Explorando la calidad en uso de sistemas CSCL para el aprendizaje de la Programación: el caso de COLLECE 2.0

Rafael Duque[1], Miguel Ángel Redondo[2], Manuel Ortega[2], Sergio Salomón[3], Ana Isabel Molina[2]

[1]Departamento de Matemáticas, Estadística y Computación (Universidad de Cantabria)
Avenida de los Castros S/N, 39005, Santander, España
duquemr@unican.es

[2]Departamento de Tecnologías y Sistemas de Información (Universidad de Castilla-La Mancha)
Paseo de la Universidad 4, 13071, Ciudad Real, España
{miguel.redondo, manuel.ortega, anaisabel.molina}@uclm.es

[3] Departamento de Inteligencia Analítica (Decide Soluciones)
Calle de Albasanz 16, 28037, Madrid, España
sergio.salomon@decidesoluciones.es

Resumen: Los sistemas CSCL (Computer-Supported Collaborative Learning) pueden ser especialmente útiles para facilitar la enseñanza y el aprendizaje de la programación, ya que reproducen el contexto profesional de trabajo en equipo. La calidad de la experiencia del aprendizaje con sistemas CSCL depende de varios factores, como, por ejemplo, el diseño y la implementación del sistema, la adecuación de las tareas planteadas a los estudiantes, la configuración del grupo de trabajo y la tutorización recibida. Este trabajo aborda el problema de medir la calidad en uso de los sistemas CSCL que dan soporte al proceso de aprendizaje de la programación. Para ello se propone un conjunto de medidas de calidad en uso y se analiza cómo calcularlas automáticamente utilizando el soporte de FAQuiS (Framework for Assessing Quality-in-use of Software), un framework basado en modelos para evaluar las características y subcaracterísticas de calidad en uso recogidas en la norma ISO 25010:2011. El artículo incluye un estudio de la aplicabilidad de esta propuesta en COLLECE 2.0, un sistema CSCL distribuido síncrono para el aprendizaje de la programación.
Palabras clave: programación de computadores, calidad en uso del software, aprendizaje colaborativo soportado por computador.

Abstract: CSCL (Computer-Supported Collaborative Learning) systems can be especially useful to facilitate the teaching and learning of programming, since they reproduce the professional context of teamwork. The quality of the learning experience with CSCL systems depends on several factors, such as, for example, the design and implementation of the system, the appropriateness of the tasks set for students, the configuration of the work group and the tutoring received. This work addresses the problem of measuring the quality in use of CSCL systems that support the programming learning process. To this end, a set of quality in use measures is proposed and how to calculate them automatically is analyzed using the support of FAQuiS (Framework for Assessing Quality-in-use of Software), a framework based on models to evaluate quality in use characteristics and subcharacteristics included in the ISO 25010:2011 standard. The article includes a study of the applicability of this proposal in COLLECE 2.0, a synchronous distributed CSCL system for programming learning.
Key words: computer programming, software quality in use, computer-supported collaborative learning.

1. Introducción

La programación de computadores implica la generación de instrucciones que la máquina puede procesar para realizar tareas específicas. La actual era digital hace que la demanda de trabajadores con formación en el ámbito de la programación de

computadores sea cada vez mayor. En un entorno profesional, es común que se trabaje de forma colaborativa para desarrollar programas que cumplan un conjunto de requisitos específicos. Los sistemas CSCL (Computer-Supported Collaborative Learning) son entornos de aprendizaje que utilizan tecnología informática para apoyar la colaboración entre estudiantes en actividades educativas. Para facilitar los procesos de enseñanza/aprendizaje de la programación de computadores, los sistemas CSCL pueden considerarse un instrumento especialmente útil en la medida que reproducen el contexto profesional en el que varios programadores participan en un mismo proceso de trabajo (Silva et al., 2020).

Los sistemas CSCL para el aprendizaje de la programación proporcionan a los estudiantes un entorno de aprendizaje interactivo que les permite trabajar en equipo en tiempo real, independientemente de su ubicación física, compartir conocimientos y recibir retroalimentación de sus compañeros y profesores. Estos sistemas pueden ofrecer variedad de recursos y herramientas como, por ejemplo, tutoriales, ejemplos de código fuente y editores compartidos. Además, los mencionados recursos y herramientas facilitan que la enseñanza/aprendizaje de la programación pueda llevarse a cabo siguiendo un paradigma basado en problemas (Dolog et al., 2016). En este paradigma los alumnos trabajan en equipo para abordar un problema que implica identificar una solución que luego deben implementar mediante la escritura del código fuente y verificarla mediante la ejecución del programa.

La calidad de la experiencia del aprendizaje con sistemas CSCL puede variar ampliamente, dependiendo de factores como el diseño y la implementación del sistema, la adecuación de las tareas propuestas a los estudiantes, la configuración del grupo de trabajo, dificultades para orquestar la colaboración o la tutorización para construir la solución. En este punto surge el reto de establecer mediciones estandarizadas de la experiencia de los alumnos como usuarios de sistemas CSCL para el aprendizaje de la programación. Este trabajo aborda este reto a través de una propuesta basada en el framework FAQuiS (Framework for Assessing Quality-in-use of Software) (Salomón et al., 2022) y en la norma ISO 25010:2011 (ISO/IEC 25010, 2011) que introduce el concepto de calidad en uso como el grado en que un producto o sistema puede ser utilizado por usuarios específicos para satisfacer sus necesidades y lograr objetivos específicos con efectividad, eficiencia, sin riesgos y con satisfacción en contextos específicos de uso. La norma ISO 25010:2011 define un modelo de calidad en uso con un conjunto de características y subcaracterísticas del software que facilitan un marco genérico de evaluación. FAQuiS integra un soporte computacional para calcular un conjunto de medidas de la calidad en uso que cuantifiquen las características y sub-características de la norma ISO 25010:2011. El objetivo principal de este trabajo es estudiar la factibilidad de usar el soporte de FAQuiS para medir la calidad en uso de sistemas CSCL que dan soporte al aprendizaje de la programación. El artículo incluye un estudio de la aplicabilidad de este modelo de evaluación de la calidad en uso utilizando COLLECE 2.0 (Lacave et al., 2019), un sistema CSCL distribuido síncrono para el aprendizaje de la programación.

El artículo incluye 4 secciones adicionales. La Sección 2 estudia una serie de trabajos existentes en la comunidad científica en el ámbito de la evaluación de procesos de aprendizaje de la Programación soportada por computador. La Sección 3 presenta nuestra propuesta para medir las características y sub-características de la norma ISO 25010:2011 en sistemas CSCL de soporte a la programación. La Sección 4 describe un caso de estudio en el que se estudiará la aplicabilidad de la propuesta al sistema COLLECE 2.0. La Sección 5 analiza las conclusiones del trabajo realizado y las nuevas líneas de investigación que se acometerán en el futuro.

2. Trabajos relacionados

El análisis de los datos de interacción del usuario de sistemas software permite extraer patrones relevantes de su actividad (van Der Aalst, 2012). Esto se extiende incluso a entornos colaborativos de múltiples usuarios que trabajan juntos para alcanzar objetivos comunes, en los cuales la interacción del grupo proporciona un conocimiento adicional relevante (Salomón et al. 2019). Para estos casos, los modelos de comportamiento y actividad presentan una herramienta útil con la que capturar patrones y tendencias de usuarios o grupos y permitir la predicción de acciones futuras, entre otros usos. En el ámbito del aprendizaje encontramos ejemplos en los

que el análisis de los datos de interacción de los alumnos se ha utilizado para establecer procesos de enseñanza personalizados (Gavriushenko el at. 2017).

En las actividades de programación, un estudiante puede generar durante el proceso de codificación una gran cantidad de datos a partir de la interacción que realiza o los cambios que aplica. Múltiples trabajos en el estado del arte muestran esto, llevando a cabo diversas estrategias de recolección de datos: edición de código (Piech et al. 2012), información de ejecución (Leinonen et al. 2017), compilación (Brown et al. 2014) o métricas de calidad del código (Pettit et al. 2015). Estos datos permiten estudiar y entender cómo el estudiante se comporta, las dificultades a las que se enfrenta, la utilidad de las herramientas que emplea o si realiza trampas (Hui & Farvolden, 2017). Spacco et al. (2015) exponen mediante técnicas de análisis de datos las correlaciones existentes entre datos de comportamiento y el proceso de aprendizaje. Este tipo de información resulta significativa en la creación de herramientas de evaluación y soporte al estudiante en la programación.

Específicamente en casos de generación automática de ayuda o "*feedback"* es donde estos datos pueden tener un papel clave. En la literatura actual, se reconoce la gran importancia de una retroalimentación de alta calidad para el aprendizaje de los alumnos (Ott et al. 2016). Sin embargo, la mayoría de enfoques existentes se basa en consejos predefinidos o enriquecimiento de los mensajes de ayuda proporcionados en la fase de compilación (Becker et al. 2016), careciendo de personalización del *feedback* individual a cada estudiante. Cicirello (2009) ha trabajado en optimizar el *feedback* individual, pero requiriendo para ello información adicional proporcionada explícitamente por el estudiante. Otros enfoques utilizando técnicas inteligentes se apoyan principalmente en el histórico de soluciones pasadas, sin adaptarse al comportamiento individual. Gross et al. (2014) ofrecen *feedback* basado en ejemplos comparando soluciones de expertos con las de estudiantes. Rivers & Koedinger (2015) identifican automáticamente en soluciones previas la secuencia de cambios necesarios para alcanzar la solución y producen ayuda individual según los pasos que el estudiante debería seguir.

En cuanto a la retroalimentación potencial basada en las dificultades para orquestar el trabajo colaborativo, es importante destacar que se puede utilizar una amplia gama de criterios (nivel de coordinación, grado de comunicación, etc.). Algunas métricas que cuantifican aspectos relacionados con los comportamientos de los estudiantes (tiempo dedicado a la tarea, procrastinación, etc.), actitudes (motivación, autoeficacia, etc.) y respuestas fisiológicas (ansiedad, frustración, etc.) pueden calcularse automáticamente (Hundhausen et al., 2017). Los valores de estas métricas son útiles para identificar a los estudiantes que necesitan apoyo mediante intervenciones en sus procesos de aprendizaje y para mejorar la adaptabilidad de los entornos de aprendizaje asistidos por computador (Martínez-Monés et al., 2020). Sin embargo, Silva et al. (2020) han realizado una revisión sistemática del aprendizaje colaborativo asistido por computador en la enseñanza de programación, concluyendo que el 37% de los estudios en este campo no realizan ningún tipo de análisis de colaboración. Esto justifica la necesidad de realizar evaluaciones que, además, sean estandarizadas. Actualmente, no se siguen evaluaciones estandarizadas, lo que implica una falta de uniformidad en los métodos de evaluación y hace difícil comparar los resultados entre diferentes estudios y contextos. Para abordar esta problemática, se propone utilizar el modelo de calidad en uso ISO 25010:2011. Este modelo proporciona un marco estandarizado que permite evaluar la calidad de los sistemas y servicios informáticos en uso, asegurando que las evaluaciones sean consistentes y comparables. Al implementar evaluaciones estandarizadas basadas en el ISO 25010:2011, se espera mejorar la precisión en la identificación de las necesidades de los estudiantes y la eficacia de las intervenciones, así como la calidad general de los entornos de aprendizaje colaborativo asistidos por computador.

3. Medición de la calidad en uso basada en modelos y archivos de log

En un trabajo anterior (Salomón et al., 2022) se describió FAQuiS (Framework for Assessing Quality-in-use of Software), un marco de trabajo para calcular medidas de calidad en uso. FAQuiS no utiliza cuestionarios ni entrevistas con usuarios, pero permite complementar estos métodos con un soporte computacional que automatice la medición de calidad

en uso procesando archivos de log y los siguientes tres modelos computacionales: (i) modelo de tareas, (ii) modelo de contexto, (iii) modelo de usuario.

El **modelo de tareas** de FAQuiS (ver Figura 1) se basa en los siguientes conceptos:

- Tarea: Un proceso que permite al usuario alcanzar algún objetivo con el soporte del sistema. Las tareas se categorizan en cuatro tipos (Li et al., 2010): (i) tareas de usuario, son realizadas exclusivamente por el usuario sin interactuar con el sistema; (ii) tareas del sistema, realizadas por la propia aplicación y no requieren de la intervención directa del usuario; (iii) tareas interactivas, implican una participación activa por parte del usuario interactuando con el sistema; (iv) tareas abstractas, se descomponen en un conjunto de subtareas más pequeñas y específicas para facilitar su ejecución y seguimiento.
- Artefacto: Se refiere a los productos, resultados o salidas que los usuarios producen al realizar una tarea utilizando un sistema informático (por ejemplo, código fuente, resultado de compilaciones o ejecuciones).
- Acción del usuario: Unidad de interacción del usuario con el sistema, que se almacena en un repositorio de log. Cada acción se clasifica de esta manera (Duque et al., 2011): acción cognitiva, interactúa con un artefacto pero no altera su estado; acción comunicativa, permite el intercambio de mensajes entre usuarios (envío de mensajes a través de chat, foros, correo electrónico, etc.); acción instrumental, modifica un artefacto en construcción (por ejemplo, cambios en el código fuente); acción basada en protocolos, permite coordinar el proceso colaborativo sin establecer diálogo entre usuarios (solicitud de acceso a un editor compartido, votación sobre una propuesta, etc.).

Además, cada acción puede tener algún riesgo asociado (económico, sanitario, etc.) cuya frecuencia debe estimarse y cuantificarse cómo el sistema mitiga su impacto. Las acciones del usuario permiten al usuario interactuar con el sistema a través de un paradigma de interacción (computación ubicua, realidad aumentada/virtual, etc.).

El **modelo de contexto** (ver Figura 1) incluye información como la ubicación del usuario, las relaciones sociales que mantiene, si establece una colaboración síncrona o asíncrona. Por último, el modelo de contexto incluye una dimensión tecnológica que especifica el soporte software y hardware del que dispone el usuario.

El **modelo de usuario** (ver Figura 1) representa información sobre el perfil de la persona que interactúa con el sistema (rango de edad, género, nacionalidad, etc.), intereses en cierto tipo de tareas, rol (alumno, profesor, etc.) y otros rasgos que pueden influir en la interacción con el sistema (estilo de aprendizaje, conocimientos en relación a la programación, etc.). Además, se establece una especificación de las habilidades técnicas e idiomáticas del usuario.

El archivo de log es un repositorio de acciones que ejecuta el usuario o el sistema. Este archivo incluye un identificador de las acciones recogidas en el modelo de tareas que se ejecutan, quién las realiza, cuándo se llevan a cabo y el espacio del sistema que soportan dichas acciones. Este espacio puede ser cualquier elemento de interfaz de usuario definido en el sistema.

FAQuiS genera un conjunto de mediciones asociadas a cada una de las características y sub-características de la norma ISO 25010:2011. Esta norma define concretamente las siguientes cinco características del software asociadas a la calidad en uso: efectividad, eficiencia, mitigación de riesgos, satisfacción y cobertura de contexto.

La característica de efectividad se relaciona, según la norma ISO 25010:2011, con la precisión y completitud con la que los usuarios alcanzan los objetivos especificados. Por tanto, se mide mediante el número de tareas finalizadas exitosamente, la calidad de los resultados de las acciones instrumentales ejecutadas y la concordancia entre el comportamiento de cada usuario y las secuencias de acciones especificadas en el modelo de tareas para alcanzar dichos objetivos.

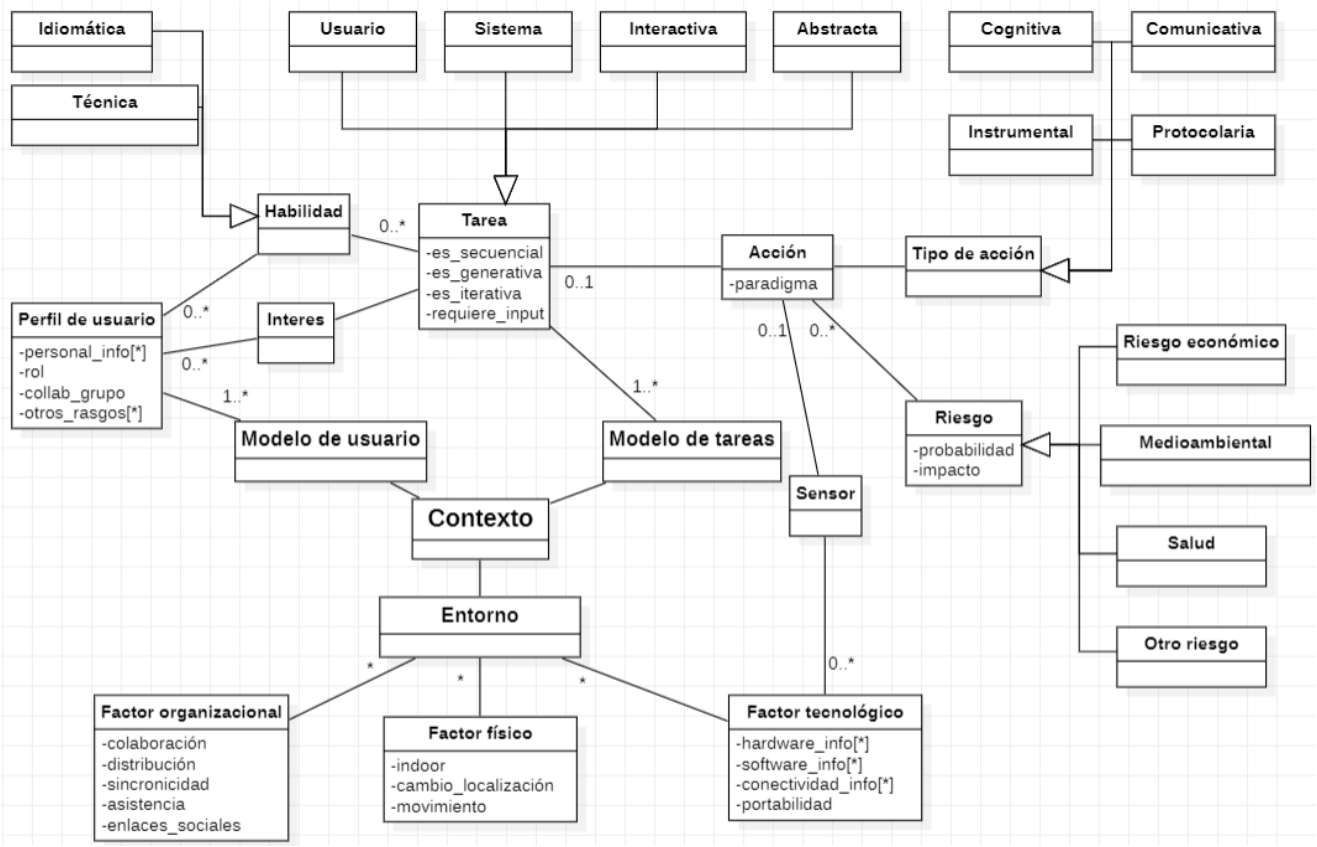

Figura 1. Metamodelo de FAQuiS

Según la norma ISO 25010:2011, la eficiencia trata de los recursos empleados para alcanzar objetivos. La medición de recursos se realiza cuantificando la cantidad de tiempo para completar las tareas, así como el número de acciones y espacios empleados.

La característica de mitigación de riesgos se define en la norma ISO 25010:2011 como el grado en el cual un sistema mitiga el riesgo potencial para el estatus económico, la vida humana, la salud o el medioambiente. Las medidas de calidad en uso asociadas a esta característica cuantifican las respuestas del sistema para mitigarlos.

La característica de satisfacción se define como el grado en el cual se satisfacen las necesidades del usuario al emplear un sistema en un contexto de uso específico. Esta característica se representa en el modelo de calidad ISO 25010:2011 mediante las siguientes sub-características:

- Utilidad. Es el grado en que un usuario está satisfecho al percibir que logra sus objetivos de forma pragmática, lo cual incluye los resultados y las consecuencias del uso del sistema. Las medidas asociadas evalúan en qué medida el usuario encuentra útil las acciones y espacios disponibles en el sistema para alcanzar sus objetivos.
- Confianza. Es el grado en el que un usuario, u otro stakeholder, tiene confianza en que un producto o sistema se comportará como se espera que lo haga. Estas medidas evalúan que el usuario ejecuta acciones asociadas a los riesgos y respuestas de otros colaboradores ya que confía en una respuesta satisfactoria por parte del sistema y de los otros participantes.
- Placer. Es el grado en que el usuario siente una experiencia placentera al satisfacer sus requisitos. Las medidas de esta sub-característica evalúan si la persona adquiere nuevas capacidades respecto a las establecidas inicialmente en el modelo de usuario tras el uso del sistema en distintas sesiones de trabajo.

- Comodidad: Grado en que el usuario está satisfecho con la comodidad física del dispositivo. Estas medidas evalúan la densidad de trabajo de cada espacio y la utilización de paradigmas de interacción basados en acciones implícitas y Realidad Aumentada/Virtual que pueden resultar más cómodos al usuario.

La cobertura del contexto define el grado en el cual un sistema se puede usar cumpliendo el resto de las características (efectividad, eficiencia, mitigación de riesgos y satisfacción) en relación al contexto de uso. La norma 25010:2011 define dos sub-características para la cobertura del contexto: cobertura y flexibilidad. La cobertura implica que la calidad en uso se evalúa en un conjunto de contextos de uso que estaban previstos. La flexibilidad implica que el sistema se utiliza por parte de los usuarios en contextos que inicialmente no estaban contemplados.

4. Caso de estudio: COLLECE 2.0

COLLECE 2.0 (COLLaborative Edition, Compilation and Execution of programs) es un plugin de Eclipse para la programación en grupo, que cuenta con una interfaz de usuario personalizable (Lacave et al., 2019). Esta interfaz (ver Figura 2) incluye un árbol de archivos del proyecto, un panel de usuarios conectados, tele-cursores para identificar quién está editando y en qué parte del código lo está haciendo; un editor de código compartido; funcionalidades para el bloqueo de regiones del código para que un alumno pueda prohibir modificaciones de un fragmento de código a otros compañeros; un panel de control de regiones bloqueadas para dar a conocer qué código quedó bloqueado y quién los restringió; chat; y, el enunciado del problema a resolver. Todos estos elementos están diseñados para permitir la colaboración distribuida síncrona de los alumnos para la resolución de problemas en el ámbito de la programación de computadores. Además, COLLECE 2.0 utiliza sistemas de control de versiones para mantener el estado persistente de los proyectos de código asociados a las sesiones.

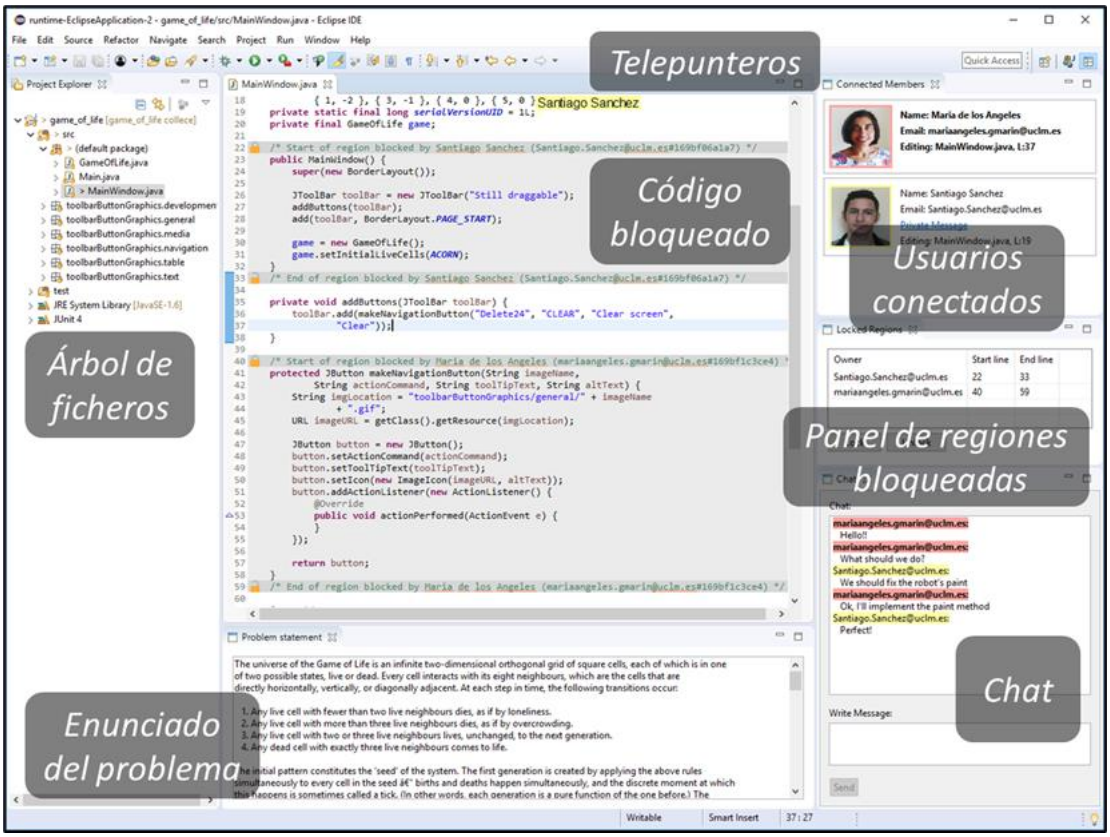

Figura 2. Interfaz de usuario principal de COLLECE 2.0

COLLECE 2.0 también dispone de un espacio que se apoya en el paradigma de Realidad Aumenta (RA) en el que los alumnos pueden visualizar el comportamiento del programa que construyen mediante la notación ANGELA (notAtioN of road siGns to facilitatE the Learning of progrAmming), basada en una metáfora de carreteras y señales de tráfico representadas por gráficos en 3D (Schez-Sobrino et al., 2020). Estas representaciones permiten visualizar de manera intuitiva el flujo de ejecución de un programa, ya que los alumnos están familiarizados con estas carreteras y señales en su vida cotidiana. Dichas visualizaciones gráficas pueden generarse automáticamente a partir del código fuente de los programas. La notación ANGELA permite la visualización tanto estática como dinámica de los algoritmos implementados. En el caso de la visualización estática, se busca facilitar la comprensión de las sentencias que componen el programa. Por otro lado, la visualización dinámica permite seguir la ejecución del programa, funcionando como un simulador de la traza del programa.

La Tabla 1 sintetiza cómo las acciones recogidas en archivo de log y el procesamiento de los tres modelos manejados por FAQuiS (modelo de tareas, modelo de contexto y modelo de usuario) permiten evaluar la efectividad, eficiencia y mitigación de riesgos de COLLECE 2.0 como instrumento de aprendizaje de la programación mediante un enfoque basado en problemas. La Tabla 1 también muestra algunas intervenciones para solucionar problemas identificados por las mediciones de efectividad, eficiencia y mitigación de riesgos.

La efectividad del proceso de resolución de problemas se mide a través del impacto que tienen las acciones instrumentales en el editor compartido ya que son las que permiten construir un artefacto que resuelva el problema planteado por el sistema, los resultados obtenidos en la consola tras ejecutar acciones de compilación y ejecución, y el grado de seguimiento de los patrones especificados en el modelo de tareas (ver Tabla 1). Las intervenciones para corregir deficiencias en la efectividad incluyen sugerencias para mejorar el código, invitar a compilar y ejecutar el programa, rediseñar el sistema para adaptarlo a los patrones de comportamiento de los usuarios.

Características y sub-características del ISO 25010:2011	**Descripción de las medidas propuestas en FAQuiS**	**Fuente de información**	**Mecanismos de intervención**
Efectividad	Porcentaje de problemas resueltos satisfactoriamente.	Editor y consola	Sugerencias de mejora en el código fuente
	Número de artefactos generados satisfactoriamente durante el proceso de trabajo.		Sugerencias para compilar y ejecutar el código
	Similitud entre los patrones de interacción del usuario y los del modelo de tareas.	Todos los espacios del sistema y modelo de tareas	Rediseño del sistema para adaptarlo a los patrones de comportamiento del usuario
Eficiencia	Número de espacios usados.	Todos los espacios del sistema	Adaptar la dificultad del problema.
	Tiempo para completar tareas.		
	Número de acciones ejecutadas.		
	Acciones ejecutadas por unidad de tiempo.		Sugerencias para incrementar la cantidad de trabajo.
	Número de tareas completadas.	Editor y consola	
	Artefactos generados por unidad de tiempo.	Editor	
Mitigación de riesgos	Acciones del sistema que impiden modificar código fuente bloqueado	Ediciones del usuario en el código bloqueado	Rediseñar el mecanismo de bloqueo

Tabla 1. Medidas de FAQuiS para efectividad, eficiencia y mitigación de riesgos

Las medidas relacionadas con la eficiencia (ver Tabla 1) computan la cantidad de tiempo que emplea el alumno en resolver el problema interactuando y ejecutando interacciones en todos los espacios del sistema. De forma más específica, se identifica el editor y la consola de COLLECE 2.0 que usa el alumno y los artefactos que genera. Ello facilita mecanismos de intervención para sugerir al alumno una mayor rapidez y adaptar la complejidad del problema.

La medición de la mitigación de riesgos se realiza en función de lo indicado en el modelo de tareas donde

se indican aquellas acciones que podría implicar algún peligro. En este caso se computan las acciones del sistema que de forma exitosa impiden modificar un fragmento del código fuente que fue bloqueado por otro alumno (ver Tabla 1).

Para cada una de las sub-características de satisfacción, a saber, utilidad, confianza, placer y comodidad, se establecen medidas específicas destinadas a evaluar y mejorar la experiencia del usuario. En primer lugar, las medidas de utilidad, que se detallan en la Tabla 2, son fundamentales para comprender cómo los usuarios interactúan con el sistema y qué aspectos podrían necesitar optimización. Estas medidas de utilidad procesan todas las acciones registradas en el repositorio de log del sistema. Este proceso de análisis minucioso permite identificar aquellas acciones previstas en el modelo de tareas que no son ejecutadas por los usuarios. Ejemplos de estas acciones incluyen la compilación de código, la ejecución de programas y el envío de mensajes en el chat.

Además de analizar las acciones no ejecutadas, estas medidas también examinan los diferentes espacios del sistema que son menos frecuentados por los usuarios. Por ejemplo, se analiza el uso de la consola y el panel de bloqueo de regiones. Estos espacios, aunque disponibles, pueden no estar siendo utilizados al máximo de su potencial. La identificación de estos espacios subutilizados proporciona una base sólida para intervenir en el proceso de colaboración. Una de las estrategias para mejorar la utilización de estos espacios es la implementación de tutoriales. Estos tutoriales pueden guiar a los usuarios sobre cómo aprovechar al máximo las funcionalidades y los espacios disponibles dentro del sistema.

El valor de estas medidas de utilidad no se utiliza sólo para emitir sugerencias al usuario del sistema CSCL. También se consideran modificaciones más estructurales en el sistema. En algunos casos, podría ser necesario un rediseño del sistema para hacer que estos espacios sean más accesibles y útiles para los usuarios. Este rediseño podría implicar cambios en la interfaz de usuario, la reubicación de ciertas funcionalidades o la introducción de nuevas características que respondan mejor a las necesidades y comportamientos observados de los usuarios.

Características y sub-características del ISO 25010:2011	**Descripción de las medidas propuestas en FAQuiS**	**Fuente de información**	**Mecanismos de intervención**
Utilidad	Número de tareas que incluyen acciones con riesgos o de tipo instrumental y se repiten en diferentes sesiones.	Editor y modelo de tareas	Tutoriales de uso del sistema. Rediseñar el sistema. Adaptar dificultad del problema
	Patrones con una respuesta exitosa de interacción RA	Espacio RA y modelo de tareas	
	Porcentaje de tareas concluidas.	Todos los espacios y modelo de tareas	
	Porcentaje de acciones utilizadas.		
	Porcentaje de espacios utilizados.		
	Porcentaje de acciones del usuario respecto aquellas que implican feedback de ayuda por parte del sistema.	Consola y RA	
	Porcentaje de tareas realizadas con éxito por el usuario con soporte del sistema	Consola	
	Porcentaje de tareas realizadas con éxito por el usuario con soporte protocolario	Bloqueo de código	
	Porcentaje de tareas terminadas respecto empezadas en todas las sesiones de trabajo.	Todos los espacios y modelo de tareas	
	Acciones ejecutadas que requieren una respuesta de otro usuario.	Chat	Modificar composición del grupo de trabajo
	Tiempo dedicado a acciones que requieren una respuesta de otro usuario.	Chat	
	Número de acciones en el paradigma RA	Espacio RA	Sugerencia para usar el espacio RA
	Tiempo dedicado a interacciones RA		Rediseñar RA

Tabla 2. Medidas de FAQuiS para utilidad

La confianza del usuario en el sistema se mide a través de un conjunto de valores diseñados específicamente para detectar varios indicadores de desconfianza o falta de compromiso por parte del

alumno (ver Tabla 3). Entre estos indicadores se encuentran situaciones en las que el alumno no recibe respuestas de sus compañeros en el chat, lo cual puede reflejar una percepción de aislamiento o falta de apoyo en el entorno colaborativo. Además, se evalúa si el alumno evita ejecutar acciones protocolarias necesarias para el bloqueo de código, una señal clara de que podría no confiar en la seguridad o en la eficacia de los procedimientos establecidos dentro del sistema.

También se analiza es si el alumno no culmina la resolución de problemas iniciados en sesiones de trabajo previas. Este patrón de comportamiento puede indicar una falta de confianza en la continuidad y en la utilidad del sistema para seguir apoyando su proceso de aprendizaje a lo largo del tiempo. Por último, se considera el uso del paradigma de Realidad Aumentada. Si el alumno evita o minimiza el uso de esta herramienta, podría ser un indicio de que no confía en su capacidad para mejorar la experiencia educativa o en su fiabilidad técnica. Todos estos factores son recopilados y analizados conforme a los valores detallados en la Tabla 3.

El análisis de estos comportamientos es esencial para identificar la necesidad de establecer mecanismos de intervención. Uno de los posibles mecanismos podría ser la modificación de la composición del grupo de trabajo. Cambiar la dinámica grupal puede ser una estrategia efectiva para restaurar la confianza del alumno, proporcionando un entorno más colaborativo y de apoyo que pueda influir positivamente en su percepción del sistema. Este enfoque se basa en la premisa de que un entorno socialmente reforzado puede incrementar significativamente la confianza y la motivación del alumno para participar activamente en el sistema.

Para cuantificar el placer derivado de la interacción con el sistema, se examina si el alumno es capaz de resolver problemas que requieren el desarrollo de nuevas habilidades (ver Tabla 3). La evaluación se enfoca en determinar si, a medida que el alumno avanza en las sesiones de trabajo, estas se vuelven menos demandantes en términos de esfuerzo, lo que indicaría una consolidación efectiva de las competencias adquiridas.

Características y sub-características del ISO 25010:2011	**Descripción de las medidas propuestas en FAQuiS**	**Fuente de información**	**Mecanismos de intervención**
Confianza	Número de tareas asociadas a riesgos y que el usuario evita ejecutar.	Bloqueo de código	Sugerencia de uso de la funcionalidad para bloqueo de código
	Acciones asociadas a riesgos y que el usuario ejecuta repetidamente.		
	Tiempos ejecutando tareas asociadas a riesgos.		
	Patrones de acciones que no siguen la secuencia de acciones esperada debido a una respuesta inesperada del sistema.	Todos los elementos del sistema y modelo de tareas	Rediseñar el sistema
	Porcentaje de tareas terminadas respecto empezadas en todas las sesiones de trabajo.		Adaptar dificultad del problema
	Acciones ejecutadas que requieren una respuesta de otro usuario.	Chat	Modificar composición del grupo de trabajo
	Tiempo dedicado a acciones que requieren una respuesta de otro usuario.	Chat	
	Número de acciones en el paradigma RA	Espacio RA	Sugerencia para usar el espacio RA
	Tiempo dedicado a interacciones RA		Rediseñar RA
Placer	Problemas resueltos con éxito que requieren nuevas habilidades	Modelo de usuario, todos los espacios del sistema y modelo de tareas	Adaptar el problema
	Variación en el tiempo de ejecución, es decir en diferentes sesiones, de las tareas que demandan nuevas habilidades.		
	Tendencia a reanudar sesiones de trabajo interrumpidas.	Todos los espacios del sistema	Recordatorio de trabajo pendiente. Adaptar el problema.

Tabla 3. Medidas de FAQuiS para confianza y placer

Cuando se detectan deficiencias en este ámbito, la principal medida de intervención es adaptar el problema para facilitar el aprendizaje del alumno (ver Tabla 3). Este enfoque puede incluir la simplificación de tareas o la introducción de recursos adicionales que apoyen la adquisición de competencias de manera más efectiva. El objetivo es ajustar el nivel de dificultad y los recursos disponibles para que el alumno pueda experimentar una progresión positiva y placentera en su proceso de aprendizaje.

Las medidas de comodidad (ver Tabla 4) están relacionadas con la cantidad de acciones que soporta un mismo espacio de la interfaz y pueden dificultar su utilización. También se considera el empleo del paradigma de Realidad Aumentada estimando que este le puede resultar al alumno más cómodo para la realización de sus tareas. El principal mecanismo de intervención para mejorar la comodidad es rediseñar el sistema en caso de que se detecten deficiencias en los valores de estas medidas.

Características y sub-características del ISO 25010:2011	**Descripción de las medidas propuestas en FAQuiS**	**Fuente de información**	**Mecanismos de intervención**
Comodidad	Densidad de trabajo en cada espacio	Modelo de tareas	Rediseñar el sistema
	Grado de interacciones RA		
Completitud	Las anteriores medidas por cada contexto de uso previsto		
Flexibilidad	Las anteriores medidas (excepto completitud) por cada contexto de uso no previsto		

Tabla 4. Medidas de FAQuiS para comodidad, completitud y flexibilidad

La completitud es una sub-característica de la cobertura del contexto que es cuantificada a través del resto de medidas para conocer si los cambios en el modelo de contexto (la colaboración es síncrona o asíncrona, composición del grupo de trabajo, etc.) influye en el resto de las características de calidad en uso (ver Tabla 4). La flexibilidad aplica las métricas ya calculadas para las otras características de calidad en uso para analizar situaciones que inicialmente no fueron identificados en el modelo del contexto (ver Tabla 4).

5. Conclusiones

Este artículo ha abordado la problemática de evaluar los procesos de resolución colaborativa de problemas en el ámbito de la programación soportada por sistemas CSCL. Esta evaluación debe tener en cuenta diversos aspectos como, por ejemplo, la adecuación de la formación de los grupos de trabajo, el soporte ofrecido por el sistema CSCL, las aportaciones de cada miembro del grupo o la calidad de los resultados finales. Con el objetivo de facilitar un modelo de evaluación estandarizado se ha presentado una propuesta basada en FAQuiS, un framework basado en modelos, para generar un conjunto de medidas de calidad en uso que cuantifican las características y sub-características de la norma ISO 25010:2011. Como caso de estudio, se ha analizado la aplicabilidad de este modelo de evaluación a COLLECE 2.0, un sistema CSCL distribuido síncrono para el aprendizaje de la programación. Este caso de estudio ha mostrado cómo la información recogida en archivos de log y en los modelos procesados por FAQuiS permite cuantificar las características y sub-características de la norma ISO 25010:2011. Ello abre la puerta a realizar evaluaciones estandarizadas de la calidad en uso de los sistemas CSCL que ofrecen soporte al proceso de aprendizaje de la programación.

En el futuro se realizarán estudios experimentales que evalúen las actividades de los alumnos con COLLECE 2.0 usando el modelo de evaluación de la calidad en uso. La utilización de este modelo sería el punto de partida para tutorizar y guiar en tiempo real las actividades de los alumnos.

Agradecimientos

Este trabajo está soportado parcialmente por el proyecto CODIFICA, ref. PID2021-125122OB-100, financiado por MCIN/AEI/ 10.13039/501100011033 y los Fondos Europeos de Desarrollo Regional (FEDER) “Una manera de hacer Europa”. Este trabajo está también soportado parcialmente por el Gobierno de Cantabria a través de los proyectos SUBVTC-2023-0013 y VP84.

Referencias

Becker, B. A., Glanville, G., Iwashima, R., McDonnell, C., Goslin, K., & Mooney, C. (2016).

Effective compiler error message enhancement for novice programming students. Computer Science Education, 26(2–3), 148–175. https://doi.org/10.1080/08993408.2016.1225464

Brown, N. C. C., Kölling, M., McCall, D., & Utting, I. (2014). Blackbox: a large scale repository of novice programmers' activity. Proceedings of the 45th ACM Technical Symposium on Computer Science Education, 223–228. Presented at the Atlanta, Georgia, USA. doi:10.1145/2538862.2538924

Cicirello, V. A. (2009). On the role and effectiveness of pop quizzes in CS1. Proceedings of the 40th ACM Technical Symposium on Computer Science Education, 286–290. Presented at the Chattanooga, TN, USA. doi:10.1145/1508865.1508971

Dolog, P., Thomsen, L. L., & Thomsen, B. (2016). Assessing Problem-Based Learning in a Software Engineering Curriculum Using Bloom's Taxonomy and the IEEE Software Engineering Body of Knowledge. ACM Trans. Comput. Educ., 16(3). doi:10.1145/2845091

Duque, R., Bravo, C., & Ortega, M. (2011). A model-based framework to automate the analysis of users' activity in collaborative systems. J. Netw. Comput. Appl., 34(4), 1200–1209. doi:10.1016/j.jnca.2011.01.005

Gavriushenko., M., Khriyenko., O., & Tuhkala., A. (2017). An Intelligent Learning Support System. Proceedings of the 9th International Conference on Computer Supported Education - Volume 1: CSEDU, 217–225. doi:10.5220/0006252102170225

Gross, S., Mokbel, B., Paassen, B., Hammer, B., & Pinkwart, N. (2014). Example-based feedback provision using structured solution spaces. Int. J. Learn. Technol., 9(3), 248–280. doi:10.1504/IJLT.2014.065752

Hui, B., & Farvolden, S. (2017). How Can Learning Analytics Improve a Course? Proceedings of the 22nd Western Canadian Conference on Computing Education. Presented at the Abbotsford, BC, Canada. doi:10.1145/3085585.3085586

Hundhausen, C. D., Olivares, D. M., & Carter, A. S. (2017). IDE-Based Learning Analytics for Computing Education: A Process Model, Critical Review, and Research Agenda. ACM Trans. Comput. Educ., 17(3). doi:10.1145/3105759

ISO/IEC 25010. (2011). ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.

Lacave, C., García, M. A., Molina, A. I., Sánchez, S., Redondo, M. A., & Ortega, M. (2019). COLLECE-2.0: A real-time collaborative programming system on Eclipse. 2019 International Symposium on Computers in Education (SIIE), 1–6. doi:10.1109/SIIE48397.2019.8970132

Leinonen, J., Leppänen, L., Ihantola, P., & Hellas, A. (2017). Comparison of Time Metrics in Programming. Proceedings of the 2017 ACM Conference on International Computing Education Research, 200–208. Presented at the Tacoma, Washington, USA. doi:10.1145/3105726.3106181

Li, J., Liying, F., Qing, X., Shi, Z., & Yiliu, X. (2010). Interface generation technology based on Concur Task Tree. 2010 International Conference on Information, Networking and Automation (ICINA), 2, V2-350-V2-354. doi:10.1109/ICINA.2010.5636493

MacNish, C. (2002). Machiine Learning and Visualisation Techniques for Inferring Logical Errors in Student Code Submissions. Retrieved from https://api.semanticscholar.org/CorpusID:60970311

Martínez Monés, A., Dimitriadis Damoulis, Y., Acquila-Natale, E., Álvarez, A., Caeiro Rodríguez, M., Cobos Pérez, R., … Sancho Vinuesa, T. (2020). Logros y retos en analítica del aprendizaje en España: La perspectiva de SNOLA. RIED-Revista Iberoamericana de Educación a Distancia, 23(2), 187–212. doi:10.5944/ried.23.2.26541

Ott, C., Robins, A., & Shephard, K. (2016). Translating Principles of Effective Feedback for

Students into the CS1 Context. ACM Trans. Comput. Educ., 16(1). doi:10.1145/2737596

Pettit, R. S., Homer, J. D., McMurry, K. M., Simone, N., & Mengel, S. A. (2015). Are automated assessment tools helpful in programming courses?. In 2015 ASEE Annual Conference & Exposition (pp. 26-230).

Piech, C., Sahami, M., Koller, D., Cooper, S., & Blikstein, P. (2012). Modeling how students learn to program. Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, 153–160. Presented at the Raleigh, North Carolina, USA. doi:10.1145/2157136.2157182

Rivers, K., & Koedinger, K. (2015). Data-Driven Hint Generation in Vast Solution Spaces: a Self-Improving Python Programming Tutor. International Journal of Artificial Intelligence in Education, 27. doi:10.1007/s40593-015-0070-z

Salomón, S., Duque, R., Montaña, J. L., & Tenés, L. (2019). Modeling Users Behavior in Groupware Applications. In Y. Luo (Ed.), Cooperative Design, Visualization, and Engineering - 16th International Conference, CDVE 2019, Mallorca, Spain, October 6-9, 2019, Proceedings (pp. 11–21). doi:10.1007/978-3-030-30949-7_2

Salomón, S., Duque, R., Montaña, J., & Tenés, L. (2022). Towards automatic evaluation of the Quality-in-Use in context-aware software systems. Journal of Ambient Intelligence and Humanized Computing, 14. doi:10.1007/s12652-021-03693-w

Schez-Sobrino, S., Gmez-Portes, C., Vallejo, D., Glez-Morcillo, C., & Redondo, M. Á. (2020). An Intelligent Tutoring System to Facilitate the Learning of Programming through the Usage of Dynamic Graphic Visualizations. Applied Sciences, 10(4). doi:10.3390/app10041518

Silva, L., Mendes, A. J., & Gomes, A. (2020). Computer-supported Collaborative Learning in Programming Education: A Systematic Literature Review. 2020 IEEE Global Engineering Education Conference (EDUCON), 1086–1095. doi:10.1109/EDUCON45650.2020.9125237

Spacco, J., Denny, P., Richards, B., Babcock, D., Hovemeyer, D., Moscola, J., & Duvall, R. (2015). Analyzing Student Work Patterns Using Programming Exercise Data. Proceedings of the 46th ACM Technical Symposium on Computer Science Education, 18–23. Presented at the Kansas City, Missouri, USA. doi:10.1145/2676723.2677297

van der Aalst, W. (2012). Process Mining: Overview and Opportunities. ACM Trans. Manage. Inf. Syst., 3(2). doi:10.1145/2229156.2229157