Asociación Colombiana de Facultades de ingeniería

Revista Educación en Ingeniería
http://www.educacioneningenieria.org

Explorando la influencia de los roles de Belbin en la calidad del código generado por estudiantes en un curso de ingeniería de software

Antonio A. Aguileta-Güemez, Juan P. Ucán-Pech & Raúl A. Aguilar-Vera *

*Facultad de Matemáticas, Universidad Autónoma de Yucatán, Mérida, México. aaguilet@correo.uady.mx, juan.ucan@correo.uady.mx, avera@correo.uady.mx**
**Autor para correspondencia*

Resumen— El artículo presenta un experimento controlado en el que se explora la bondad de utilizar la Teoría de Roles de Belbin para la integración de equipos de desarrollo software. El estudio se desarrolla en un entorno académico con estudiantes de la carrera de Ingeniería de Software y compara la calidad de la legibilidad del código generado por equipos integrados con roles Compatibles —de acuerdo con la Teoría de Belbin— y equipos Tradicionales, en nuestro caso, equipos integrados con estudiantes seleccionados de manera aleatoria. Los resultados obtenidos aportan evidencia positiva en torno al uso de dicha teoría y motivan a los investigadores a continuar estudios en otras actividades vinculadas al proceso de desarrollo software; por otro lado, desde la perspectiva pedagógica, las resultados obtenidos con el experimento, permiten proponer como alternativa para integración de equipos de trabajo, en escenarios de aprendizaje para la Ingeniería de Software, la Teoría de Roles de Belbin.

Palabras Clave— Calidad del Código, Equipos de Desarrollo, Ingeniería de Software, Roles de Belbin.

Recibido: 15 de octubre de 2016. Revisado: 1 de noviembre de 2016. Aceptado: 28 de noviembre de 2016.

Exploring the influence of Belbin roles in the quality code generated by students in a course on software engineering

Abstract— This paper presents a controlled experiment in which the goodness of using Theory Belbin roles for the integration of software development teams is explored. The study takes place in an academic environment with students from the Engineering of Software and compares the quality of the readability of the code generated by integrated teams with roles Compatible -according to the Theory of Belbin- and traditional teams, in our case, integrated teams with students selected randomly. The results provide positive evidence on the use of this theory and motivate researchers to continue studies in other activities related to the software development process; on the other hand, from a pedagogical perspective, the results obtained with the experiment, allow proposing as an alternative to integration teams, learning scenarios related courses Software Engineering, Theory of Roles of Belbin.

Keywords— Belbin Roles, Quality Code, Software Development Teams, Software Engineering.

1. Introducción

La Ingeniería de Software (IS) surgió como respuesta a una serie de problemáticas presentadas en los procesos de desarrollo software a finales de la década de los sesenta, y a pesar de ser una disciplina joven, dispone de un cuerpo de conocimientos que es aceptado por profesionistas e investigadores de la disciplina [1]; dicho cuerpo de conocimientos integra un conjunto de áreas vinculadas con los procesos de gestión —Calidad, Configuración, Proceso Software— así como con los procesos de desarrollo —Requisitos, Diseño, Construcción, Pruebas y Mantenimiento. En el contexto de la Educación en Ingeniería de Software, la estructura curricular de un Plan de Estudios en el nivel licenciatura, suele incluir cursos obligatorios en los que se promueve el desarrollo de competencias vinculadas con las áreas de conocimiento antes citadas.

La mejora de los procesos de desarrollo software ha sido analizada desde varias aristas, una de las que ha tenido singular interés en los últimos años, estudia la importancia del factor humano al interior del equipo de trabajo; algunos autores han concluido que los principales problemas o causas del fracaso de proyectos desarrollados por equipos, no son de índole tecnológica, sino más bien se deben a factores de naturaleza sociológica [2]. Humphrey en [3] comenta que el proceso de formar y construir un equipo de desarrollo no sucede por accidente, el equipo necesita establecer relaciones de trabajo, acordar objetivos y determinar roles para los miembros del grupo; en este sentido, Belbin [4] establece una clara diferencia entre el rol que hace referencia a las habilidades técnicas y conocimientos funcionales que un individuo aporta a la organización —conocido como rol funcional— y el rol de equipo, que hace referencia a la forma de comportarse, contribuir y relacionarse con otras personas en el trabajo.

El estudio exploratorio que se describe en el presente artículo, se circunscribe en la implementación de un curso sobre Construcción Software impartido, en el período agosto-diciembre de 2015; tiene como propósito explorar si el uso de la teoría de roles de equipo propuesto por Belbin puede ser de utilidad en la formación de equipos efectivos, equipos que generen artefactos de mayor calidad —en cuanto a la legibilidad del código— que aquellos generados por equipos formados sin alguna estructura en particular —agrupados aleatoriamente.

Como citar este artículo: Aguileta-Güemez, A.A., Ucán-Pech, J.P. & Aguilar-Vera, R.A., Explorando la influencia de los roles de Belbin en la calidad del código generado por estudiantes en un curso de ingeniería de software. Revista Educación en Ingeniería 12 (23) 93-100, Febrero, 2017.

© Los autores; licenciado a ACOFI.

2. Trabajos relacionados

Diversos investigadores [5-8] afirman haber identificado conjuntos de roles que definen el comportamiento de los individuos en los equipos de trabajo; dichos roles son conocidos bajo el nombre de roles de equipo, y aunque estos roles no se encuentran asociados a algún trabajo o a las habilidades de la tarea, su ausencia o presencia se dice que tiene influencia significativa en el trabajo y en los logros del equipo [9]. De entre todas las propuestas sobre roles de equipo, el trabajo de Belbin [4,8] es probablemente el más utilizado entre consultores y entrenadores, su popularidad radica en que no solamente ofrece una categorización de roles, sino que describe una serie de recomendaciones para la conformación de equipos —resultado de sus investigaciones— que pueden ser consideradas una especie de teoría, conocida también como la Teoria de Roles de Belbin.

Johansen [10] en su disertación sobre gestión de recursos humanos, se propuso validar el inventario de autopercepción propuesto por Belbin a través de la observación del comportamiento de equipos de trabajo; entre sus conclusiones, indicó que vale la pena utilizar dicho instrumento como herramienta para evaluar la composición y posible desempeño del equipo.

Pollock [11] explora si la diversidad de roles y de personalidades entre los integrantes de un equipo —con estudiantes de sistemas de información— puede mejorar la efectividad del mismo; concluye que la diversidad no presenta influencia significativa en la efectividad del equipo, sin embargo, comenta que la presencia de ciertos roles como el Impulsor, Coordinador y Finalizador, pueden incrementar la efectividad. En este mismo sentido, Henry & Stevens [12] reportaron un experimento controlado con estudiantes en el que exploran la mejora en la eficacia de equipos de desarrollo software, en función del conjunto de roles propuestos por Belbin; dichos autores analizaron la utilidad general de los roles de Belbin —en particular los roles que pueden desempeñar el papel de líder— en términos de dos aspectos: desempeño y viabilidad; concluyen, que los equipos que contienen solo un rol de líder presentan mejor desempeño que aquellos que no lo incluyen o que incluyen más de uno.

Richard [13] condujo un experimento con estudiantes con el objetivo de determinar el grado en que ciertas dinámicas de grupo impactan sobre la eficacia de los equipos de desarrollo de software; entre sus conclusiones se encuentra el que los roles de Belbin no parecen predecir el éxito y el fracaso del grupo de desarrollo, no obstante, comenta que identificó una pequeña mejora en las notas de los alumnos obtenidos en sus proyectos.

Guzmán y Peña [14] condujeron un experimento controlado con alumnos en el cual observaron el desempeño de los roles de equipo en tareas vinculadas con el proceso de desarrollo software, como parte de sus conclusiones reportaron que algunos roles presentan mayor aportación con ciertas actividades, como fue el caso del rol Implementador en la tarea de codificación.

Otros estudios han incorporado la perspectiva de las persona y sus capacidades humanas para la adopción de roles en la ingeniería de software empleando modelos psicológicos [15-17]; dichos estudios, si bien no citan explícitamente a los roles de Belbin, analizan el comportamiento de los individuos para el desempeño de funciones —roles funcionales— en equipos de desarrollo software.

3. Contexto del estudio

El estudio reportado en el presente artículo, al igual que varios de los experimentos citados anteriormente, fue desarrollado en un contexto educativo con estudiantes, y aunque podría parecer conveniente realizar este tipo de experimentos en entornos con profesionales, es aconsejable —para estudios exploratorios en el área de la Ingeniería de Software— realizar primero un experimento en un entorno académico con estudiantes y una vez que se tenga una primera evidencia, entonces replicarlo en entornos industriales [18].

3.1. Descripción de la asignatura construcción y evolución del software

Construcción y Evolución del Software, es una de las veintitrés asignaturas obligatorias de corte disciplinario del plan de estudios de la Licenciatura en Ingeniería de Software que se ofrece en la Universidad Autónoma de Yucatán, primer programa curricular en ofrecer —desde 2004— el título de Ingeniero de Software en México [19]. La asignatura tiene como objetivo general que el alumno sea capaz de desarrollar aplicaciones utilizando prácticas y técnicas que aseguren altos estándares de programación, así como la aplicación de guías que faciliten el mantenimiento; para alcanzar dicho objetivo, la asignatura se imparte en un semestre, con un total de 45 sesiones —cada una de 90 minutos. El contenido de la asignatura se encuentra organizado en seis unidades cuyos nombres y objetivos se listan a continuación:

I. *Construcción*. Al término de esta unidad el alumno aplicará las prácticas de ingeniería de software utilizadas en la codificación que implementen el diseño, como son: estándares de programación, diseño como marco de referencia directo al código, documentación interna y externa.
II. *Refactorización*. Al término de esta unidad el alumno aplicará las técnicas para reestructurar software con la finalidad de hacerlo fácil de entender y modificar a un bajo costo, sin cambiar el comportamiento observable del mismo.
III. *Reusabilidad*. Al término de esta unidad el alumno aplicará las guías para la reusabilidad durante la codificación de software.
IV. *Evaluación del código*. Al término de esta unidad el alumno analizará componentes de software para identificar y clasificar los tipos de fallos, así como definir una estrategia de integración que garantice la obtención de sistemas de calidad.
V. *Liberación del sistema*. Al término de esta unidad el alumno describirá los tipos de entrenamiento y documentación necesaria para la correcta entrega de un sistema.

VI. Mantenimiento. Al término de esta unidad el alumno describirá las actividades, herramientas y técnicas involucradas en el mantenimiento de software.

3.2. Dinámica pedagógica de la asignatura

La dinámica de la asignatura se encuentra organizada en tres tipos de sesiones: 2 sesiones de introducción, que denominaremos Sesiones Introductorias; 33 sesiones de desarrollo y evaluación de conceptos analizados a lo largo de las 6 unidades de que consta el curso, que llamaremos Sesiones de Conceptos; y 10 sesiones de presentación y evaluación del Proyecto Integrador —desarrollado en equipos de trabajo— a las que denominaremos Sesiones de Proyecto.

A través de estos tres tipos de sesiones los estudiantes desarrollan tres clases de productos, a los que hemos denominado: Resumen, Conclusión Grupal y Proyecto Integrador. Los dos primeros se desarrollan en cada una de las sesiones de Conceptos, y el tercero, es revisado en las sesiones de Proyecto. El Resumen es un documento elaborado de manera individual, que consta de dos partes, la primera parte sintetiza los conceptos de las unidades de la asignatura, enriquecidos con investigación bibliográfica, y la segunda parte incluye ejemplos en los que se evidencie la aplicación de los conceptos. La Conclusión Grupal es un reporte realizado en equipos de trabajo, que presenta una solución tentativa a un problema específico; dicha solución se basa en los conceptos de las unidades de la asignatura. En cuanto al Proyecto Integrador, es un documento que contiene el manual de usuario, el manual técnico —con el diseño de clases y el diseño de entidad relación— así como el código fuente de un software funcional, desarrollado en equipo, bajo la filosofía orientada a objetos, y en el cual se aplican los conceptos vinculados con las seis unidades de la asignatura.

A continuación se describen cada uno de los tres tipos de sesiones que se desarrollan a lo largo del curso, y con base en las cuales se promueve el logro del objetivo de la asignatura.

En las *Sesiones Introductorias*, en particular en la primera sesión se presenta el objetivo de la asignatura y de cada una de las unidades, los productos a desarrollar en equipos de trabajo, la dinámica del curso, la forma de evaluación del mismo, así como la bibliografía recomendada. En la segunda sesión se explica a los equipos de alumnos en qué consiste el Proyecto Integrador y se les presenta el plan de trabajo (ver Tabla 1) con los productos del Proyecto Integrador a entregar ("entregables") así como las fechas de entrega; al final de la sesión se aplica el inventario de autopercepción de Belbin.

Tabla 1
Plan de Trabajo del Proyecto Integrador

No	Entregables	Fecha	Sesiones de Proyecto
1	Altas	Primera, Segunda	2 Sesiones de Proyecto de la última semana del primer mes
2	Bajas, Altas	Tercera, Cuarta	2 Sesiones de Proyecto de la última semana del segundo mes
3	Cambios, Bajas y Altas	Quinta, Sexta	2 Sesiones de Proyecto de la última semana del tercer mes
4	Reportes, Cambios, Bajas y Altas	Séptima, Octava	2 Sesiones de Proyecto de la última semana del cuarto mes
5	Entrega final del Proyecto	Novena, Décima	2 Sesiones de Proyecto de la última semana del quinto mes

Fuente: Los autores.

En la Tabla 1 se puede observar que los entregables 2, 3 y 4 son incrementales en cuanto a su funcionalidad, y se espera que los equipos apliquen los conceptos desarrollados hasta el momento de la entrega, lo que pudiera implicar mejoras en ciertas partes del producto. Con relación a las diez fechas de entrega, estas se distribuyen en dos sesiones de la última semana de cada mes.

En las *Sesiones de Conceptos*, al inicio de cada sesión los alumnos de manera individual hacen entrega del producto denominado "Resumen" con los temas a ser abordado en la sesión —de acuerdo con la planeación del curso. Posteriormente, los alumnos se organizan en grupos y atienden a la presentación oral —por parte del profesor– de los temas que correspondan a la sesión en cuestión; durante las presentaciones orales, el profesor realiza interrogatorios y motiva las discusiones al interior de cada grupo. Finalmente, después de la presentación oral, los alumnos trabajan en sus equipos para desarrollar el producto "Conclusiones Grupales"; durante esta parte final de la sesión el profesor asesora las discusiones al interior del equipo, para dar solución al problema específico en cuestión.

En cuanto a las *Sesiones de Proyecto*, con base en el plan de trabajo de la Tabla 1, al inicio cada equipo debe entregar la documentación del Proyecto Integrador, definida previamente; esta documentación se corresponde con el entregable en cuestión, de tal manera que muestra únicamente la información de las partes del proyecto realizadas hasta el momento del entregable. Posteriormente, cada uno de los equipos realiza una presentación del "entregable" funcional del Proyecto Integrador. En esta presentación se muestra en ejecución el mencionado entregable y cada uno de los integrantes explica alguna parte del código. Durante la presentación de cada equipo, el profesor motiva los interrogatorios y las discusiones entre el equipo expositor y el resto de los equipos de la clase. Finalmente, el profesor expresa sus observaciones sobre los códigos y guía las reflexiones de los alumnos.

3.3. Evaluación de la asignatura

El Resumen se evalúa en una escala de 1 a 100 puntos, considerando que contenga la síntesis de los conceptos abordados en la sesión que corresponda, que presente alguna cita y referencia bibliográfica, y que presente cuando menos un código de ejemplo donde aplique alguno de los mencionados conceptos. De manera similar, la conclusión Grupal se evalúa, en una escala de 1 a 100 puntos, considerando que presente un código con una solución tentativa a alguna parte del Proyecto Integrador; dicha solución está basada en conceptos abordados en la sesión correspondiente. El entregable del Proyecto Integrador se evalúa —en una escala del 1 a 100 puntos, según se satisfaga con los aspectos de la solución ideal— tomando en consideración que el 20% de los puntos corresponde a la presentación del avance del proyecto y el 80% a la Calidad en la Legibilidad del Código. Para la evaluación de la presentación del avance se considera la participación de todos los integrantes del equipo al explicar los conceptos aplicados en diferentes partes del código.

Tabla 2
Criterios de Evaluación de la Asignatura

Método	Porcentaje	Descripción
Tareas	20%	El Resumen de cada sesión.
Participaciones	20%	La Conclusión Grupal de cada sesión.
Proyecto	60%	El Proyecto Integrador

Fuente: Los autores.

En cuanto a la evaluación a la Calidad en la Legibilidad del Código se considera la propuesta de McConnell [20]; dicha propuesta analiza el tipo de dato abstracto de las clases del código, la interfaz de las clases del código y la implementación de los métodos de las clases. Se resalta que se consideran requisitos la documentación definida previamente en la definición del Proyecto Integrador y que funcione el entregable del proyecto integrador. La calificación final del Proyecto Integrador, en una escala del 1 al 100 puntos, se calcula sumando el 10% del primer entregable, el 20% del segundo entregable, el 20% del tercer entregable, el 25% del cuarto entregable y el 25% del quinto entregable. La Tabla 2 ilustra los criterios antes citados.

4. Metodología de estudio

Con base en los trabajos relacionados citados anteriormente, se diseñó un experimento controlado para explorar la influencia del uso de la Teoría de Roles de Belbin en la integración de equipos de desarrollo software, en nuestro caso, la influencia se evaluó en los artefactos software generados, en particular, en la legibilidad del código generado por los equipos de estudiantes que se encuentran en proceso de formación como Ingenieros de Software y cursan la asignatura Construcción y Evolución de Software.

4.1. Objetivo, hipótesis y variables

Con el objetivo de explorar si los equipos de desarrollo de software integrados con base en la Teoría de Roles de Belbin —a los que denominaremos Compatibles— generan artefactos de mejor calidad en cuanto a la legibilidad del código, que los equipos integrados sin utilizar algún criterio en particular —a los que denominaremos Tradicionales— se plantearon las siguientes hipótesis:

H0: La calidad media de la legibilidad del código generado por los Equipos Tradicionales es mayor o igual a la calidad media de la legibilidad del código generado por los Equipos Compatibles.

H1: La calidad media de la legibilidad del código generado por los Equipos Tradicionales es menor que la calidad media de la legibilidad del código generado por los Equipos Compatibles.

El factor a controlar en este experimento es el mecanismo de integración de los equipos de desarrollo software, el cual tiene dos alternativas: (1) Equipos Compatibles, y (2) Equipos Tradicionales. Por su parte, la variable respuesta —dependiente— Calidad de la Legibilidad del Código, será medida en los artefactos que entregan los estudiantes a lo largo del curso.

Aspectos como la complejidad del problema a resolver, el lenguaje de programación utilizado, y la expertiz de los participantes, se considera parámetros que no afectan o sesgan los resultados del estudio, en el caso de los dos primeros, son parámetros homogéneos para todos los equipos de desarrollo, y para el caso de la expertiz, al ser equipos integrados por estudiantes que se encuentran apenas en su proceso de formación, resultan también homogéneos.

4.2. Participantes/Sujetos

Los participantes en el experimento fueron 33 estudiantes de carrera de Ingeniero de Software de la Universidad Autónoma de Yucatán, que cursaban la asignatura Construcción y Evolución Software en el semestre agosto-diciembre de 2015, asignatura ubicada en el quinto semestre de la carrera; de acuerdo a la clasificación sobre experiencia en programación propuesta por Dreyfus & Dreyfus [21], los estudiantes de este experimento se clasifican como novicios avanzados, es decir, cuentan con experiencia práctica y conocimiento sobre los aspectos fundamentales del lenguaje de programación Java.

Con los 33 estudiantes inscritos a la asignatura se formaron 11 equipos de desarrollo software de 3 integrantes cada uno; para la conformación de los equipos se utilizó la información obtenida con la administración del inventario de autopercepción de Belbin en la segunda sesión introductoria, con base en los roles primarios identificados en los estudiantes, se integraron —utilizando la Teoría de Belbin— 6 equipos con roles compatibles, y 5 equipos adicionales con alumnos asignados de manera aleatoria —sin observar el rol primario de cada estudiante.

En virtud de que las mediciones se obtendrían sobre los productos generados por los equipos de desarrollo, los sujetos experimentales en este caso fueron los 11 equipos de desarrollo integrados por los investigadores (ver Tabla 3), los cuales fueron agrupados utilizando como referente la teoría de roles de Belbin (ver Fig. 1).

Tabla 3
Sujetos experimentales por tratamiento

Tratamientos	Sujetos (Equipos)
Equipos Compatibles	I, II, III, IV, V, VI
Equipos Tradicionales	VII, VIII, IX, X, XI

Fuente: Los autores.

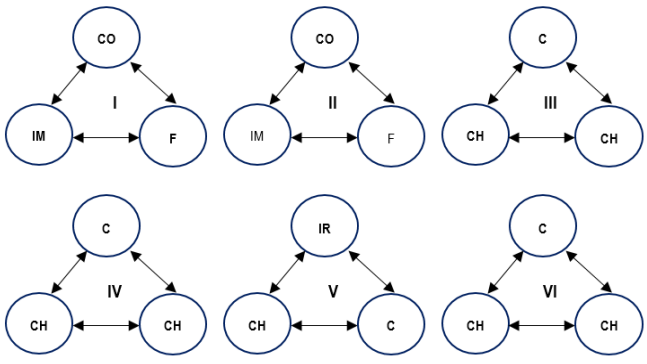

Figura 1. Conformación de los equipos de desarrollo compatibles.
Fuente: Los autores.

Tabla 4
Calidad del Código Generado por los Equipos

Equipos	Calidad	$\bar{x}$
Compatibles	74, 83, 79, 90, 75, 80	80.16
Tradicionales	74, 79, 66, 82, 68	73.80

Fuente: Los autores.

4.3. Procedimiento de recolección de datos

En el ámbito del contexto del estudio, los equipos de desarrollo software construyen su sistema software en sesiones no presenciales, y de acuerdo con lo descrito en la sección referente a la dinámica pedagógica de la asignatura, en las sesiones de Proyecto —de acuerdo con el Plan de trabajo— los equipos hacen entrega del código generado según el avance del curso.

Para medir el grado de la calidad en la legibilidad del producto software, se analizó el código y se registró el progreso realizado en el software. Los criterios y métricas usados para evaluar la calidad en la legibilidad del código generado por los equipos fueron basados de la propuesta de McConnell [20]: Tipo de datos abstracto (número de atributos y operaciones que guardan cohesión entre sí, están escritos en términos del problema y definen la entidad), interfaz (número de métodos que ocultan detalles de implementación, guardan cohesión entre ellos, están escritos en términos del problema y definen la clase), método (número de operaciones que contribuyen con el propósito del método, están escritos en término del problema, guardan cohesión entre sí y es evidente su organización). Utilizando un esquema similar al reportado en [22] los criterios antes mencionados fueron evaluados en una escala del 1 al 100, según se satisfaga con los aspectos de la solución ideal. De la calificación total de la calidad en la legibilidad del software, el 20% corresponde a la participación y el 80% evalúa los criterios anteriormente mencionados (ver ecuación 1). La calificación se evalúa en un rango de 0 a100.

$$\begin{aligned} &Calidad\ del\ entregable \\ &\quad = (((TipoDatoAbstracto \times 0.2) \\ &\quad + (Interfaz \times 0.4) + (Metodo \times 0.4)) \\ &\quad \times 0.8) + (Participacion \times 0.2) \end{aligned} \tag{1}$$

La fórmula, presentada en la ecuación (1) fue aplicada a cada uno de los cinco entregables definidos conforme el plan de trabajo establecido. Para calcular la calidad en la legibilidad del proyecto software se utilizó la formula presentada en (2).

$$\begin{aligned} Calidad = &(Calidad\ del\ entregable\ 1 \times 0.1) \\ &+ (Calidad\ del\ entregable\ 2 \times 0.2) \\ &+ (Calidad\ del\ entregable\ 3 \times 0.2) \\ &+ (Calidad\ del\ entregable\ 4 \times 0.25) \\ &+ (Calidad\ del\ entregable\ 5 \times 0.25) \end{aligned} \tag{2}$$

5. Resultados

En la Tabla 4 se muestran las mediciones obtenidas de los once equipos de desarrollo una vez que se concluyó el curso y los cinco entregables habían sido revisados.

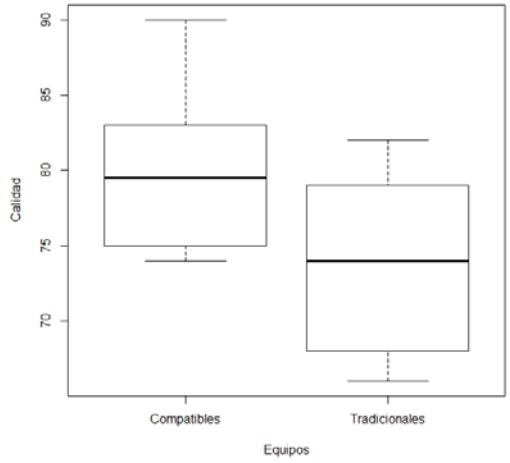

Figura 2. Gráfico de Caja para ambos Tratamientos.
Fuente: Los autores.

5.1. Análisis descriptivo

Para ilustrar gráficamente el comportamiento de los datos, se construyeron diagramas de cajas y bigotes (ver Fig. 2); dichos diagramas son útiles porque nos permiten ver la dispersión de los datos respecto de la mediana, tener una visión general de la simetría de la distribución de los mismos, así como el poder comparar los subconjuntos de mediciones. En la gráfica podemos observar que las medidas obtenidas de los equipos Tradicionales son más simétricas que las que provienen de equipos Compatibles, los cuales, por cierto, presentan un sesgo positivo; también se puede observar que la mediana de los valores recogidos en los equipos Compatibles, coincide con el tercer cuartil del conjunto de mediciones de los equipos Tradicionales, o dicho de otra manera, el 75% de las mediciones que provienen de los equipos Tradicionales se ubican por debajo del valor que representa al 50% de las mediciones recogidas en los equipos Compatibles.

5.2. Prueba de hipótesis

Las observaciones descritas nos inducen a pensar que existen diferencias entre los subconjuntos de datos, y que estas son positivas respecto de las mediciones obtenidas en los equipos formados considerando la Teoría de Roles de Belbin, lo cual coincide con las hipótesis planteadas inicialmente, por lo que procederemos a probar dichas hipótesis a través de un análisis estadístico. El modelo estadístico que se utilizará la prueba de hipótesis, es el modelo de un solo factor —mecanismo de conformación de los equipos de desarrollo software— el cual se representa con la ecuación descrita en (3).

$$\begin{aligned} y_{ij} &= \mu + \tau_i + \epsilon_{ij}, \\ con\ i &= 1, \cdots, a;\ j = 1, \cdots, n_i \end{aligned} \tag{3}$$

Donde y_{ij} es la observación j-ésima del i-ésimo tratamiento (nivel i-ésimo del factor), μ es la media común de todos los tratamientos, τ_i es el efecto del nivel i del factor, y $\epsilon_{ij} \sim N(0, \sigma^2)$

representa el error aleatorio, y que σ^2 es la varianza común de las poblaciones. En particular, utilizaremos la prueba t para muestras independientes con contraste unilateral, cuyas hipótesis estadísticas se representan como sigue:

$$Ho: \mu1 \leq \mu2 \text{ ó también } \mu1 - \mu2 \leq 0$$
$$H1: \mu1 > \mu2 \text{ ó también } \mu1 - \mu2 > 0$$

Para la realización del análisis estadístico se seleccionó el lenguaje R y el paquete R-Commander, ambos de libre distribución; al aplicar la prueba antes citada con un nivel de confianza del 90%, se obtuvo un p-valor (0.06531) es menor que el nivel de significación α (0.1), se rechaza la hipótesis nula y por tanto se acepta la alternativa, la cual indica que existe diferencia positiva entre la calidad media de la legibilidad del código generado por los equipos Compatibles y la calidad media del código generado por los equipos Tradicionales.

5.3. Validación de supuestos

Una vez obtenidos los resultados de nuestra prueba de hipótesis, resulta necesario comprobar la validez de los supuestos de nuestro modelo esatdísticos, en nuestro caso, se deben verificar los supuestos de:

- *Normalidad*: los datos tienen un comportamiento como el de la distribución normal.
- *Homocedasticidad*: la varianza de los errores es constante.
- *Independencia*: las mediciones provienen de poblaciones independientes.

El tercer supuesto lo abordaremos explorando la normalidad de manera gráfica. En la gráfica de Q-Q (ver Fig. 3) se observa una tendencia de línea recta, por lo que no tenemos evidencia en contra de la normalidad de los residuos. El segundo supuesto se verifica explorando la homocedasticidad de las varianzas de manera gráfica; en el gráfico de residuos contra tratamientos (ver Fig. 4) se observan distribuciones similares entre los tratamientos, por lo que podemos considerar que se cumple este segundo supuesto.

El primer supuesto se aborda tomando en consideración la propiedad de aleatoriedad, en el caso de los equipos Compatibles, se logró a través de la asignación aleatoria de los estudiantes según su rol, a los sujetos (equipos) experimentales; y en el caso de los equipos Tradicionales, la asignación de los estudiantes fue totalmente aleatoria. Para corroborar este seupuesto, se decidió elaborar la gráfica de residuos (ver Fig. 5) como una función de los valores estimados; del análisis a la gráfica podemos observar que la nube de datos dispersos nos indica que no hay razones para sospechar cualquier violación de los supuestos de independencia.

6. Discusión

Los resultados del estudio abonan información en cuanto a que la formación de equipos de desarrollo software que utilizan la Teoría de Roles de Belbin para su integración, resulta ser una estrategia adecuada ya que presentan evidencia de mejoras positivas en el desempeño de los equipos, en particular, en la calidad de la legibilidad del código generado.

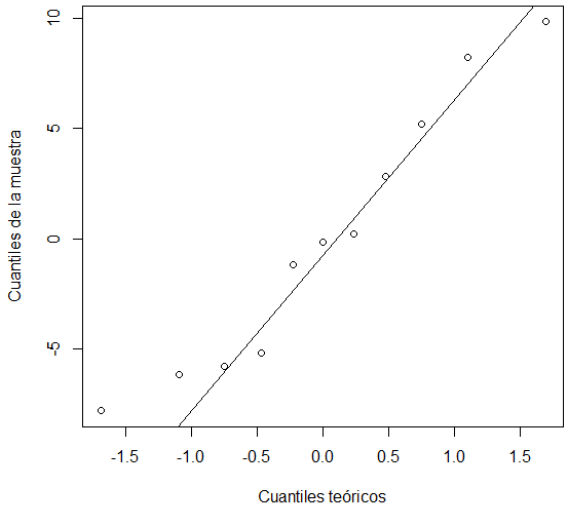

Figura 3. Gráfico Q-Q normal.
Fuente: Los autores.

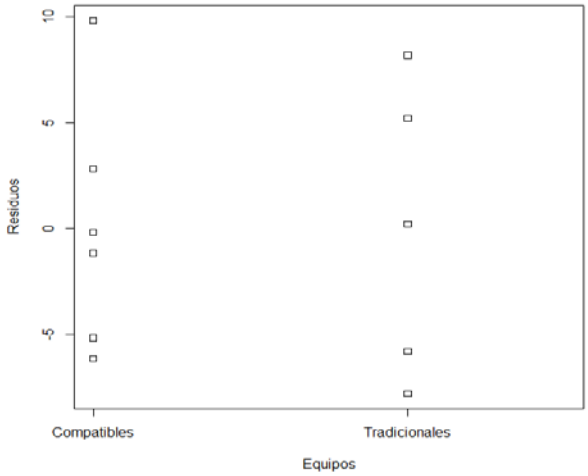

Figura 4. Gráfico de residuos contra tratamientos.
Fuente: Los autores.

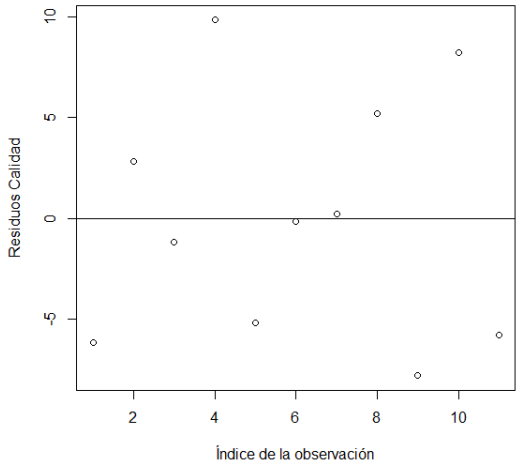

Figura 5. Gráfico de residuos de las mediciones.
Fuente: Los autores.

En el estudio reportado, el número de participantes y por ende, de sujetos experimentales —equipos de tres alumnos cada uno— fue limitado, por lo que la conformación de los equipos no permitió indagar sobre si la presencia de ciertos roles, o si la diversidad de los mismos puede influir en el desempeño o efectiviad del equipo Los experimentos suelen estar están sujetos a diferentes tipos de amenazas que pueden incidir de manera negativa en sus resultados, y por ello es necesario ser muy cuidadosos en su paneación y ejecución. Cook & Campbell [23] proponen cuatro tipos de amenazas a la validez: conclusión, interna, externa y de constructo.

6.1. Validez de conclusión

Estas amenazas se refieren a cuestiones que pueden afectar la habilidad de obtener una conclusión correcta acerca de la existencia de una relación entre el tratamiento y la variable de respuesta. Una posible amenaza que pudo haber impactado en la validez de conclusión es la referente a la confiabilidad de las mediciones; aunque todas las mediciones fueron realizadas por el mismo profesor y las métricas fueron definidas al inicio del curso, es posible cierta subjetividad en las medidas después de largas jornadas de revision. Sin embargo, esta subjetividad podría haberse mitigado con la observación periódica de los entregables de los sujetos.

6.2. Validez interna

Las amenazas en esta categoría se refieren a si los resultados observados se debieron a otros factores distintos a los tratamientos considerados. Para abordar estas amenazas, los sujetos se asignaron de manera aleatoria a los tratamientos, según su tipo de Rol Belbin, todos ellos se conformaron por alumnos clasificados como novicios avanzados [21] por lo que los sujetos participaron en el experimento bajo mismas condiciones (con proyecto diferentes, aunque equivalentes funcionalmente).

6.3. Validez de constructo

Las amenazas en esta categoría se refieren a aspectos que pueden influir negativamente en la relación que hay entre la teoría y la observación. Para abordar esta amenaza los temas fueron desarrollados durante las clases de todo un semestre, evaluando y retroalimentando los conocimientos teóricos de manera periódica.

6.4. Validez externa

Estas amenazas se refieren a aspectos que pueden limitar la habilidad de generalizar los resultados del experimento a otros contextos, por ejemplo generalizar los resultados a la industria; en nuestro estudios, el uso de estudiantes como sujetos en lugar de profesionales tal vez pudo afectar este tipo de validez. No obstante, como se describe en [18], el uso de estudiantes como sujetos, permite al investigador obtener evidencia preliminar para confirmar o refutar hipótesis que pueden ser contrastadas posteriormente en contextos industriales.

7. Conclusiones

El presente estudio exploratorio tiene como propósito el contribuir con evidencia empírica en torno a la bondad de utilizar la Teoría de Roles de Belbin para la integración de equipos de desarrollo software. Los resultados del estudio, ciertamente limitado por el número de sujetos, genera evidencia que nos permite concluir —con base en el modelo estadístico seleccionado y con un nivel de confianza del 90%— que los equipos formados con base en la Teoría de Roles de Belbin, presentan resultados significativamente mejores —en cuanto a la legibilidad de la calidad del código— que los equipos que fueron integrados de manera aleatoria. Por otro lado, las lecciones aprendidas al usar la Teoria de Roles de Belbin como estrategia de formación de equipos en entornos académicos, en función de los resultados obtenidos, resulta ser una alternativa positiva, a los esquemas tradicionalmente utilizados basados en: aleatoriedad, o preferencias de los estudiantes.

La información obtenida con el experimento controlado nos motiva a continuar realizando estudios comparativos entre equipos compatibles y equipos tradicionales, en otras actividades vinculadas con el proceso de desarrollo software (p.e. Estimación, Requistos, Diseño).

Agradecimientos

Agradecemos a los estudiantes de la carrera de Ingeniería de Software de la Universidad Autónoma de Yucatán, su invaluable participación en el presente estudio. La dedicación evidenciada en el desarrollo de los proyectos a lo largo de curso, es sin lugar a dudas, una de las principales fortalezas que brinda validez a nuestras aportaciones.

Referencias

[1] Bourque P. and Fairley, R., Guide to the software engineering body of knowledge (SWEBOK V3.0). IEEE Computer Society, 2014. DOI: 10.1109/52.805471

[2] DeMarco, T. and Lister, T., Peopleware productive projects and teams 2nd Ed. New York, USA: Dorset House Publishing Co., 1999.

[3] Humphrey, W., Introduction to the team software procesess. Reading, USA: Addison Wesley Longman Inc., 2000.

[4] Belbin, M., Team roles at work. Oxford, USA: Elsevier Butterworth Heinemann, 1993.

[5] Briggs-Myers, I. and Briggs, K.C., Myers-Briggs Type Indicator (MBTI). Palo Alto, CA: Consulting Psychologists Press., 1985

[6] Mumma, F.S., What makes your team tick?. King of Prusia, PA: HRDQ, 1994.

[7] Margerison, C.J. and McCann, D.J., Team management profiles: Their use in managerial development, Journal of Management Development, 4(2), pp. 34-37, 1985. DOI: 10.1108/eb051580

[8] Belbin, M., Management teams. Why they succeed or fail. New York, USA: John Wiley & Sons, 1981.

[9] Aritzeta, A., Swailes, S. and Senior, B.B., Team roles: Psychometric evidence, construct validity and team building. Hull, UK. University od Hull, 2005.

[10] Johansen, T., Predicting a team's behaviour by using Belbin's Team Role self perception inventory. Thesis dissertation at Department of Management & Organisation, University of Stirling, Stirling, U.K., 2003.

[11] Pollock, M., Investigating the relationship between team role diversity and team performance in information systems teams. Journal of Information Technology Management, 20(1), pp. 42-55, 2009.

[12] Henry, S. and Stevens, K., Using Belbin's leadership role to improve team effectiveness: An empirical investigation. Journal of Systems and Software, 44(3), pp. 241-250, 1999. DOI: 10.1016/S0164-1212 (98)10060-2

[13] Richard, T., Group dynamics and software engineering. In: Manns, M.L. (Ed.) Object Oriented Programming Systems Languages and applications: Educators' Symposium, 1-5 November 1999, Denver, Colorado, USA, 1999.

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

[15] Wynekoop, J. and Walz, D., Investigating traits of top performing software developers. Information Technology and People, 13(3), pp. 186-195, 2000. DOI: 10.1108/09593840010377626

[16] Acuña, S. and Juristo, N., Assigning people to roles in software projects. software: Practice and Experience, 34(7), pp. 675-696, 2004. DOI: 10.1002/spe.586

[17] Jarillo, P., Enríquez, C. y Sánchez, R., Identificación del factor humano en el seguimiento de procesos de software en un medio ambiente universitario, Computación y Sistemas, 19(3), pp. 577-588, 2015. DOI: 10.13053/cys-19-3-2206

[18] Genero, M., Cruz-Lemus, J. y Piattini, M., Métodos de investigación en ingeniería de software. RA-MA, 2014.

[19] Aguilar, R. y Díaz, J., La ingeniería de software en México: Hacia la consolidación del primer programa de licenciatura. Revista de Tecnología Educativa, 2(2), pp. 6-17, 2015.

[20] McConnell, S., Code Complete. 2E, Microsoft Press, Redmond, WA, 2004.

[21] Dreyfus, H. and Dreyfus, S., Mind over machine. The power of human intuition and expertise in the era of the computer. New York: Basil Blackwell, 1986.

[22] Acuña, S., Gómez, M. and Juristo, N., How do personality, team process and task charcaterístics relate to job satisfaction and softwarequality?. Information and Software Technology, 51(2), pp. 627-639, 2009. DOI: 10.1016/j.infsof.2008.08.006

[23] Cook T. and Campbell, D., Quasi-experimentation – Design and analysis issues for field settings. Houghton Mifflin Company, Boston, 1979.

A.A. Aguileta-Güemez, es Lic. en Ciencias de la Computación por la Universidad Autónoma de Yucatán (UADY), México; MSc. en Ciencias Computacionales por el Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), campus Monterrey, México. Actualmente es profesor asociado en la Facultad de Matemáticas e integrante del Grupo Académico de Tecnologías para la Formación en Ingeniería de Software de la UADY. La línea de investigación de su interés es en torno a la calidad en la Ingeniería de Software.
ORCID: 0000-0001-5155-3543.

J.P. Ucán-Pech, es Dr. en Sistemas Computacionales por la Dirección de Posgrado e Investigación de la Universidad del Sur, campus Mérida, México. MSc. en Sistemas Computacionales con especialidad en Ingeniería de Software por el Instituto Tecnológico de Mérida, México. Es profesor titular en la Facultad de Matemáticas de la Universidad Autónoma de Yucatán e integrante del Grupo Académico de Tecnologías para la Formación en Ingeniería de Software de la UADY. Es miembro de la Academia Mexicana de Computación (AMEXCOMP). Su trabajo de investigación se centra en temas relacionados con la Ingeniería de Software, Ingeniería Web e Informática Educativa.
ORCID: 0000-0002-1013-6396.

R.A. Aguilar-Vera, obtuvo el grado de Dr. por la Universidad Politécnica de Madrid, España (Mención de Doctor Europeo) y el de MSc. en Ingeniería de Software por la misma Institución, es MSc. en Educación Superior por la Universidad Autónoma de Yucatán. Actualmente es profesor de tiempo completo en la Facultad de Matemáticas e integrante del Grupo Académico de Tecnologías para la Formación en Ingeniería de Software de la Universidad Autónoma de Yucatán. Es miembro de la Academia Mexicana de Computación (AMEXCOMP), e integrante del Comité de Acreditación del Consejo Nacional de Acreditación en Informática y Computación (CONAIC). Su trabajo de investigación incluye las siguientes áreas: Ingeniería de Software e Informática Educativa.
ORCID: 0000-0002-1711-7016.