimiento, el aseguramiento de la calidad (QA) juega un papel clave al integrarse de manera continua en el ciclo de vida[6].

El aseguramiento de la calidad del software representa un enfoque integral y proactivo que atraviesa todo el ciclo de vida del desarrollo. Lejos de ser una fase aislada que se ejecuta al final del proceso, constituye una filosofía orientada a prevenir defectos desde las etapas iniciales de conceptualización hasta el despliegue y mantenimiento. Este paradigma establece que la calidad no se puede "probar" posteriormente en un producto, sino que debe "construirse" desde su concepción[1, 2, 6].

En este contexto, las pruebas de software emergen como una disciplina fundamental cuyo objetivo principal no es demostrar la ausencia de errores, sino descubrir su presencia[2]. Esta distinción, aunque sutil, representa un cambio fundamental en el enfoque: las pruebas eficaces requieren una mentalidad orientada a la detección de problemas más que a la validación de funcionalidades ya implementadas.

Además de la estrategia de pruebas, es esencial contar con un equipo de pruebas capacitado y con experiencia. Los probadores deben tener un profundo conocimiento del dominio de la aplicación, así como habilidades técnicas para diseñar y ejecutar pruebas efectivas[2, 7]. La colaboración entre desarrolladores y probadores es crucial para asegurar que las pruebas sean integrales y que los defectos se identifiquen y solucionen rápidamente[8].

La automatización de las pruebas ha revolucionado esta disciplina, permitiendo la ejecución sistemática, consistente y repetible de casos de prueba sin intervención humana directa. No obstante, sería erróneo considerar la automatización como una panacea universal. Las pruebas manuales conservan un valor irreemplazable en aspectos como la evaluación de usabilidad, las pruebas exploratorias y aquellas situaciones donde la intuición y experiencia humanas resultan insustituibles. La estrategia óptima consiste en una combinación equilibrada de ambos enfoques, aprovechando las fortalezas de cada uno para crear un ecosistema de calidad robusto y eficiente[9, 10].

Esta transición hacia metodologías ágiles y DevOps en la industria del desarrollo de software ha evidenciado la necesidad crítica de integrar pruebas automatizadas como parte fundamental del ciclo de desarrollo. Sin embargo, múltiples organizaciones siguen operando sin incorporar estas prácticas de manera sistemática en sus proyectos, lo que genera ineficiencias operativas y riesgos potenciales para la calidad del producto final.

Este es precisamente el caso de **AVANGENIO**, una empresa de desarrollo de software que, hasta hace poco, llevaba a cabo sus proyectos sin contar con equipos dedicados a la automatización de pruebas. Los procesos de verificación y validación se realizaban predominantemente de forma manual, lo que implicaba limitaciones significativas en términos de cobertura, consistencia y capacidad de regresión. Esta situación, común en muchas empresas del sector, suele derivar de factores como la percepción de alto costo inicial de implementación, la escasez de personal especializado o la subestimación del retorno de inversión de las pruebas automatizadas.

Estas limitaciones en la eficiencia y la cobertura de las pruebas provocaron el retraso en entregas anteriores de los módulos del sistema de gestión para Habana Club International, tomándose la decisión de añadir un equipo para realizar pruebas automatizadas al nuevo módulo en desarrollo, llamado “Carga y Descarga”.

Partiendo de la **situación problemática** anteriormente descrita, es posible plantear el **problema a resolver**:

¿Cómo se puede diseñar e implementar un sistema de pruebas automatizadas para el nuevo módulo en desarrollo que sea compatible con las tecnologías utilizadas?

Para resolver la problemática planteada, se propone el siguiente **objetivo general**:

Desarrollar un sistema de pruebas automatizadas para el nuevo módulo garantizando la compatibilidad con las tecnologías utilizadas para su desarrollo.

Para cumplir con el objetivo general planteado, se definen los **objetivos específicos** siguientes:

1. Identificar los elementos fundamentales para desarrollar un sistema de pruebas automatizadas compatible con las tecnologías utilizadas en la creación del sistema de gestión de recursos.
2. Diseñar los casos de prueba para el módulo “Carga y Descarga” que garanticen la explotación de las funcionalidades
3. Implementar los casos de prueba siguiendo patrones de diseño que favorezcan la reutilización y el mantenimiento de los conjuntos de prueba.

# Referencias Bibliográficas

1. Pressman, R.S., *Software engineering: a practitioner's approach*. 2005: Palgrave macmillan.

2. Myers, G.J., C. Sandler, and T. Badgett, *The art of software testing*. 2011: John Wiley & Sons.

3. Krasner, H., *The cost of poor software quality in the US: A 2020 report.* Proc. Consortium Inf. Softw. QualityTM (CISQTM), 2021. **2**: p. 3.

4. Pargaonkar, S., *A Study on the Benefits and Limitations of Software Testing Principles and Techniques: Software Quality Engineering.* International Journal of Scientific and Research Publications (IJSRP) 13(08) (ISSN: 2250-3153), 2023.

5. Carter, D.O., *Advancing Software Quality: A Comprehensive Exploration of Code Quality Metrics, Static Analysis Tools, and Best Practices.* Journal of Science & Technology, 2024. **5**(1): p. 69-81.

6. Pargaonkar, S., *A comprehensive research analysis of software development life cycle (SDLC) agile & waterfall model advantages, disadvantages, and application suitability in software quality engineering.* International Journal of Scientific and Research Publications (IJSRP), 2023. **13**(08): p. 345-358.

7. Crispin, L. and J. Gregory, *Agile Testing: A Practical Guide for Testers and Agile Teams*. 2009: Addison-Wesley.

8. Jorgensen, P.C., *Software Testing: A Craftsman’s Approach, Fourth Edition*. 2013: Taylor & Francis.

9. Wang, J., et al., *Software testing with large language models: Survey, landscape, and vision.* IEEE Transactions on Software Engineering, 2024.

10. Pargaonkar, S., *Synergizing Requirements Engineering and Quality Assurance: A Comprehensive Exploration in Software Quality Engineering* International Journal of Science and Research (IJSR), 2022. **12**(8).