Revista Internacional de Investigación e Innovación Tecnológica

Página principal: www.riiit.com.mx

Influencia de los Tipos de Roles de Belbin en la Revisión del Modelado de Casos de Uso

Influence of Belbin's Role Type on Use Case Modeling Review

Ucán-Pech, J.P.[a]*, Aguilar-Vera, R.A.[a], Díaz-Mendoza, J.C.[a], Mex-Álvarez, D.C.[b], Hernández-Cruz, L.M.[b]

[a] Facultad de Matemáticas, Universidad Autónoma de Yucatán, Mérida, Yucatán, México.
[b] Facultad de Ingeniería, Universidad Autónoma de Campeche, San Francisco de Campeche, Campeche, México.
juan.ucan@correo.uady.mx*; avera@correo.uady.mx; julio.diaz@correo.uady.mx; diancmex@uacam.mx; lmhernan@uacam.mx

Innovación tecnológica: Estudio de la influencia de los roles de Belbin en el aprendizaje de la ingeniería de software.

Área de aplicación: En el área de educación, específicamente en actividades del aprendizaje de la ingeniería de software.

Recibido: 22 enero 2025
Aceptado: 28 junio 2025

Abstract

This report explores the influence of Belbin's role theory on fault detection in software design during the use of the Unified Modeling Language, particularly in use case diagrams. From a research perspective, in Software Engineering, not only the engineering process for building a software product is studied, but also the human behavior in this discipline is of interest to many researchers, for example, software development teams. Considering that the acquisition of professional practice skills begins during the professional career studies, to carry out this research, a controlled experiment was developed with students who were enrolled in a Software Design course. The experimental design used corresponds to a factorial design with one source of variation and three alternatives. The dependent variable of interest for this study was the numerical metric of effectiveness. The results of this experiment demonstrate that the means of the effectiveness index in detecting faults in Use Case Modeling, by the three Belbin Role Types, do not present significant differences.

Keywords: Belbin roles, Faults in use case diagrams, Software testing, learning design.

Resumen

Este reporte explora la influencia de la teoría de los roles de Belbin en la detección de faltas en el diseño de software durante el uso del Lenguaje Unificado de Modelado, particularmente en los diagramas de casos de uso. Desde un enfoque de investigación, en la Ingeniería de Software, no solo se estudia el proceso de ingeniería para la construcción de un producto de software, si no, también es de interés de muchos investigadores el comportamiento humano en esta disciplina, por ejemplo, los roles y los equipos de desarrollo de software. Considerando que la obtención de las competencias de la práctica profesional inicia durante los estudios de la carrera profesional, para realizar esta investigación, se desarrolló un experimento controlado con estudiantes que estaban matriculados en un curso Diseño de Software, el diseño experimental usado corresponde a un diseño factorial con una fuente de variación y tres alternativas. La variable dependiente de interés para este estudio fue la métrica numérica de efectividad. Los resultados de este experimento demuestran que las medias del índice de efectividad en la detección de faltas en el Modelado de Casos de Uso, por los tres Tipos de Roles Belbin, no presentan diferencias significativas.

Palabras clave: Roles de Belbin, Faltas en diagramas de casos de uso, Pruebas de software, Aprendizaje de diseño.

1. Introducción

El diseño de software, dentro del ciclo de vida de la ingeniería de software, se encuentra al comienzo de las primeras fases de esta disciplina (ver Figura 1). Dicha disciplina, aunque no es tan antigua como otras disciplinas de la ingeniería, por ejemplo, la ingeniería civil o la ingeniería industrial. La ingeniería de software en pocas décadas ha logrado avanzar con éxito en el tema del diseño de software, en tal sentido que actualmente se tiene un estándar para los ingenieros de software, este es el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés). Los autores Agner et al. (2019) confirman que UML se considera como tal para el diseño de software y que ha sido ampliamente utilizado en la educación y la industria.

Vale la pena señalar que, en el breve proceso de desarrollo de la ingeniería de software, el ciclo de vida del desarrollo del producto de software cuenta con una amplia base de conocimientos. A pesar de esto, es posible cometer faltas en el proceso de construcción como sucede con otras ingenierías. Afortunadamente, la disciplina de ingeniería de software cuenta con la fase de pruebas del sistema.

Desde la perspectiva de la dualidad de proceso y producto, como se muestra en la Figura 1, la fase de prueba se encuentra al final del modelo. Sin embargo, como se mencionó anteriormente, el modelo de proceso de software también ha evolucionado. En este marco, las diferentes etapas del modelo no se implementan en un orden predeterminado, esto depende del problema a resolver (Aguilar et al., 2017).

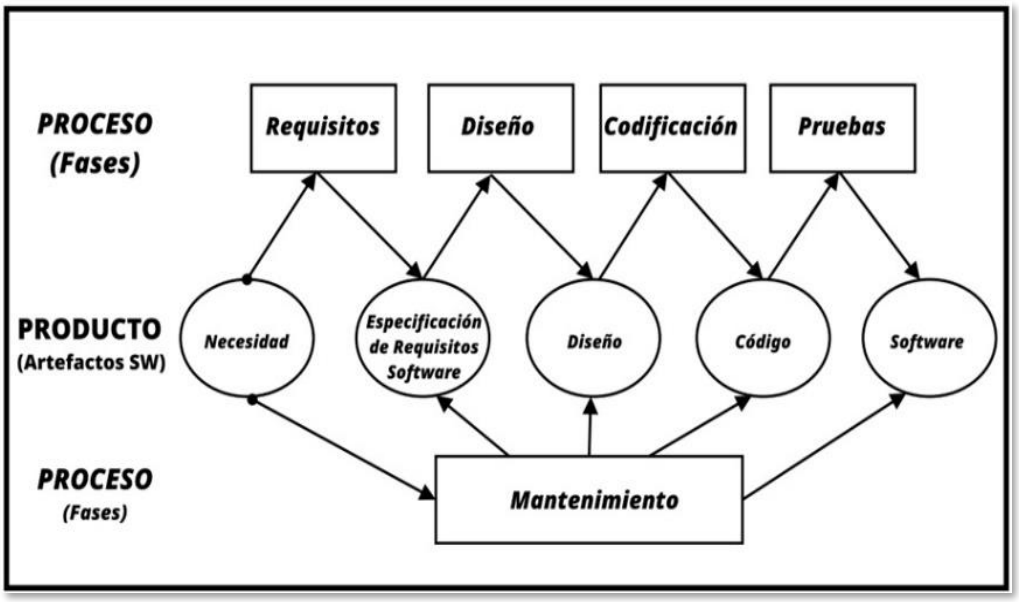


Figura 1. Dualidad Proceso-Producto en la Ingeniería de Software.
Fuente: Aguilar et al., (2017).

En este sentido, es necesario integrar la fase de pruebas desde la primera fase del ciclo de vida del producto de software para evitar faltas durante la construcción de productos de software. En la literatura, varios estudios informan sobre la detección de faltas en la fase de requisitos (Huang, 2021; Singh & Walia, 2021a; Arora et al., 2017; Singh et al., 2018; Porter & Votta, 1994; Singh & Walia, 2021b), de codificación (Dutta, 2022; Klinik et al., 2022; Watanabe et al., 2024; Vahabzadeh et al., 2015; Basili & Perricone, 1984; Ucán et al., 2023a), y el diseño no es una excepción.

Por otro lado, explorando mejoras en el proceso de software, es de interés de muchos investigadores el comportamiento humano en esta disciplina, en este sentido, también es de nuestro interés el comportamiento de los equipos de desarrollo de software. La justificación, muchas tareas en el proceso de construcción de software se realizan en equipos de trabajo.

2. Marco teórico

Roles de Belbin

Belbin (1981, 1993) sostiene que un rol de equipo se refiere a la forma de comportarse, contribuir y relacionarse con otras personas en el trabajo y aunque algunos de los roles son naturales, otros roles podrían ser adoptados por el propio individuo y algunos incluso pueden ser descubiertos después de ser adoptados. Los nueve roles propuestos por Belbin pueden agruparse en torno al tipo de conducta en tres categorías diferentes como se describe en la Tabla 1.

Tabla 1. Categorías de los roles de Belbin.

Tipo	Rol	Características
Acción	Impulsor Implementador Finalizador	Son aquellos roles que inician, desarrollan y finalizan tareas.
Mental	Cerebro Monitor-Evaluador Especialista	Son aquellos roles que cuentan con los conocimientos y habilidades requeridas para la tarea, así como una visión crítica para su realización.
Social	Coordinador Investigador de Recursos Cohesionador	Son aquellos que fomentan la comunicación y la cohesión tanto entre los miembros del equipo como con las personas con las que el equipo interactúa.

Fuente: Ucán et al. (2023a).

Detección de faltas en el modelado

Cuando evaluamos el código de programación o el modelado, las dos palabras "error" y "falta" a menudo se confunden o se usan casualmente como si fueran la misma cosa. Una situación similar ocurre con las otras dos palabras, estas son "fallo" y "defecto". Según Juristo et al. (2006), hay cuatro términos que suelen usarse indistintamente, pero representan texturas diferentes:

- Error: acción humana que produce una falta.
- Falta: algo anda mal con el producto (modelo, código, documento, etc.).
- Fallo: expresión de una falta.
- Defecto: error, falta o fallo.

En la literatura es posible encontrar una variedad de faltas y causas de faltas que pueden ser realizadas, principalmente por alumnos que apenas inician con sus primeros modelados; en este estudio se trabajó con los diagramas de casos de uso, basado en la revisión sistemática presentada por los autores Ucán-Pech et al. (2023). Cabe destacar que estos mismos autores en ese mismo reporte citado, también listan los diagramas que forman parte del UML y entre ellos se encuentran los diagramas de casos de uso. Finalmente, como resultado de la revisión sistemática previamente citada, en la Tabla 2 se presenta una propuesta de una taxonomía de diez clases de faltas que fueron usadas para esta investigación.

Tabla 2. Clases de faltas en los Diagrama de Casos de Uso.

Clase de Falta	Descripción
C1	No empezar el nombre de un caso de uso con un verbo en infinitivo.
C2	Redacción del contenido del diagrama.
C3	Se usa la simbología UML para representar los actores (muñequito), casos de uso (eclipses), tipos de relaciones (flechas) etc.
C4	Diferentes elementos para representar acciones similares.
C5	Elementos similares para representar diferentes acciones.
C6	Inadecuación en la denominación de los elementos.
C7	Uso incorrecto de generalizaciones/especializaciones.
C8	Reconocer acciones recurrentes en casos de uso – include.
C9	Reconocer acciones de suscripción entre casos de uso: extend.
C10	Reconocer cuándo reutilizar información de otros diagramas.

Fuente: elaboración propia.

Trabajos relacionados

Estrada & Peña (2013) presentan el uso de la teoría de roles de Belbin en tareas individuales en el contexto de la Ingeniería de Software. En este trabajo se reporta un experimento controlado con estudiantes que realizan actividades relacionadas con las etapas de requerimientos, diseño y codificación; los autores concluyen que algunos roles tienen mayor incidencia en ciertas actividades, destacando particularmente el rol de Implementador en la tarea de codificación.

Otro estudio relacionado con la detección de faltas en el diseño de software, en específico el modelado de casos de uso, se encuentra el trabajo de Tianual & Pohthong (2019). En este estudio se reporta un experimento de casos simples con estudiantes en dicho experimento; los autores presentan una investigación preliminar con el fin de comparar entre la detección manual de faltas y un sistema automatizado propuesto para la detección faltas en las vistas de casos de uso; los resultados que obtuvieron, muestran que la eficiencia de la detección manual de faltas es ligeramente menor que la técnica propuesta.

Un estudio más reciente presentado por Ucán et al. (2023b), explora la influencia de la Teoría de Belbin en la tarea de pruebas de software, particularmente la detección de faltas en el código. En dicho estudio se analizó la eficiencia y la confusión en la detección de faltas. Entre los resultados del experimento, no muestran diferencias significativas en el índice de eficiencia obtenido por los sujetos en el proceso de pruebas, así como tampoco con el índice de confusión.

3. Materiales y métodos

La presente investigación empírica para cumplir con su propósito adopta un enfoque cuantitativo, experimental, donde el objetivo de este experimento controlado es explorar si alguno de los tipos de roles propuestos por Belbin (Acción, Mental o Social) presenta un desempeño en el proceso de Revisión de Modelado de Casos de Uso que sea estadísticamente diferente al de los otros dos. Se ha considerado particularmente un indicador de desempeño, el índice de efectividad y se define como el porcentaje de faltas correctamente detectados por el sujeto experimental.

El experimento controlado se llevó a cabo con estudiantes que completaron su primer curso de diseño de software. De acuerdo con la clasificación de experiencia en programación creada por Dreyfus & Dreyfus (1986), los sujetos experimentales pueden clasificarse como estudiantes principiantes.

Factor y alternativas

De acuerdo con la Tabla 1, Belbin clasificó los nueve roles de equipo en tres tipos de roles: Roles de Acción (RA), Roles Mentales (RM) y Roles Sociales (RS); por lo que en este análisis experimental se considera el tipo de rol como un factor y los tres tipos como alternativas: RA, RM y RS.

Hipótesis y variables

La hipótesis estadística utiliza como variable dependiente, una métrica vinculada a la calidad del proceso individual, la efectividad (índice de efectividad) en la identificación de faltas en el Modelado de Casos de Uso.

- H_{01}: Las medias del índice de efectividad en la detección de faltas en el Modelado de Casos de Uso, por los tres Tipos de Roles Belbin, no presentan diferencias.
- H_{11}: Las medias del índice de efectividad en la detección de faltas en el Modelado de Casos de Uso, por los Tipos de Roles Belbin, difieren en al menos un par de estas.

Unidad experimental

Una unidad experimental, también llamado objeto experimental, es un artefacto o muestra que se utiliza para obtener valores que representan los resultados de un experimento. En nuestro estudio, dado que el propósito de la actividad es detectar faltas en el modelado de casos de uso, el objeto experimental es el modelado de casos de uso revisado por los sujetos durante el experimento.

En la Figura 2 se muestra el modelado del diagrama de casos de uso que los sujetos utilizaron en el experimento.

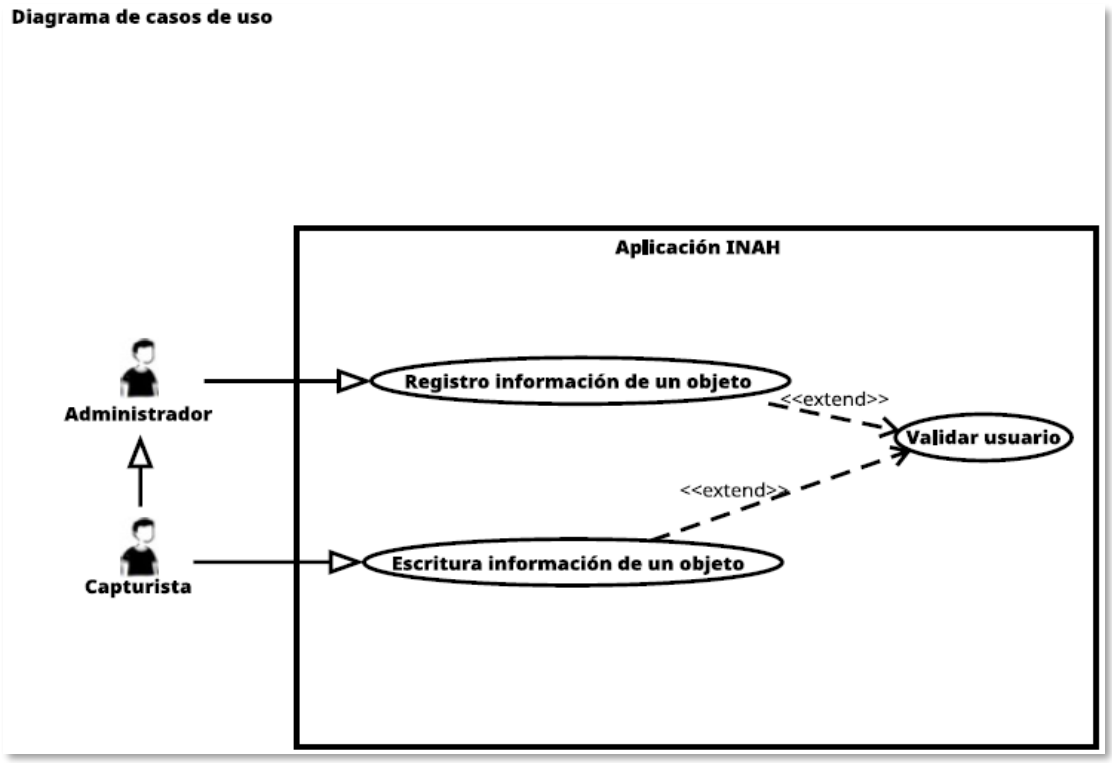


Figura 2. Diagrama de Casos de Uso. *Fuente: elaboración propia.*

De la taxonomía de faltas propuesta en la Tabla 1, al ser este un primer experimento con los diagramas de casos de uso, se decidió por conveniencia experimentar con 5 clases o tipos de faltas, en este sentido en el diagrama de casos uso ilustrado en la Figura 2, se le inyectaron 10 faltas donde 2 fueron de tipo C1, 1 de tipo C2, 4 de tipo C3, 1 de tipo C7 y 2 de tipo C9.

Diseño experimental

En la Tabla 3 se puede observar el diseño experimental usado para este estudio, corresponde a un diseño factorial con una fuente de variación y tres alternativas. La variable dependiente de interés para este estudio es la métrica numérica de efectividad.

Tabla 3. Diseño factorial con una fuente de variación y tres alternativas.

Factor	Alternativas	Variable Dependiente
Tipo de Rol	Acción (Tipo 1)	Efectividad
	Social (Tipo 2)	
	Mental (Tipo 3)	

Fuente: elaboración propia.

4. Resultados y métodos experimentales

El experimento se realizó en las instalaciones de la Facultad de Matemáticas de la Universidad Autónoma de Yucatán en el

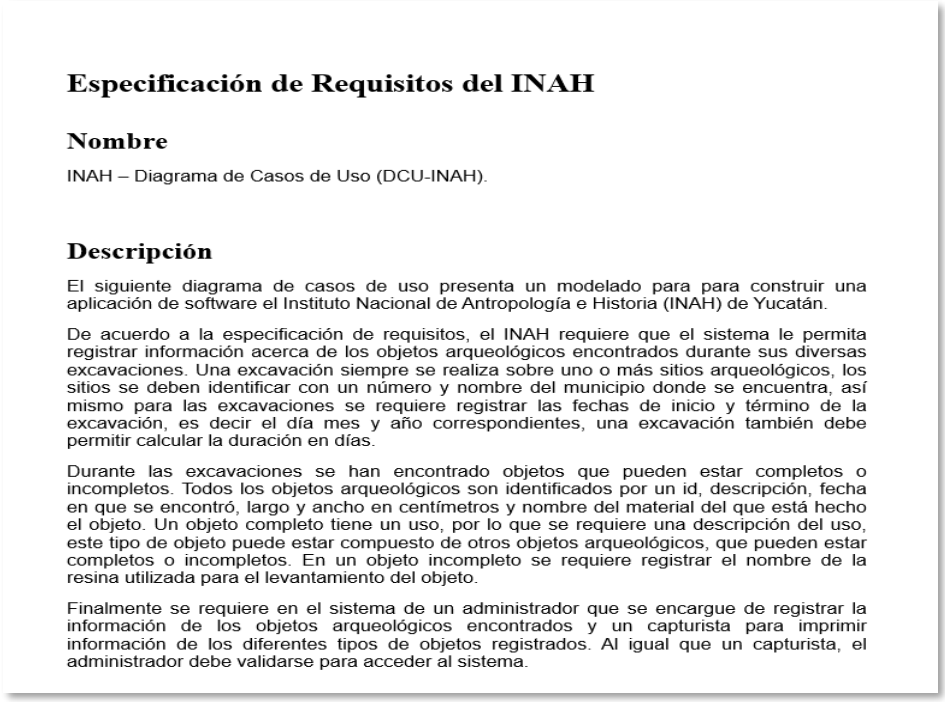
periodo enero-mayo de 2024. Los participantes fueron estudiantes de la Licenciatura en Ingeniería de Software que estaban inscritos a uno de los cursos de la asignatura de Diseño de Software que se impartía en el semestre correspondiente al periodo mencionado. En el transcurso del curso, uno de los autores, en su calidad de docente, enseñó y estudió con el grupo temas vinculados al modelado de software, entre ellos el Lenguaje de Modelado Unificado, en particular, los Diagramas de Casos de Uso. Como se mencionó en la sección del marco teórico, para este estudio se utilizó los tipos de faltas en los diagramas de casos de uso listados en la Tabla 2.

Sujetos experimentales

La muestra utilizada en el experimento por conveniencia estuvo compuesta por 35 estudiantes matriculados en el curso Diseño de Software, estos participantes ya contaban con los conocimientos esenciales del Lenguaje de Modelado Unificado, específicamente en Modelado de Casos de Uso. Para identificar el tipo de rol de cada sujeto, al final de una de las sesiones de clase durante el curso, se aplicó una prueba de autopercepción de Belbin a todos los estudiantes matriculados en el curso. Con ello, los autores identificaron el rol asumido —y por tanto el tipo de rol— para cada uno de los posibles sujetos. Finalmente, se depuró la lista de estudiantes que se ofrecieron como voluntarios para el estudio y 31 estudiantes participaron en el experimento. Genero et al. (2014) argumentaron que una muestra basada en estudiantes permite al investigador obtener evidencia preliminar para confirmar o refutar hipótesis que luego pueden ser puestas a prueba en un contexto industrial.

Sesión experimental

La sesión experimental se realizó en dos momentos, esto fue durante las dos últimas semanas del curso. El primer momento consistió en una sesión de entrenamiento, durante esta sesión a los sujetos experimentales se les capacitó con la detección de faltas en el modelado de casos de uso, para ello se utilizó los tipos de faltas propuestas en la sección: Detección de faltas en el modelado. El segundo momento se realizó en la semana siguiente, durante esta sesión se realizó el experimento previamente planeado, al inicio, cada sujeto experimental recibió la especificación de requisitos de un problema (ver Figura 3), un instrumento de recolección de datos, y el diagrama de casos de uso con las faltas inyectadas presentado en la Figura 2.



Especificación de Requisitos del INAH

Nombre

INAH – Diagrama de Casos de Uso (DCU-INAH).

Descripción

El siguiente diagrama de casos de uso presenta un modelado para para construir una aplicación de software el Instituto Nacional de Antropología e Historia (INAH) de Yucatán.

De acuerdo a la especificación de requisitos, el INAH requiere que el sistema le permita registrar información acerca de los objetos arqueológicos encontrados durante sus diversas excavaciones. Una excavación siempre se realiza sobre uno o más sitios arqueológicos, los sitios se deben identificar con un número y nombre del municipio donde se encuentra, así mismo para las excavaciones se requiere registrar las fechas de inicio y término de la excavación, es decir el día mes y año correspondientes, una excavación también debe permitir calcular la duración en días.

Durante las excavaciones se han encontrado objetos que pueden estar completos o incompletos. Todos los objetos arqueológicos son identificados por un id, descripción, fecha en que se encontró, largo y ancho en centímetros y nombre del material del que está hecho el objeto. Un objeto completo tiene un uso, por lo que se requiere una descripción del uso, este tipo de objeto puede estar compuesto de otros objetos arqueológicos, que pueden estar completos o incompletos. En un objeto incompleto se requiere registrar el nombre de la resina utilizada para el levantamiento del objeto.

Finalmente se requiere en el sistema de un administrador que se encargue de registrar la información de los objetos arqueológicos encontrados y un capturista para imprimir información de los diferentes tipos de objetos registrados. Al igual que un capturista, el administrador debe validarse para acceder al sistema.

Figura 3. Especificación de requisitos.
Fuente: elaboración propia.

En cuanto al instrumento de recolección de datos, este consiste en un documento con campos para registrar información general de la sesión y del evaluador, entre los datos recolectados incluye campos para la fecha, el nombre del alumno, la hora de inicio, de finalización, y el total de faltas encontradas. El mismo documento también incluye una tabla con tres columnas: la primera para registrar el número de la falta; una segunda columna registrar Actor, Caso de Uso o Relación entre ellos; y una tercera columna para describir detalladamente las inconsistencias o faltas encontradas.

Análisis del experimento

Después de haber presentado el experimento previamente descrito, ahora en esta sección se presenta el análisis estadístico descriptivo e inferencial de los datos recolectados.

Estadística descriptiva

La Tabla 4 muestra los resultados obtenidos al evaluar la tendencia central y la variabilidad de los datos de investigación correspondientes al índice de efectividad. Este índice corresponde al porcentaje de detección de faltas en un diagrama de casos de uso por los sujetos organizados en los tres tipos de roles de Belbin. Se puede observar que la media es más alta con el rol de tipo 1 mientras que la mediana es más alta con el rol de tipo 3.

Tabla 4. Resumen estadístico para el índice de efectividad.

Tipo	Número	Media	Mediana	Desviación Estándar
1	12	0.641666	0.65	0.156427
2	16	0.55625	0.55	0.193110
3	3	0.633333	0.7	0.115470
Total	31	0.596774	0.6	0.174133

Fuente: elaboración propia.

Para comparar estas tres alternativas se puede utilizar una gráfica de caja y bigotes como se muestra en la Figura 4. En la figura mencionada, se puede observar que con los tipos 1 y 3 presentan menos dispersión, sin embargo, esta es apenas ligeramente menor,

en este sentido se puede sugerir que no hay una diferencia significativa con los tres tipos de roles.

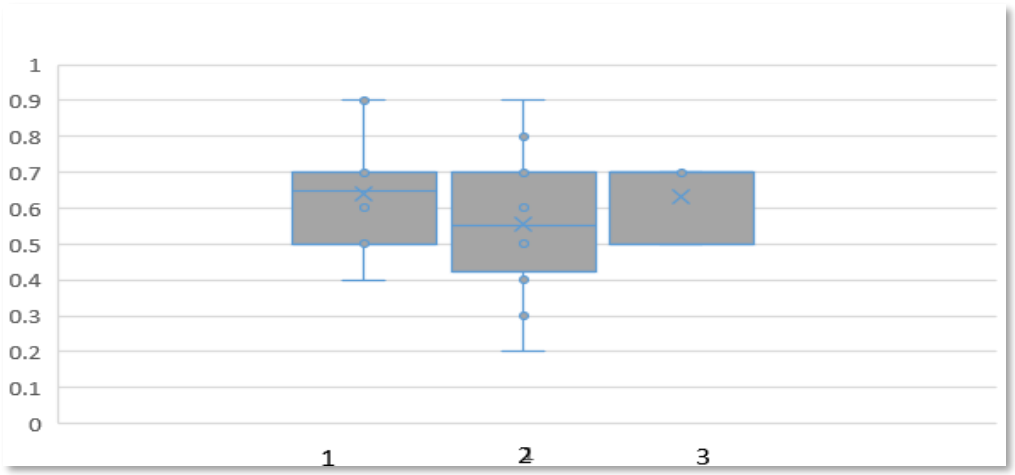


Figura 4. Gráfica de caja y bigote para la efectividad.
Fuente: elaboración propia.

Estadística inferencial

Continuando con la evaluación estadística relacionada con las diferencias de la efectividad entre los tres tipos de roles, ahora se presenta los resultados obtenidos al aplicar el análisis de varianza (ANOVA). En la Tabla 5 se puede observar que el valor obtenido para la variable p en mayor que 0.05, esto indica que la hipótesis nula es aceptada.

Tabla 5. ANOVA para la variable de efectividad.

Métrica	Estadístico F	Valor-p
Total	0.8917	0.4213

Fuente: elaboración propia.

Para verificar si el resultado obtenido con el análisis de varianza es verdadero, es necesario realizar validaciones del modelo, los resultados de estas validaciones se presentan en la siguiente sección.

Validación de los supuestos del modelo

Para validar el primer supuesto, se utilizó la prueba de Shapiro Wilk para evaluar si la efectividad en este estudio tiene o no una distribución normal. En la Tabla 6 se puede observar que el valor de la variable p es mayor que 0,05, por lo tanto, es posible asumir que la muestra tiene una distribución normal.

Tabla 6. Prueba de Shapiro Wilk para la variable de efectividad.

Métrica	Prueba	Valor-p
Efectividad	0.9404	0.08442

Fuente: elaboración propia.

Para validar el segundo supuesto de homocedasticidad, es decir, diferencias significativas entre las varianzas de los datos, se utilizó la prueba de Levene. En la Tabla 7 se puede observar que el valor de la variable p es mayor que 0.05, esto indica que no hay diferencias significativas entre las varianzas de los tres tipos de alternativas con un nivel de confianza del 95%.

Tabla 7. Prueba de Levene para la variable de efectividad.

Métrica	Prueba	Valor-p
Efectividad	1.1272	0.3382

Fuente: elaboración propia.

Finalmente, para validar el supuesto de independencia de los datos se utilizó la prueba de Durbin-Watson, esta prueba permite identificar si no existe relación en al menos la secuencia temporal de los datos. En la Tabla 8 se puede observar que el valor obtenido con esta prueba es por arriba del límite superior, esto indica que no hay correlación entre los residuos. Es decir, se confirma el supuesto de independencia de los datos.

Tabla 8. Prueba de Durbin-Watson para la variable de efectividad.

Métrica	dL	dU	DW
Efectividad	1.15	1.27	2.198782

Fuente: elaboración propia.

5. Discusiones y conclusiones

En esta investigación se exploró la influencia de la teoría de los roles de Belbin en la detección de faltas en el diseño de software, particularmente en los diagramas de casos de uso. Para este estudio se realizó un experimento controlado en el que participaron como sujetos estudiantes de Ingeniería de Software que estaban por finalizar su curso de Diseño de Software. Para el análisis de los resultados se utilizó una estadística descriptiva para evaluar la tendencia central y la variabilidad de los datos de investigación correspondientes al índice de efectividad. Como se ha mencionado, este índice corresponde al porcentaje de detección de faltas en un diagrama de casos de uso por los sujetos organizados en los tres tipos de roles de Belbin. En los resultados del análisis estadístico descriptivo se observó que el rol de tipo 1 (Acción) tuvo la media más alta, mientras que el rol de tipo 3 (Mental) presentó la mediana más alta. El rol de tipo 2 (Social) tuvo una media y mediana ligeramente más bajas. En cuanto a la dispersión, la desviación estándar fue menor para los roles de tipo 3 y tipo 1, en comparación con el rol de tipo 2. La gráfica de caja y bigotes presentada en la Figura 4, también sugiere que los tipos 1 y 3 presentan una dispersión ligeramente menor, aunque esta diferencia no es marcadamente significativa.

Así mismo, con una estadística inferencial, en específico un análisis de varianza, se obtuvo una p mayor que 0.05. Con este resultado se concluye que las medias del índice de efectividad en la detección de faltas en el Modelado de Casos de Uso, por los tres Tipos de Roles Belbin, no presentan diferencias significativas. Finalmente, se corrobaron los supuestos del modelo para verificar que el resultado obtenido con el análisis de varianza es verdadero. Estos hallazgos son consistentes con estudios previos que también exploraron la influencia de los roles de Belbin en tareas de ingeniería de software. Por ejemplo, Ucán et al. (2023b) encontraron que no existían diferencias significativas en la eficiencia y confusión en la detección de faltas en el código por parte de sujetos con diferentes tipos de roles de Belbin. Aunque Estrada & Peña (2013) señalaron que algunos roles pueden tener mayor incidencia en ciertas actividades de desarrollo de software, como el rol de Implementador en la codificación; este estudio actual, centrado en la detección de faltas en el modelado de casos de uso, no se encontró un impacto diferencial significativo entre los tipos de roles de Belbin.

A pesar de las diferencias observadas en las medias y medianas descriptivas, la inferencia estadística sugiere que los tipos de roles de Belbin (Acción, Mental y Social) no influyen de manera significativa en la efectividad de la detección de faltas en los diagramas de casos de uso dentro del contexto de este experimento controlado. Esto podría deberse a varios factores, incluyendo la naturaleza específica de la tarea de detección de faltas en diagramas de casos de uso, donde las habilidades técnicas y la comprensión de las reglas de modelado pueden tener un peso mayor que las características de rol de equipo

definidas por Belbin. Además, los sujetos experimentales, al ser estudiantes principiantes en el curso de Diseño de Software, podrían no haber desarrollado completamente las características asociadas a sus roles de Belbin o las diferencias en estas características podrían no ser lo suficientemente pronunciadas como para manifestar un impacto estadísticamente significativo en la tarea realizada.

Como trabajos futuros se ha planeado continuar explorando la influencia de la teoría de roles Belbin en el modelado de los otros diagramas del UML, en específico con los diagramas de secuencia, de colaboración, de estados y de actividades entre otros; todo esto con el propósito de seguir contribuyendo al cuerpo de conocimientos sobre factores humanos en la enseñanza de la ingeniería de software. Como reflexión final de los autores, ante los resultados obtenidos, en los trabajos futuros se tiene considerado reforzar el aprendizaje en los estudiantes con las clases o tipos de faltas en los diagramas de estudio, así como también explorar con los otros tipos de faltas del mismo; finalmente, también realizar ajustes con los tamaños de las muestras en los experimentos posteriores.

6. Agradecimientos

A los alumnos participantes en este proyecto que, estaban inscritos en la asignatura Diseño de Software durante el periodo enero-mayo de 2024 en la Facultad de Matemáticas de la Universidad Autónoma de Yucatán.

Referencias

(1) Agner, L. T., Lethbridge, T. C., & Soares, I. W. (2019). Student experience with software modeling tools. *Software and System Modeling*, 18(5), 3025–3047. https://doi.org/10.1007/s10270-018-00709-6

(2) Aguilar, R., Oktaba, H., Ramírez, R., Aguilar, J., Fernández, C., Rodríguez, O., & Ucán, J. (2017). Ingeniería de Software. In: Pineda, L. (ed.) *Computación en México por Especialidades Académicas*, pp.1167-194. Academia Mexicana de Computación.

(3) Huang, F. (2021). Software Requirement Criteria based on Human Errors. *In: 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 77-82. Wuhan, China. https://doi.org/10.1109/ISSREW53611.2021.00047

(4) Singh, M., & Walia, G. S. (2021). Automated mapping of fault logs to SRS requirements using key-phrase extraction. In: *Proceedings of the 2021 ACM Southeast* Conference (ACM SE '21). Association for Computing Machinery, pp. 138–145. New York, NY, USA. https://doi.org/10.1145/3409334.3452043

(5) Arora, C., Sabetzadeh, M., Briand, L., & Zimmer, F. (2017). Automated Extraction and Clustering of Requirements Glossary Terms. In: *IEEE Transactions on Software Engineering*, vol. 43, no. 10, pp. 918-945. https://doi.org/10.1109/TSE.2016.2635134

(6) Singh, M., Anu, V., Walia, G.S., & Goswami, A. (2018). Validating Requirements Reviews by Introducing Fault-type Level Granularity: A Machine Learning Approach. In: *Proceedings of the 11th Innovations in Software Engineering Conference* (ISEC '18). Association for Computing Machinery, Article 10, pp. 1-11. Hyderabad, India.

https://doi.org/10.1145/3172871.3172880

(7) Porter, A. A., & Votta, L. G. (1994). An experiment to assess different defect detection methods for software requirements inspections. In: *Proceedings of the 16th international conference on Software engineering (ICSE '94)*. IEEE Computer Society Press, pp.103–112. Washington, DC, USA. https://doi.org/10.1109/ICSE.1994.296770

(8) Singh, M., & Walia, G. S. (2021). Automating Key Phrase Extraction from Fault Logs to Support Post-inspection Repair of Software Requirements. In: *Proceedings of ACM Innovations in Software Engineering Conference (ISEC'21)*. Association for Computing Machinery, Article 3, pp. 1–12. New York, NY, USA. https://doi.org/10.1145/3452383.3452386

(9) Dutta, S. (2022). Localizing Faults Using Verification Technique. In: *Proceedings of the 15th Innovations in Software Engineering Conference (ISEC '22)*. Association for Computing Machinery, Article 17, pp. 1–11. New York, NY, USA.https://doi.org/10.1145/3511430.3511445

(11) Klinik, M. Koopman, P., & Wal, R. (2022). Personal Prof: Automatic Code Review for Java Assignments. In: *Proceedings of the 10th Computer Science Education Research Conference (CSERC '21)*. Association for Computing Machinery, pp. 31–38. New York, NY, USA. https://doi.org/10.1145/3507923.3507930

(11) Watanabe, M., Kashiwa, Y., Lin, B., Hirao, T., Yamaguchi, K., & Iida, H. (2024). On the Use of ChatGPT for Code Review: Do Developers Like Reviews By ChatGPT? In*: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE '24)*. Association for Computing Machinery, pp. 375–380. New York, NY, USA. https://doi.org/10.1145/3661167.3661183

(12) Vahabzadeh, Fard, A. M., & Mesbah, A. (2015). An empirical study of bugs in test code, In: *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 101-110. Bremen, Germany. https://doi.org/10.1109/ICSM.2015.7332456

(13) Basili, V., & Perricone, B. (1984). Software errors and complexity: an empirical investigation. *Communications of the ACM*, 27(1), 42-52. https://doi.org/10.1145/69605.2085

(14) Ucán, J. P., Vera, R. A., Mendoza, J. C., & Aguileta, A. A. (2023). Software Testing Using the Code Review Technique: an Exploratory Study. ReCIBE, *Revista electrónica de Computación, Informática, Biomédica y Electrónica*, 12(2), C1-16.

(15) M. Belbin. (1981). *Management teams. Why they succeed or fail*. New York, USA: John Wiley & Sons.

(16) M. Belbin. (1993). *Team roles at Work*. Oxford: Elsevier Butterworth Heinemann.

(17) Juristo, N., Moreno, A. M., & Vegas, S. (2006). *Técnicas de evaluación de*

software. Universidad Politécnica de Madrid.

(18) Ucán-Pech, J. P., Aguilar-Vera, R. A., Díaz-Mendoza, J. C., & Gómez-Gómez, O. S. (2023). Faltas en el aprendizaje del modelado de clases y casos de uso: una revisión sistemática. *Revista Científica*, 46(1), 93-106. https://doi.org/10.14483/23448350.19655

(19) E. Estrada & A. Peña. (2013). Influencia de los roles de equipo en las actividades del desarrollador de software, *Revista Electrónica de Computación, Informática, Biomédica y Electrónica*, Vol. 2(1), pp. 1-19.

(20) Tianual, P., & Pohthong, A. (2019). Defects Detection Technique of Use Case Views during Requirements Engineering. In: *Proceedings of the 2019 8th International Conference on Software and Computer Applications (ICSCA '19)*. Association for Computing Machinery, pp. 277–281. New York, NY, USA. https://doi.org/10.1145/3316615.3316631

(21) Ucán, J., Aguilar, R., Díaz, J., & Aguileta, A. (2023). Exploring the Influence of Belbin Role Types on Code Fault Detection, *Mexican International Conference on Computer Science (ENC)*, Guanajuato, Guanajuato, Mexico, 2023, pp. 1-6, https://doi.org/10.1109/ENC60556.2023.10508693

(22) Dreyfus, H L and Dreyfus, S E. (1986). *Mind over Machine: the power of human intuition and expertise in the age of the computer*, Oxford, Basil Blackwell.