Actas de las Jenui, vol. 8. 2023.
Páginas: 105-112

Atención a la diversidad de estilos de aprendizaje: experiencia en la docencia de arquitectura del software

Laura M. Castro
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña (A Coruña)
lcastro@udc.es

Resumen

La arquitectura del software (AS) es una parte de la ingeniería del software que cobra importancia a medida que los proyectos software son más ambiciosos, complejos y críticos. Por la naturaleza de los conceptos de los que se ocupa, y sus efectos desde la definición hasta el mantenimiento y evolución de los sistemas, la AS es una disciplina compleja en sí misma, con grandes desafíos a la hora de diseñar su docencia.

La mayoría de los métodos de docencia-aprendizaje se apoyan en la comunicación verbal y escrita, lo que hace que se alineen con aquellos estilos de aprendizaje para los que este tipo de comunicación resulta más significativa. En este contexto, quienes presentan un estilo de aprendizaje que se beneficia más de actividades psicomotrices o kinestésicas, deben recurrir a procesos cognitivos que les resultan menos naturales.

Este trabajo presenta una experiencia docente diseñada para favorecer la atención a la diversidad de estilos de aprendizaje en el aula, en el contexto de la arquitectura del software (AS), y como complemento a otros elementos más clásicos (clases magistrales, prácticas de laboratorio). En concreto, se propone un innovador uso del *role-playing*, donde el alumnado adopta el papel de componentes software, como estrategia para la adquisición efectiva de conceptos y características de un conjunto de modelos arquitecturales de referencia.

Abstract

Software architecture (SA) is an aspect of software engineering that becomes increasingly important as projects grow more ambitious, complex and critical. SA is a complex discipline, due to the nature of the architectural concepts, and their effects on systems' definition, maintenance, and evolution. Teaching SA, thus, is an ongoing challenge.

Most teaching methods rely on verbal and written communication, which aligns them with those learning styles for which this type of communication is most significant. In this context, people who have a learning style that benefits more from psychomotor or kinaesthetic activities, must resort to cognitive processes that are less natural to them.

This work presents a teaching experience designed to favour attention to the diversity of learning styles in the classroom, in the context of software architecture (SA), and as a complement to master classes or lab work. Specifically, we propose an innovative use of *role-playing*, where students adopt the role of software components, as a strategy for the effective acquisition of concepts and characteristics of a set of architectural reference models.

Palabras clave

Arquitectura del software, estilos de aprendizaje, *role-playing*, juegos de rol, modelos arquitecturales.

1. Introducción

Transmitir los conceptos relacionados con la arquitectura de software (AS) de forma significativa y eficaz es difícil. Son múltiples las razones que se esgrimen para explicar esta realidad, aunque parece haber cierto consenso en torno a que la AS expone a los estudiantes a nociones y problemas de una escala significativamente mayor que los manejados en niveles previos (programación, diseño), y que además (y precisamente por esa razón de escala) no son fáciles de reproducir dentro del contexto del aula y del ámbito académico [32]. Los conceptos de AS requieren el desarrollo de mayores capacidades de abstracción, y con frecuencia abordan por primera vez cuestiones como el rendimiento, la escalabilidad o la seguridad [11]. Por último, pero no por ello menos importante, hay competencias no técnicas en juego, como la comunicación [16] y la toma de decisiones [5], que también deben desarrollarse [8].

Paralelamente, sabemos que las personas experimentamos de manera diferente los procesos mentales involucrados en la adquisición de nuevas habilidades o conocimientos (es decir, *aprendemos* de maneras distintas), y que esas particularidades hacen que ciertos tipos de experiencias de docencia-aprendizaje sean más o menos efectivas dependiendo del individuo. Una de las caracterizaciones de esta diversidad de experiencias

Figura 1: Estilos de aprendizaje de Kolb.[1]

más utilizada es la propuesta por el modelo de aprendizaje experiencial de Kolb [22] (cfg. figura 1), que describe los siguientes estilos de aprendizaje:

- *Estilo asimilador*. Personas que aprenden mejor mediante la observación de la nueva información o experiencia: analizan los nuevos datos o conocimientos bajo el prisma de su experiencia previa.
- *Estilo convergente*. Personas que aprenden mejor mediante la reflexión pura, siendo capaces de generar nuevas ideas a partir de las que modificar sus conceptos, conocimientos o habilidades.
- *Estilo acomodador*. Personas que aprenden mejor poniendo en práctica los nuevos conceptos mediante la experimentación en el mundo real.
- *Estilo divergente*. Personas que aprenden mejor enfrentándose a una experiencia o situación concreta que genera la necesidad de adquirir los nuevos conceptos o habilidades.

Aunque los estilos de aprendizaje forman un continuo, también sabemos que un determinado individuo no puede destacar al mismo tiempo en variables del mismo eje (por ejemplo, *pensar* y *sentir*). Teniendo esto en cuenta, los beneficios de conocer y abordar los diferentes perfiles de aprendizaje en la docencia se han investigado ampliamente, tanto en el contexto de la ingeniería [28] como de la informática [10, 20], donde se han aplicado especialmente en programación [14, 18].

Teniendo en cuenta que las actividades docentes tradicionales, como las clases magistrales o el desarrollo de proyectos, están más alineadas con algunos perfiles de aprendizaje, en este trabajo proponemos una innovadora actividad docente de la AS, diseñada pensando en los perfiles generalmente olvidados, para mejorar la comprensión de las diferentes características, fortalezas y debilidades de las AS de referencia. En concreto, nuestra contribución central es el *uso de juegos de rol de manera inédita*: asignando a los estudiantes diferentes *roles de componentes software* en las arquitecturas de referencia que se estudian en la materia.

[1] *Fuente:* https://www.actualidadenpsicologia.com/la-teoria-de-los-estilos-de-aprendizaje-de-kolb/

2. Estado del arte

2.1. Docencia en AS

Las dificultades de la docencia de la AS se han abordado de diversas maneras. Por un lado, se han puesto en práctica metodologías innovadoras, normalmente tratando de sacar el máximo partido a la organización de estudiantes en grandes equipos con roles diferenciados [32]. A estos grandes equipos se les suele pedir que trabajen en sistemas realistas de gran escala [7], teniendo los proyectos de código abierto una presencia relevante en este tipo de experiencias [6, 32]. Aunque los métodos docentes y de evaluación específicos varían (desde la clásica resolución de problemas [12] hasta la alineación constructiva [4] o el aprendizaje colaborativo y distribuido [13]), el objetivo común suele ser reproducir, en la medida de lo posible, cómo trabajan los arquitectos de software en un entorno profesional [19], tanto técnica como metodológicamente [36].

Complementariamente, encontramos enfoques que se centran en atraer y mantener la atención y la participación de los estudiantes, utilizando el desarrollo de juegos como motivación [34, 37].

En la literatura encontramos también trabajos enfocados a las herramientas de soporte a la docencia de AS, que proponen desde el uso de versiones específicas de UML en 3D [25], o del modelo C4 [33] para alcanzar niveles más altos de expresividad, hasta la identificación de requisitos [31], o el uso de herramientas de razonamiento específicas [21].

Aunque algunas de estas iniciativas han sido evaluadas durante largos períodos de tiempo, incluso de más de una década [26, 32, 38], no hemos encontrado ninguna que se centre en los propios estudiantes, en lugar de en los conocimientos y capacidades que queremos que desarrollen. En ese sentido, únicamente se han realizado propuestas de adelanto de la AS a los primeros años de la formación [3, 9].

2.2. Juegos de rol en la docencia

Los métodos docentes participativos ofrecen una amplia gama de opciones aplicables al contexto de la informática [15]: lluvia de ideas, diálogos dirigidos, grupos de discusión, debates, mesas redondas o juegos de rol, entre otros. De este abanico de opciones, en este trabajo hemos decidido utilizar los juegos de rol como herramienta didáctica, pues están ampliamente reconocidos como una técnica eficaz en múltiples ámbitos de la formación y la educación.

Los juegos de rol, en contextos educativos, constituyen un método de aprendizaje experiencial en el que el alumnado participa en un escenario, diseñado por la/el docente, representando una de las partes que interactúan en él. El escenario propuesto representa un entorno seguro donde los estudiantes desarrollan una experiencia significativa con margen a la improvisación.

En ingeniería del software, contamos con experiencias previas, en su mayoría pensadas para simular las condiciones de un entorno profesional en lo que se refiere a los procesos de desarrollo o la extracción de requisitos [24, 39, 40]. Así, habitualmente se pide a los estudiantes que desempeñen diferentes papeles, desde los clientes o *stakeholders* hasta las distintas funciones técnicas dentro del equipo de desarrollo.

No obstante, en este trabajo hemos elegido usar los juegos de rol como instrumento docente de la AS de manera diferente: en lugar de asignar roles *humanos* a los estudiantes, les asignamos roles de *componentes software* de las diferentes arquitecturas de referencia, (ver Sección 3). Si bien el juego de rol "tradicional" (es decir, centrado en roles personales) se ha utilizado antes en el contexto de la docencia de la AS [23], solo hemos encontrado un enfoque similar al nuestro (es decir, centrado en el software) en el contexto de un curso de programación en el que los estudiantes desempeñan el papel de "objetos" para comprender los conceptos de la orientación a objetos [2].

3. Juegos de rol arquitecturales

Las metodologías docentes que se venían utilizando en el curso de AS impartido por los autores de este trabajo se alinean más fuertemente con algunos de los estilos de aprendizaje de Kolb. Las clases magistrales y los materiales proporcionados en relación con estas (referencias bibliográficas, artículos, vídeos, tareas individuales) funcionan bien para las personas cuyos rasgos de aprendizaje más destacados son observar y pensar, es decir, aquellos estudiantes que son reflexivos y sacan el máximo partido de la observación, y la conceptualización. De forma complementaria, el trabajo práctico durante las sesiones de laboratorio da cobertura adicional a quienes se guían más por la experimentación activa. Sin embargo, aquellas personas con estilos de aprendizaje *acomodador* y *divergente*, en los que la experimentación o la observación se beneficiarían más de la combinación con la experiencia tangible, quedaban al arbitrio de la autogestión de sus necesidades como parte de su trabajo autónomo.

Para abordar esta situación, nuestra propuesta ha consistido en *diseñar un juego de rol para cada una de las diferentes arquitecturas de referencia estudiadas* en la materia. La intención es proporcionar a los estudiantes una experiencia en primera persona de las características de cada propuesta arquitectural, sus implicaciones prácticas en relación con diferentes requisitos no funcionales (disponibilidad, rendimiento, seguridad) y, por tanto, las ventajas y desventajas de los diferentes modelos arquitectónicos.

3.1. Jugando a las capas

Las instrucciones del juego de rol de la arquitectura en capas se muestran en el cuadro 1 (página 4). El alumnado debe formar sistemas con la misma funcionalidad pero distinto número de capas. Dependiendo del número de estudiantes, cada capa puede asimilarse a una persona o varias. Los resultados esperados son experiencias de primera mano de:

- Potencial de reutilización de las capas (i.e. intercambio de estudiantes) entre sistemas, que aumentan cuando sus responsabilidades son específicas en lugar de generalistas, y sus interfaces bien especificadas. Impacto en la mantenibilidad y evolución del software.
- Limitación del impacto de los cambios a cada capa individual (o adyacentes, en todo caso).
- Limitaciones estructurales al rendimiento, relacionadas con el número de capas.
- Posibilidad de actuar sobre cada solicitud en dos momentos diferentes en cada capa (al recibirla, antes de redirigirla a la siguiente, y al devolverla a la anterior) y sus implicaciones (por ejemplo, en términos de seguridad).

Un ejemplo concreto utilizado en esta sesión consistió en diseñar y ejecutar sistemas para calcular la suma de números escritos en lenguaje natural. Las peticiones a la capa de entrada (entregadas en papel por el cliente/docente al estudiante con ese rol en cada sistema) eran del tipo "*trescientos cincuenta y cuatro mil, ochocientos veintiuno*", y la salida que los estudiantes debían producir (en papel, entregado al cliente/docente), 23 ($3 + 5 + 4 + 8 + 2 + 1$). Por grupos, diseñan libremente diferentes sistemas, todos organizados en capas, pero cuyo número y responsabilidad concreta varía ligeramente (según opten por tener responsabilidades de grano más o menos fino en cada capa/persona). Estas variaciones permiten experimentar, por ejemplo, que la salida más rápida no es conseguida nunca por los sistemas con mayor número de capas/estudiantes (por muy rápidos que traten de ser), aunque si lo que se premia es la precisión, sistemas con capas de verificación de resultados intermedios tendrán menores tasas de fallo. A lo largo de la sesión, se fuerzan intercambios de capas/estudiantes entre sistemas/grupos (por ejemplo, quienes calculan las sumas), o cambios de "implementación" (uso interno de números romanos en lugar de arábigos, soporte de representación escrita en otros idiomas) para comprobar su impacto en el resto.

Roles	Instrucciones
Eres la capa de entrada	Recibes solicitudes del cliente (docente) y debes transmitirlas a la siguiente capa una vez procesadas. Espera por la respuesta, que tendrá que entregarte esa misma capa siguiente. Devuelve la respuesta al cliente que hizo la solicitud. No aceptes una nueva solicitud hasta que hayas recibido y entregado la respuesta a la última.
Eres una capa intermedia	Admite solicitudes solo de tu capa anterior, y pásalas a la siguiente después de procesarlas. No admitas una nueva solicitud hasta que hayas recibido la respuesta a la última. Devuelve la respuesta solo a la capa anterior.
Eres la capa más interna	Acepta peticiones solo de tu capa anterior y procésala. No aceptes nuevas solicitudes hasta que hayas terminado. Devuelve la respuesta solo a tu capa anterior.

Cuadro 1: Roles en juego en la arquitectura en capas[a]

[a] Por limitaciones de espacio, estos cuadros de descripción de roles son versiones resumidas de las proporcionadas al estudiantado.

3.2. Jugando al *pipeline*

Las instrucciones del juego de rol de la arquitectura en *pipeline* se recogen en el cuadro 2 (página 6). En este caso, se pide a los estudiantes agruparse y definir los filtros de diferentes sistemas en *pipeline* para un mismo objetivo. Los resultados esperados son experiencias de:

- Potencial de reutilización de los filtros de un sistema a otro, mayor cuando las responsabilidades son concretas, sus interfaces bien definidas, y el formato de datos, estándar.
- Limitación del impacto de los cambios a filtros individuales (o adyacentes, como mucho).
- Consecuencias del paralelismo estructural (e.g., rendimiento) y de la posibilidad de duplicar etapas de filtrado que son cuellos de botella.
- Implicaciones de la unidireccionalidad de las peticiones (por ejemplo, en el manejo de errores).
- Implicaciones de que la entrada y salida del sistema sean puntos diferenciados.

3.3. Jugando al repositorio

El cuadro 3 muestra las instrucciones para el juego de la arquitectura en repositorio. Si el número de estudiantes lo permite, puede usarse el encerado como repositorio; alternativamente, puede limitarse a una hoja de papel. Las experiencias de primera mano incluyen:

- Independencia entre clientes (en términos de desarrollo, objetivos, temporización, etc.).
- Optimización de comunicaciones *many-to-many*.
- Ausencia de "objetivos del sistema" (condición de parada) y sus consecuencias (ausencia/necesidad de coordinación, conflictos).
- Condición de SPoF (*Single Point of Failure*) del componente repositorio.

3.4. Jugando a cliente-servidor

El cuadro 4 recoge las instrucciones del juego correspondiente a la arquitectura cliente-servidor. Los estudiantes crean, despliegan y protagonizan la evolución de varios sistemas, experimentando:

- Independencia entre servicios (en términos de desarrollo, objetivos, disponibilidad, recursos, etc.).
- Robustez (capacidad del sistema de entrar en funcionamiento tan pronto como se despliegan el primer servicio y el directorio, que junto con la independencia entre servicios, permite mantener la operatividad ante caídas de los propios servicios).
- Condición de SPoF del componente directorio (que puede mitigarse incorporando varios directorios, introduciendo el concepto de catálogos de servicios personalizados para diferentes clientes).

3.5. Jugando a líder-trabajadores

Las instrucciones ofrecidas para el juego correspondiente a la arquitectura líder-trabajadores se detallan en el cuadro 5. Los estudiantes protagonizan la evolución dinámica durante su funcionamiento de sistemas con esta arquitectura. Se esperan las siguientes experiencias como resultado:

- Independencia entre trabajadores (en términos de disponibilidad, recursos).
- Robustez (capacidad del sistema de mantenerse operativo ante caídas de trabajadores).
- Escalabilidad (capacidad del sistema de reaccionar elásticamente a la demanda, arrancando más trabajadores).
- Optimización del consumo de recursos (capacidad de reaccionar elásticamente a la demanda, deteniendo los trabajadores ociosos).
- Condición de SPoF del componente líder.

3.6. Jugando a *peer-to-peer*

Por último, el cuadro 6 incluye las instrucciones proporcionadas para el juego correspondiente a la arquitectura *peer-to-peer*. Dependiendo del número de estudiantes, pueden configurarse uno o varios sistemas; parte de la clase integrará el sistema como *peer*s, mientras que el resto actuarán como clientes. Se esperan las siguientes experiencias:

- Necesidad de que cada *peer* tome decisiones tanto de gestión como de lógica de negocio.

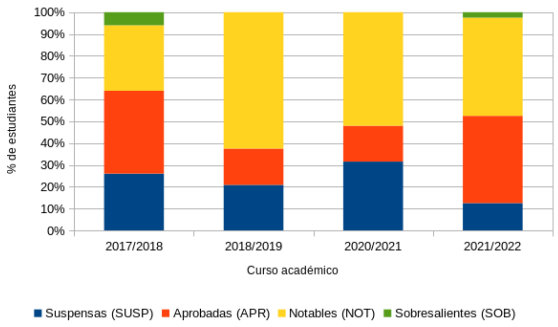

Figura 2: Comparativa de resultados académicos

- Necesidad de gestionar la vida de las peticiones, para asegurar que los clientes reciben una respuesta incluso bajo condiciones de alta carga para el sistema (muchos más clientes que *peer*s).
- Dificultades para implementar medidas de seguridad (por ejemplo, para diferenciar *peer*s maliciosos de errores de funcionamiento o caídas).
- Ausencia de coordinación (e impacto en la escalabilidad y la gestión descentralizada de la carga).
- Ausencia de SPoFs: disponibilidad máxima.

4. Evaluación

Dado que los juegos en sí mismos no constituyeron elementos puntuables para el estudiantado, nuestra evaluación preliminar se basa, por el momento, en las notas finales obtenidas por el alumnado del curso 21/22, primero en el que se incorporan los juegos de rol a las metodologías docentes de la materia, toda vez que el curso 22/23 se encuentra todavía en desarrollo en el momento de escribir este artículo. La figura 2 muestra una comparativa de resultados que muestra varios cursos anteriores, para favorecer el contraste de datos.

Percibimos un aumento de las calificaciones más altas (SOB), pero sobre todo de aprobados (APR) en el curso 21/22 en comparación con los años anteriores. Incluso en comparación con el curso 17/18, en el que se registran más notas de las más elevadas (SOB), el porcentaje de suspensos es significativamente menor. Por su parte, el curso 18/19 es significativamente mejor en relación con el número de notables (NOT), dato que relacionamos con el número de personas matriculadas ese año (habitualmente en el rango de las 60 personas, en el curso 18/19 fueron solo 27). La ratio reducida es una causa probable de ese mejor rendimiento [17].

Por su parte, hemos omitido de la comparativa el curso 19/20, el directamente afectado por la pandemia de COVID-19, por considerar que las circunstancias extraordinarias en las que se vieron envueltas tanto las metodologías docentes como de evaluación, convierten los datos de ese año en un claro *outlier*.

En resumen, el impacto cuantitativo percibido tras un curso académico de aplicación de la metodología docente propuesta parece positivo en relación con el número de personas que superan la materia, aunque no tanto en relación con la consecución de un rendimiento excelente (objetivo para el que la reducción de ratio parece más efectiva). No obstante, dado el limitado alcance de este análisis, circunscrito a los datos de un único año, será necesaria la reevaluación en años sucesivos para confirmar o puntualizar este análisis preliminar.

5. Discusión

La investigación ha demostrado la efectividad del uso de proyectos en los que el estudiantado ejercita diferentes roles como estrategia docente [35]. Cuando dichas estrategias involucran juegos de rol, concurren tres factores que favorecen la participación y el aprendizaje: experiencias personales diferenciadas, profundización en los contenidos, y mejora de las habilidades cooperativas [30]. En efecto, de acuerdo con nuestra percepción docente, la participación del estudiantado durante las clases en las que se llevan a cabo los juegos de rol es notablemente superior a los niveles habituales en ese tipo de sesiones, y no solamente por los motivos obvios que supone participar en los juegos propuestos, sino en términos de sus reflexiones, puestas en común, y preguntas realizadas sobre la materia involucrada.

De hecho, juzgando no solo por la cantidad, sino por la profundidad y las implicaciones de sus comentarios, preguntas y observaciones, podemos decir que esta nueva actividad docente tiene un impacto positivo también en la comprensión e interiorización de conceptos por parte del alumnado. También hemos percibido, aunque resulta más difícil de evaluar comparativamente, una mejora cualitativa en los trabajos individuales realizados durante el curso. Estos trabajos son tradicionalmente propuestas voluntarias que implican la búsqueda de ejemplos documentados de sistemas reales para cada modelo arquitectónico estudiado, y la realización de un comentario crítico al respecto. El hecho de que el contenido de estas entregas voluntarias haya sido percibido como de mejor calidad por el profesorado podría respaldar la intuición anterior en relación con una mejor comprensión de los conceptos arquitectónicos y una mayor confianza en los conocimientos adquiridos por parte del propio alumnado.

Aunque podríamos haber otorgado una puntuación a los juegos de rol propuestos, la evidencia parece aconsejar en sentido contrario [27]. En lo que respecta al *role-playing*, se han encontrado evidencias de su capacidad de mejorar las experiencias dentro del aula y la adquisición de conocimientos [1], incluso de correlación entre alta puntuación en estos ejercicios con altas cualificaciones en pruebas finales [29], pero no encontramos trabajos que evalúen el propio impacto en la experiencia o rendimiento del alumnado de otorgar puntuación a este tipo de oportunidades de aprendizaje.

Roles	Instrucciones
Eres el filtro de entrada	Recibes peticiones del cliente (docente) y debes transmitirlas al siguiente filtro después de procesarlas. Puedes aceptar una nueva solicitud inmediatamente después de terminar de procesarla y transmitir la anterior.
Eres un filtro intermedio	Admite solicitudes solamente de tu filtro previo, y pásalas al siguiente después de procesarlas. Admite cada nueva petición solo tras procesar y transmitir la anterior.
Eres el filtro de salida	Acepta peticiones solo de tu filtro anterior. Procesa la petición y envía/vuelca tu resultado a la salida esperada. Admite cada nueva petición tras terminar el procesado de la anterior.

Cuadro 2: Roles en juego en la arquitectura en *pipeline*

Roles	Instrucciones
Eres un productor	Genera datos de acuerdo con tu programación (varias temporizaciones posibles), y envíalos al componente repositorio en cuanto los tengas.
Eres un consumidor	Consulta el repositorio de acuerdo con tu programación (varias temporizaciones posibles, incluso dependiendo de consultas anteriores) y realiza tu función (que no puede generar datos, pero sí generar más necesidades de consulta, o alterarlas) según los resultados de la consulta (aparición o no de información nueva, etc.).
Eres un productor/consumidor	De acuerdo con tu programación (varias temporizaciones posibles, incluso dependiendo de consultas o datos generados previos), genera datos y envíalos al repositorio, o consúltalo y reacciona como te corresponda (generando más datos o más consultas).

Cuadro 3: Roles en juego en la arquitectura en repositorio

Roles	Instrucciones
Eres un cliente	Envía peticiones a los servicios ofrecidos por el directorio. Puedes enviar una o varias peticiones, al mismo servicio o a servicios diferentes, en el momento que quieras.
Eres el directorio	Recibe las peticiones de los clientes y reenvíalas a alguna de las instancias del servicio solicitado. Atiende cada petición lo más rápido que puedas, momento en el que puedes procesar la siguiente.
Eres un servicio	Recibe las peticiones de los clientes que te haga llegar el directorio, procésalas y genera una respuesta. No atiendas peticiones directas de los clientes. Atiende una nueva petición solo después de procesar la anterior, y enviar la respuesta al cliente correspondiente.

Cuadro 4: Roles en juego en la arquitectura cliente-servidor

Roles	Instrucciones
Eres un cliente	Envía peticiones al sistema a través de su líder. Puedes enviar una o varias peticiones, en cualquier momento que consideres.
Eres el líder	Recibe las peticiones de los clientes y elige uno de los trabajadores para que la atienda. Decide cuál según tu programación (aleatoriamente, round-robin, etc.). Puedes incorporar un nuevo trabajador al sistema si todos los existentes están ocupados. Tan pronto como redirijas una petición, atiende la siguiente. Cada cierto tiempo revisa si hay trabajadores ociosos y decide si quieres liberarlos. Puedes enviar la misma petición a varios trabajadores.
Eres un trabajador	Recibe las peticiones de los clientes que te haga llegar el líder, procésalas y genera una respuesta. No atiendas peticiones directas de los clientes. Atiende una petición nueva solo después de procesar la anterior, y enviar la respuesta al cliente correspondiente.

Cuadro 5: Roles en juego en la arquitectura líder-trabajadores

Roles	Instrucciones
Eres un cliente	Envía peticiones al sistema, a través de cualquiera de los *peer*. Puedes enviar una o varias peticiones, en cualquier momento que consideres.
Eres un peer	Recibe peticiones, que pueden venir de los clientes o de otros *peer*s. En cada caso, si tienes la capacidad o los datos necesarios, procesa la petición y envía una respuesta al cliente, o bien reenvíala a uno o varios de tus *peer*s vecinos. Atiende una nueva petición solo después de procesar la anterior y enviar la respuesta correspondiente. Cada cierto tiempo revisa tu *vecinanza* y decide si mantienes, eliminas o buscas nuevos vecinos.

Cuadro 6: Roles en juego en la arquitectura *peer-to-peer*

Ahora bien, algunas investigaciones alertan de que, si bien muchos estudiantes perciben que los juegos de rol, en el contexto de su aprendizaje, refuerzan su interés y resultados, esta opinión no es generalizable [30]. Aunque no hemos encontrado ningún trabajo sobre esta hipótesis en concreto, esta diferencia de opiniones podría estar relacionada precisamente con los diferentes perfiles de aprendizaje. Al mismo tiempo, sería un resultado coherente con nuestra evaluación cuantitativa: la mejora podría no producirse tanto en el rango de personas con estilos de aprendizaje que se benefician de metodologías docentes más tradicionales, sino en el de personas con estilos de aprendizaje menos atendidos (que cabría esperar obtengan peores resultados), consiguiendo que más personas superen la materia.

En todo caso, identificamos dos posibles líneas de investigación necesarias para continuar la evaluación de la eficacia y efectividad de las actividades de juego de rol como estrategia docente cuando la intención es proporcionar una mayor atención a los perfiles de aprendizaje que menos se benefician del resto de metodologías docentes más tradicionalmente presentes en el contexto universitario. Por un lado, dado que las experiencias de aprendizaje experimental que abordan todos los estilos de aprendizaje no se pueden considerar práctica habitual, cabría suponer que los estudiantes, especialmente en el momento en que alcanzan el nivel universitario, han desarrollado cierto grado de adaptación a la falta de métodos y actividades que mejor resuenan con ellos, en el caso de los perfiles de aprendizaje menos abordados. De confirmarse esta hipótesis, podría explicar el impacto relativamente modesto de esfuerzos innovadores como el aquí propuesto. Por otro lado, dado que no sabemos cómo se distribuyen los diferentes perfiles de aprendizaje entre los estudiantes en general, y tampoco entre los estudiantes universitarios, desconocemos si constituyen un porcentaje significativo cuando se consideran como parte de una clase de 60-80 personas. La cuantificación de esta distribución sería útil para anticipar de manera más precisa el impacto en el rendimiento académico esperable tras la incorporación de propuestas como la nuestra.

6. Conclusiones

En este trabajo hemos querido ampliar las metodologías docentes empleadas en Arquitectura del Software (AS). El objetivo ha sido proporcionar un entorno de aprendizaje más justo, teniendo en cuenta los diferentes perfiles de aprendizaje de las personas, de manera que cada individuo tenga oportunidades de conectar con el tipo de experiencias de aprendizaje que le resultan más significativas y efectivas.

Dado que las metodologías docentes más tradicionales no se alinean con estilos de aprendizaje *acomodaticios* o *divergentes* (en la nomenclatura de Kolb), hemos abordado el problema incorporando juegos de rol de un modo innovador no solo en el contexto de la AS en particular, sino de la Ingeniería del Software en general: hemos diseñado un conjunto de escenarios en los que los estudiantes adoptan el papel de componentes software en diferentes modelos arquitectónicos. A través de estos juegos, proporcionamos experiencias kinestésicas de la estructura, ventajas y desventajas de un conjunto de arquitecturas de referencia.

De nuestra experiencia concluimos que es posible diseñar escenarios jugables para todas las arquitecturas contempladas, y que este tipo de propuestas representan una oportunidad de aprendizaje que favorece la interacción y participación del alumnado. Además, nuestra evaluación preliminar indica un impacto positivo en el número de personas que superan la materia.

Referencias

[1] Harneel Acharya, Rakesh Reddy, Ahmed Hussein, Jaspreet Bagga y Timothy Pettit. The effectiveness of applied learning: an empirical evaluation using role playing in the classroom. *Journal of Research in Innovative Teaching & Learning*, 12:295–310, 2019.

[2] Steven K. Andrianoff y David B. Levine. Role playing in an object-oriented world. *SIGCSE Bulletin*, 34(1):121–125, 2002.

[3] Paolo Bucci, Timothy J. Long y Bruce W. Weide. Teaching sw architecture principles in cs1/cs2. En *Procs. of the Intl. Workshop on Sw Architecture*, p. 9–12, 1998.

[4] Andrew Cain y Muhammad Ali Babar. Reflections on applying constructive alignment with formative feedback for teaching intro programming and software architecture. En *Procs. of the Intl. Conference on Sw Engineering*, p. 336–345, 2016.

[5] Rafael Capilla, Olaf Zimmermann, Carlos Carrillo y Hernán Astudillo. Teaching students software architecture decision making. En *Procs. of the European Conference on Sw Architecture*, pp. 231–246, 2020.

[6] Cristóbal Costa-Soria y Jennifer Pérez. Teaching sw architectures and aspect-oriented sw development using OOS. *ACM SIGCSE Bulletin*, 41(3):385, 2009.

[7] Remco C. de Boer, Rik Farenhorst y Hans van Vliet. A community of learners approach to software architecture education. En *Procs. of the Conference on Sw Engineering Education and Training*, pp. 190–197, 2009.

[8] Remo Ferrari, Nazim H. Madhavji y Mark Wilding. The impact of non-technical factors on software architecture. En *Procs. of the Workshop on Leadership and Management in Sw Architecture*, pp. 32–36, 2009.

[9] B. Ricardo Gacitúa, Mauricio Diéguez y Elizabeth Vidal. Forming sw architects in early stages: From craft to engineering. En *Procs. of the Intl. Conference of the Chilean Computer Science Society*, pp. 1–8, 2017.

[10] Vashti C. Galpin, Ian D. Sanders y Pei-yu Chen. Learning styles and personality types of computer science students at a south african university. En *Procs. of the Conference on Innovation and Technology in Computer Science Education*, p. 201–205, 2007.

[11] Matthias Galster y Samuil Angelov. What makes teaching software architecture difficult? En *Procs. of the Intl. Conference on Sw Engineering*, p. 356–359, 2016.

[12] Kirti Garg y Vasudeva Varma. An effective learning environment for teaching problem solving in software architecture. En *Procs. of the India Sw Engineering Conference*, p. 139–140, 2009.

[13] Fáber D. Giraldo, Sergio F. Ochoa, Myriam Herrera, Andrés Neyem, José Luis Arciniegas, Clifton Clunie, Sergio Zapata y Fulvio Lizano. Applying a distributed cscl activity for teaching software architecture. En *Procs. of the Intl. Conference on Information Society*, pp. 208–214, 2011.

[14] Richard A. Howard, Curtis A. Carver y William D. Lane. Felder's learning styles, bloom's taxonomy, and the kolb learning cycle: Tying it all together in cs2. En *Procs. of the Technical Symposium on Computer Science Education*, p. 227–231, 1996.

[15] James S. Jones. Participatory teaching methods in cs. En *Procs. of the Technical Symposium on Computer Science Education*, p. 155–160, 1987.

[16] Patricia Lago y Hans van Vliet. Teaching a course on sw architecture. En *Procs. of the Conference on Sw Engineering Education and Training*, pp. 35–42, 2005.

[17] Douglas J. Lamdin. Evidence of student attendance as an independent variable in education production functions. *J. Educational Research*, 89(3):155 – 162, 1996.

[18] Sonsoles López-Pernas, Mohammed Saqr y Olga Viberg. Combining learning analytics methods and data sources to understand students' approaches to learning programming. *Sustainability*, 13(9), 2021.

[19] Tomi Mannisto, Juha Savolainen y Varvana Myllarniemi. Teaching software architecture design. En *Procs. of the Working IFIP Conference on Sw Architecture*, p. 117–124, 2008.

[20] Chris Manolis, David J. Burns, Rashmi Assudani y Ravi Chinta. Assessing experiential learning styles: A methodological reconstruction and validation of the kolb learning style inventory. *Learning and Individual Differences*, 23:44–52, 2013.

[21] John D. McGregor, Felix Bachman, Len Bass, Philip Bianco y Mark Klein. Using an architecture reasoning tool to teach software architecture. En *Procs. of the Conference on Sw Engineering Education Training*, pp. 275–282, 2007.

[22] Saul A. McLeod. Kolb - learning styles and experiential learning cycle, 2017. Disponible en `https://www.simplypsychology.org/learning-kolb.html`.

[23] Claudia Hidalgo Montenegro, Hernán Astudillo y María Clara Gómez Álvarez. Atam-rpg: A role-playing game to teach architecture trade-off analysis method (atam). En *Procs. of the Latin American Computer Conference*, pp. 1–9, 2017.

[24] Lennart Ohlsson y Christian Johansson. A practice driven approach to software engineering education. *IEEE Transactions on Education*, 38(3):291–295, 1995.

[25] Claudia S.C. Rodrigues y Cláudia M.L. Werner. Making the comprehension of software architecture attractive. En *Procs. of the Conference on Sw Engineering Education and Training*, pp. 416–420, 2011.

[26] Chandan R. Rupakheti y Stephen V. Chenoweth. Teaching software architecture to undergraduate students: An experience report. En *Procs. of the Intl. Conference on Sw Engineering*, volumen 2, pp. 445–454, 2015.

[27] Jeffrey Schinske y Kimberly Tanner. Teaching more by grading less (or differently). *CBE Life Sciences Education*, 13(2):159 – 166, 2014.

[28] Julie E. Sharp, John N. Harb y Ronald E. Terry. Combining kolb learning styles and writing to learn in engineering classes. *Journal of Engineering Education*, 86(2):93–101, 1997.

[29] Richard Berntsson Svensson y Björn Regnell Svensson. Is role playing in requirements engineering education increasing learning outcome? *Requirements Engineering*, 22:475–489, 2017.

[30] David Toth y Mary Kayler. Integrating role-playing games into computer science courses as a pedagogical tool. En *Procs. of the Technical Symposium on Computer Science Education*, p. 386–391, 2015.

[31] Juan Sebastián Urrego y Darío Correal. Archinotes: A tool for assisting software architecture courses. En *Procs. of the Intl. Conference on Sw Engineering Education and Training*, pp. 80–88, 2013.

[32] Arie Van Deursen, Maurício Aniche, Joop Aué, Rogier Slag, Michael De Jong, Alex Nederlof y Eric Bouwers. A collaborative approach to teaching software architecture. En *Procs. of the Technical Symposium on Computer Science Education*, p. 591–596, 2017.

[33] Andrea Vázquez-Ingelmo, Alicia García-Holgado y Francisco J. García-Peñalvo. C4 model in a software engineering subject to ease the comprehension of uml and the software. En *Procs. of the Global Engineering Education Conference*, pp. 919–924, 2020.

[34] Alf Inge Wang. Extensive evaluation of using a game project in a software architecture course. *ACM Transactions on Computing Education*, 11(1), 2011.

[35] Bruno Warin, Omar Talbi, Christophe Kolski y Frédéric Hoogstoel. Multi-role project (mrp): A new project-based learning method for stem. *IEEE Transactions on Education*, 59(2):137–146, 2016.

[36] Gero Wedemann. Scrum as a method of teaching software architecture. En *Procs. of the European Conference of Sw Engineering Education*, p. 108–112, 2018.

[37] Bian Wu, Jan-Erik Strom, Alf Inge Wang y Trond Blomholm Kvamme. Xquest used in software architecture education. En *Procs. of the Intl. Consumer Electronics Society's Games Innovations Conference*, pp. 70–77, 2009.

[38] Li Zhang, Yanxu Li y Ning Ge. Exploration on theoretical and practical projects of software architecture course. En *Procs. of the Intl. Conference on Computer Science Education*, pp. 391–395, 2020.

[39] D. Zowghi y S. Paryani. Teaching requirements engineering through role playing: lessons learnt. En *Procs. of the Intl. Requirements Engineering Conference*, pp. 233–241, 2003.

[40] Sara Zuppiroli, Paolo Ciancarini y Maurizio Gabbrielli. A role-playing game for a software engineering lab. En *Procs. of the Conference on Sw Engineering Education and Training*, pp. 13–22, 2012.