# *Sistema COLLECE mejorado para soportar aprendizaje colaborativo de la programación en tiempo real sobre Eclipse*

**Santiago Sánchez, María Ángeles García, Crescencio Bravo, Miguel Ángel Redondo**

Universidad de Castilla-La Mancha
Paseo de la Universidad, 4, 13071, Ciudad Real (España)
{Santiago.Sanchez, MariaAngeles.GMarin, Crescencio.Bravo, Miguel.Redondo}@uclm.es

**Resumen**: La enseñanza de la programación mediante un enfoque de resolución de problemas y la colaboración entre estudiantes, tiene un efecto directo en el proceso de aprendizaje. Las herramientas destinadas a facilitar este aprendizaje en entornos distribuidos y en tiempo real tienden a ofrecer características y funcionalidades limitadas, no logrando por tanto su objetivo principal de consolidarse como un entorno completo y usable con fines didácticos o profesionales. Los trabajos previos que presentan herramientas existentes fallan a la hora de ofrecer un ecosistema completo de enseñanza de la programación en tiempo real y de forma distribuida, debido a su enfoque particular en ciertos problemas recurrentes como sincronización y garantía de consistencia entre documentos compartidos. Este artículo describe la implementación de COLLECE 2.0, un plugin para la plataforma Eclipse que proporciona un entorno de programación distribuido y colaborativo en tiempo real, escalable y preparado para incluir tecnologías de aprendizaje emergentes. De este desarrollo se discuten sus características principales, su arquitectura y problemáticas encontradas desde un punto de vista técnico. El artículo finaliza con varios comentarios finales y algunas líneas de trabajo futuro que quedan planteadas.
**Palabras clave**: Colaboración, programación en tiempo real, desarrollo distribuido, aprendizaje de la programación, eclipse.

**Abstract:** Programming teaching through problem-solving approaches and student collaboration has a direct effect on the learning process. Tools designed to facilitate this learning in distributed and real-time environments tend to offer limited features and functionalities, thus not achieving their primary goal of being consolidated as a complete and usable educational or professional environment. Previous works presenting existing tools fail to offer a complete ecosystem of programming learning in a distributed and a real-time way, due to their focus on some recurrent problems such as synchronization and consistency assurance in shared documents. This paper describes the implementation of COLLECE 2.0, a plugin for the Eclipse platform that provides a real-time distributed and collaborative environment, that is scalable and adapted to include emerging learning technologies. From this work, we discuss its main features, its architecture, and problems found from a technical point of view. The paper closes with a few last comments and some future working lines.
**Key words:** Collaboration, real-time programming, distributed development, programming learning, eclipse.

 

## 1. Introducción

Los métodos de aprendizaje mediante programación colaborativa en tiempo real de forma distribuida han sido un importante tema de discusión durante mucho tiempo. El auge de herramientas que permiten a los usuarios trabajar de tal forma y las mejoras logradas en redes de alta velocidad lo demuestran. Sin embargo, estas nuevas herramientas intentan ser demasiado ambiciosas, construyendo completos sistemas que ofrecen a los usuarios una solución integrada en un entorno de programación colaborativo completo, sin llegar a alcanzar un resultado óptimo.

A estos resultados se llega a través de centralizar los esfuerzos en tareas de bajo nivel como la sincronización de proyectos, más concretamente la de archivos individuales entre colaboradores. Por ejemplo, trabajos previos como el de la herramienta CoEclipse se enfocan principalmente en temas como el mantenimiento de consistencia entre archivos remotos, es decir, garantizando la consistencia semántica y sintáctica, y no abarcando sin embargo un objetivo mayor que complete un entorno colaborativo de programación [1, 2].

Un escenario de uso común de programación colaborativa en tiempo real sería aquel en que varios estudiantes se enfrentan a la resolución mediante programación de un problema. Investigaciones existentes [3] confirman que la programación colaborativa en tiempo real es capaz de acelerar el proceso de resolución de problemas, facilitando la creación de mejores diseños y consiguiendo una menor longitud en líneas de código de los programas desarrollados, logrando por tanto una mayor calidad de los proyectos software.

El trabajo que se describe en este artículo se basa en una herramienta ya existente llamada *COLLaborative Edition, Compilation and Execution of programs* (COLLECE por sus siglas en inglés) [4], enfocado en guiar a los usuarios a resolver problemas de programación distribuida y en parejas. La nueva versión propuesta, denominada COLLECE 2.0, mejora las características y funcionalidades de la anterior para lograr un completo entorno de programación colaborativa en tiempo real. Estas nuevas características van más allá de resolver inconsistencias entre archivos remotos e integran paradigmas útiles de programación colaborativa como percepción y conocimiento del contexto, y resolución de problemas. Además, también cumple con los elementos de percepción del entorno de trabajo que un software de trabajo en grupo debe poseer [5], es decir, participación de los usuarios (presencia), localización de los usuarios, nivel de actividad, acciones de los usuarios, intenciones, cambios, etc.

COLLECE 2.0 se construye como un plugin para la plataforma Eclipse. El entorno de desarrollo de plugins de Eclipse (PDE por sus siglas en inglés) permite extender la plataforma respondiendo a distintas necesidades, lo que facilita la incorporación de nuevas características en un entorno de trabajo ya conocido por los estudiantes de programación.

Con todo lo anterior, la herramienta se engloba dentro de un contexto más ambicioso cuyo objetivo final es el de construir una nueva generación de herramientas para el aprendizaje de la programación con tecnologías interactivas emergentes. Una vez que la herramienta se consolide como un sistema sólido sobre el que trabajar, se integrarán estas nuevas tecnologías para lograr un entorno software avanzado de aprendizaje de la programación.

En este artículo se discute la nueva herramienta propuesta, COLLECE 2.0, como un entorno completo y orientado a la enseñanza de la programación mediante resolución de problemas. En la Sección 2 se presentan ciertas herramientas similares y el trabajo previo sobre el que se ha basado el desarrollo. Después, en la Sección 3 se presenta COLLECE 2.0; sus características, arquitectura y problemática, desde un punto de vista técnico. Finalmente, en la Sección 4 se plantean algunos comentarios de la investigación y algunas posibles líneas de trabajo futuro.

## 2. Antecedentes y trabajo previo

El presente trabajo se centra en un método de aprendizaje basado en la resolución de problemas de forma colaborativa. Estudios existentes sobre esta forma de aprendizaje sostienen las ventajas de aplicar metodologías relacionadas. En [6], por ejemplo, se plantea una investigación que evalúa un entorno de aprendizaje basado en retos con resultados

satisfactorios. Otros autores concluyen que un entorno de aprendizaje cooperativo de la programación aumenta la adquisición de conocimiento, desarrolla un pensamiento activo y creativo y mejora las habilidades sociales, entre otros beneficios [7, 8].

En lo relativo a herramientas similares a la discutida en este trabajo e integradas en la plataforma Eclipse, la anteriormente nombrada CoEclipse proporcionaría un entorno de trabajo colaborativo multi-usuario. Sin embargo, CoEclipse basa sus esfuerzos en garantizar la consistencia semántica y sintáctica entre los archivos compartidos, y en implementar una solución de bloqueo de regiones mediante dependencias [1].

Otra aplicación de este tipo sería Saros [9], desarrollada desde 2008 y que destaca por el ecosistema que ofrece. Además de implementar las mismas tecnologías de consistencia que CoEclipse, incluye distintos mecanismos de percepción y conocimiento del entorno: comunicación por chat, miembros conectados y estado de archivos compartidos, entre otros.

Por otra parte, la versión previa de la herramienta presentada en este artículo, COLLECE [4], conformaba un sistema de trabajo por sesiones donde varios usuarios podían trabajar de forma colaborativa, por turnos, en un archivo compartido de código fuente escrito en Java, siempre con fines de aprendizaje.

La herramienta era accesible como una aplicación auto-contenida y multi-plataforma vía web a través de *Java Web Start* (JavaWS). Una vez los usuarios conectaban con la aplicación, debían autenticarse con sus credenciales en una sesión de trabajo.

Por tanto, varios usuarios podían trabajar de forma conjunta en un archivo de código fuente en Java, cargando dicho archivo y el problema asociado a resolver en la sesión. El concepto de trabajo por turnos permitía a los usuarios de la sesión solicitar permiso para editar, compilar y ejecutar el programa. Esta ejecución del programa era compartida entre todos los clientes conectados a la sesión, por lo que la salida de dicha ejecución solicitada por un usuario era visualizada colaborativamente por el resto.

COLLECE contaba con diversos mecanismos de percepción y conocimiento:

- Panel de sesión con información del estado actual.
- Bloqueo del archivo de trabajo para conseguir una edición síncrona mediante el sistema de turnos.
- Estadísticas de usuarios para análisis futuro.
- Múltiples cursores identificativos de los usuarios conectados a la sesión.
- Histórico de acciones pasadas.
- Sonidos.
- Ventana de comunicación mediante chat estructurado.
- Funciones de solicitud de compilación y ejecución.

Entre sus desventajas se encontraban la edición de unidades de compilación independientes, es decir, el concepto de proyecto con múltiples archivos no estaba soportado; el soporte único de Java como lenguaje de trabajo; la edición del archivo de trabajo en texto plano, es decir, la ausencia de mecanismos como resaltado de sintaxis, autocompletado, etc., hoy en día existentes en IDEs modernos; la dificultad de configurar un servidor de COLLECE debido a las dependencias existentes (por ejemplo de MySQL).

De este trabajo previo y las pruebas realizadas con él se extrajeron varias conclusiones [10], entre ellas, el sistema de chat estructurado no resultó ser demasiado útil para los usuarios; es necesaria una herramienta de votaciones o similar, que permita a los usuarios sincronizar su trabajo con el resto; la herramienta puede ser usada en proyectos profesionales aplicando metodologías como *eXtreme Programming* (XP por sus siglas en inglés), más concretamente programación distribuida y en parejas.

## 3. COLLECE 2.0

Con todo lo anterior, en esta sección se presenta y discute el sistema COLLECE mejorado que evoluciona respecto de su antecesor incorporando nuevas características y facilidades para el usuario, con el fin último de convertirse en una herramienta para el aprendizaje colaborativo de la programación en tiempo real.

### 3.1. Características generales

Esta nueva herramienta se ha desarrollado como un plugin que pudiera incorporarse al entorno de desarrollo Eclipse, para que de esta forma fuera fácilmente instalable por los usuarios mediante el propio sistema de paquetes del entorno. Además de aprovecharse de la escalabilidad que este enfoque puede aportar.

Eclipse es un entorno de desarrollo integrado (o IDE) que permite desarrollar aplicaciones sobre distintos lenguajes de programación. Aunque inicialmente sólo soporta el lenguaje Java, mediante su sistema de plugins es fácilmente extensible para soportar otros lenguajes como C++, Python o JavaScript. Partir de un IDE ya existente y que es familiar para la mayoría de estudiantes de programación, proporciona un entorno base que evita tener que re-implementar herramientas de análisis y procesamiento de lenguajes. Los plugins desarrollados para la plataforma pueden aprovecharse de este sistema base para abordar otras tareas más complejas, relacionadas directamente con su propósito inicial, sin tener que preocuparse de otras no tan relacionadas como gestión de proyectos en el sistema de archivos, resaltado de sintaxis dependiente del lenguaje, o despliegue de la interfaz gráfica de usuario.

Sobre esta plataforma, COLLECE 2.0 provee de un conjunto de características que facilitan el aprendizaje de la programación mediante la participación y colaboración remota y síncrona de usuarios ante un problema propuesto. Estas funcionalidades permiten a los usuarios ser conscientes en todo momento del entorno que les rodea dentro de Eclipse y cooperar entre ellos para finalmente obtener una solución funcional al problema, así como aprender de lo que ven realizar a sus compañeros. Con este planteamiento, COLLECE 2.0 incorpora las siguientes características:

- *Modular*. La herramienta se instala fácilmente como un plugin a través del sistema de repositorios de Eclipse. Gracias a esto, la herramienta proporciona una robusta extensibilidad a través de otros plugins.
- *Sesiones de trabajo*. La forma de organizar el trabajo con la herramienta se basa en sesiones donde múltiples usuarios se conectan y trabajan simultáneamente. Estas sesiones están asociadas

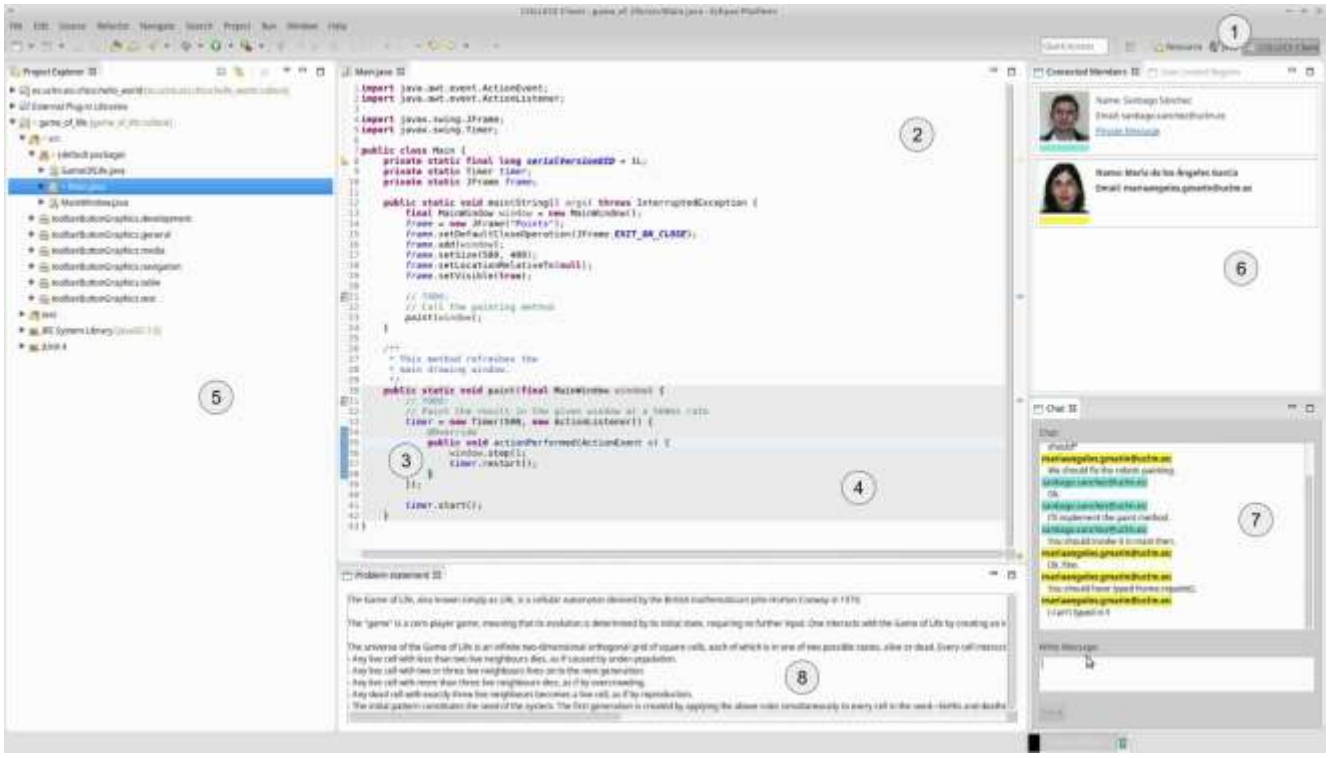


**Figura 1.** Vista de COLLECE 2.0 con dos usuarios colaborando en una sesión de trabajo; los elementos numerados se encuentran descritos en el texto

directamente con un problema de programación a resolver y pueden ser creadas por cualquier usuario.

- *Edición multi-usuario*. Para que varios usuarios puedan editar concurrentemente los mismos archivos, COLLECE 2.0 proporciona un sistema de edición colaborativa en tiempo real de baja latencia que permite dicha edición de forma fluida.
- *Tele-punteros*. Los usuarios deben conocer en todo momento quién está editando qué sección del archivo. Los tele-punteros proporcionan esta información mediante regiones coloreadas asociadas a un usuario.
- *Bloqueo de regiones*. Impedir que otros usuarios modifiquen secciones de código dentro de un archivo, previene la aparición de conflictos y mejora la coordinación entre usuarios. El bloqueo de regiones permite esto mediante el sombreado de líneas de código que se encuentran bloqueadas y que por tanto pertenecen a algún usuario.
- *Chat*. La herramienta proporciona un mecanismo de mensajería instantánea que identifica mediante un color y su nombre a los usuarios participantes en la sesión. Esta comunicación puede ser pública (a todos los miembros de la sesión) o privada (a un usuario concreto).
- *Sistemas de control de versiones*. COLLECE 2.0 hace un uso intensivo de los sistemas de repositorios para mantener de forma persistente el estado de los proyectos de código asociados a las sesiones. Los usuarios se ven directamente beneficiados de esto al poder compartir cualquier repositorio remoto (por ejemplo, desde un proveedor como Bitbucket o GitHub) en una sesión de trabajo con otros usuarios.

En la Figura 1 puede observarse una captura de pantalla de un cliente de COLLECE 2.0 en ejecución para dos usuarios, con distintos elementos: la perspectiva actual (1), la zona de edición multi-usuario (2), un tele-puntero indicando la posición del otro usuario (3), una región de código bloqueada (4), el explorador de proyectos sincronizado (5), la lista de miembros conectados a la sesión con sus colores identificativos (6), el panel de comunicación por chat (7) y el enunciado del problema a resolver (8).

### 3.2. Arquitectura

Como se ha mencionado anteriormente, COLLECE 2.0 sigue una implementación en forma de plugin para Eclipse. Los plugins desarrollados para esta plataforma tienen que ser escritos en lenguaje Java y empaquetados como un archivo JAR. La instalación manual del plugin pasaría simplemente por copiar ese archivo JAR en el directorio correspondiente de Eclipse. Sin embargo, el propio entorno proporciona un sistema de paquetes que permite la instalación automática junto a las dependencias del plugin, a partir de un sitio web que sirva por red una copia del plugin.

Para facilitar la escalabilidad del plugin, su arquitectura se ha organizado en módulos o subsistemas, maximizando la independencia entre ellos de tal forma que sea posible la sustitución de un módulo por otro que realice la misma función. Estos módulos desempeñan distintas tareas a bajo nivel como pueden ser aquellas relativas a la comunicación de red, sincronización, o dibujado, entre otras. La Figura 2 muestra una visión general de esta arquitectura.

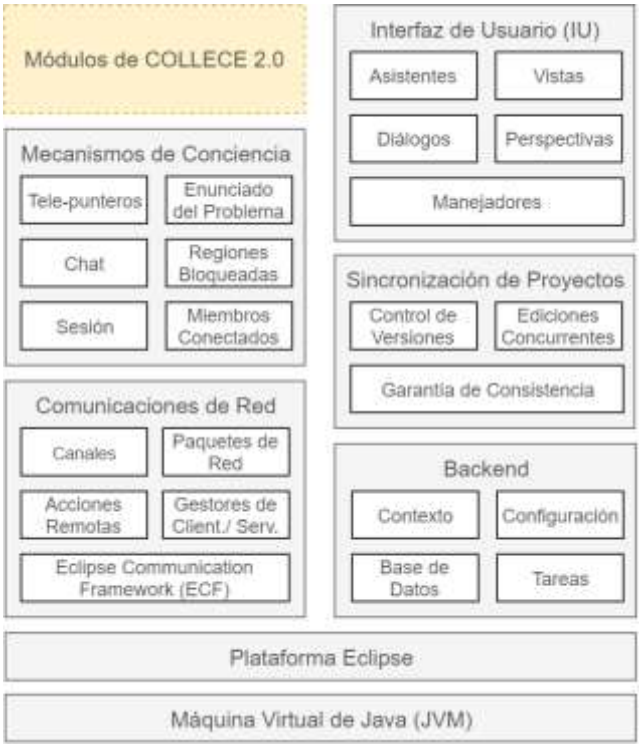


**Figura 2.** Diagrama de módulos que componen COLLECE 2.0

El sistema sigue un modelo de red cliente-servidor, donde un servidor central toma el control y mantiene la sincronización de sesiones entre el resto de clientes que se conectan a él. Un servidor puede gestionar al mismo tiempo distintas sesiones de trabajo. Estas sesiones de trabajo mantienen el contexto global entre los clientes conectados y el servidor, es decir, mantienen en memoria los datos de los usuarios, el estado de los proyectos asociados, y la información relativa al propio servidor. En última instancia, y a diferencia de los clientes, el servidor mantiene de forma persistente en disco una base de datos con la información relativa a las sesiones y las cuentas de usuario.

Esta arquitectura cliente-servidor permite mantener un diseño centralizado y consolidado, donde se tiene un nodo común contra el que realizar validaciones y mantener la sincronización de forma segura.

Dada la naturaleza tipo plugin de la herramienta, cualquier equipo de trabajo puede actuar como cliente o servidor si así lo desea, ya que la instalación del plugin no hace distinción entre ambos roles funcionales. El despliegue de un servidor de COLLECE 2.0 no supone ningún esfuerzo extra para los usuarios, más allá del de redireccionar los puertos de red a los equipos que se encuentren en una red privada en caso de que el servidor quiera trabajar a través de Internet.

Una vez que un usuario ha ejecutado un servidor, otros clientes pueden conectarse a este para comenzar a trabajar en alguna de las sesiones disponibles. Para ello, los usuarios deben haber registrado previamente una cuenta de inicio de sesión en el servidor al que se quieren conectar.

Estas sesiones mantienen la información relativa al proyecto de programación asociado, el problema a resolver, las fechas de inicio y finalización de la sesión, y la privacidad de esta. Esto último se logra mediante una lista de acceso de usuarios que pueden trabajar en la sesión, elaborada por el creador de la sesión. En cualquier otro caso en que la sesión sea pública, esta lista no es necesaria ya que se permite la conexión de cualquier usuario a la sesión.

Los clientes disponen de una interfaz de usuario totalmente integrada con Eclipse y proporcionada mediante el *framework* SWT. Esta interfaz de usuario, completamente personalizable, está compuesta por paneles o vistas, asistentes, cuadros de diálogo, manejadores de lógica o *handlers*, y perspectivas que agrupan dichos componentes. Esta última perspectiva permite cargar toda la configuración de la interfaz de usuario asociada al rol de cliente o servidor.

La comunicación cliente-servidor se realiza usando como apoyo el *framework* de comunicaciones de Eclipse (ECF), mediante el cual COLLECE 2.0 establece una conexión TCP entre los clientes y el servidor. A través de esta conexión se definen varios canales de comunicación asociados a cada sesión contra los que los clientes pueden unirse, siguiendo un patrón de publicación-suscripción, y que responden a las distintas características de la herramienta: un canal para la mensajería instantánea, otro para la sincronización de archivos, otro para las notificaciones, etc. Existe un canal especial compartido por todas las sesiones, planteado como una API pública, que puede ser usado por cualquier cliente, pensado para tareas de autenticación, y acceso a la base de datos de forma remota, entre otras.

### 3.3. Programación colaborativa en tiempo real

Una de las partes fundamentales de COLLECE 2.0, y que mejora respecto de su antecesor, es la posibilidad de editar concurrentemente, y sin turnos, el mismo archivo por múltiples usuarios ubicados remotamente. Y es que, la primera versión de la herramienta sólo soportaba una edición por turnos donde los usuarios debían solicitar y obtener permiso para modificar el archivo.

Esta edición distribuida en tiempo real se logra aplicando un algoritmo de *operaciones transformacionales* [11-13], más concretamente la implementación proporcionada por el módulo *Sync* de ECF, donde los clientes reciben los cambios de otros clientes, y los corrigen antes de aplicarlos a sus copias locales del archivo resolviendo posibles conflictos. Un ejemplo de aplicación de este algoritmo sería aquel donde dos usuarios intentan editar al mismo tiempo la misma región de texto. En este caso ocurriría un conflicto que debe resolverse: si partiendo del texto «abc» el usuario A inserta el

carácter «d» en la segunda posición (quedando «adbc») y el usuario B elimina el carácter de la tercera posición (quedando «ab») se obtendría un estado inconsistente entre ambos usuarios, y por tanto uno de los dos debería resolver el conflicto existente aplicando el algoritmo. En el caso de ejemplo, el usuario A debería transformar la operación de eliminación proveniente del usuario B (ver Figura 3).

Mediante este sistema de correcciones, que en última instancia implementa un algoritmo de ordenación causal [14], se mantiene un estado consistente entre todos los clientes del sistema. Al involucrar un servidor central, en el caso de COLLECE 2.0, se debe tener en cuenta además el estado del archivo que el propio servidor mantiene. El flujo de operaciones en este caso pasaría por que el servidor validara los cambios enviados por los clientes y los corrigiera en caso de conflictos. En última instancia, el servidor deberá reenviar los cambios al resto de clientes para que los apliquen a sus copias locales del archivo.

El rendimiento obtenido en lo relativo a estas operaciones transformacionales en distintos entornos de red es totalmente satisfactorio, obteniendo unas latencias mínimas en las comunicaciones a través de una red local e Internet, que hacen a la herramienta perfectamente utilizable en un entorno real. Las pruebas presentadas en la Figura 4 exponen cinco muestras tomadas en dos escenarios distintos: Internet y una red local de computadores. En ambos casos, las muestras se tomaron realizando operaciones de eliminación e inserción, con distintas longitudes de texto. Los resultados muestran para Internet una latencia media de 570 milisegundos,

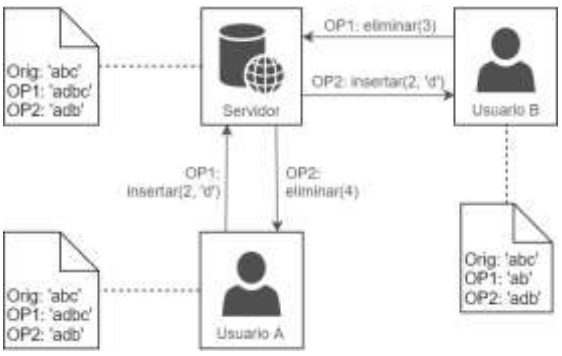


**Figura 3.** Sincronización de cambios entre archivos mediante operaciones transformacionales; todas las copias mantienen la consistencia después de aplicar el algoritmo de resolución de conflictos

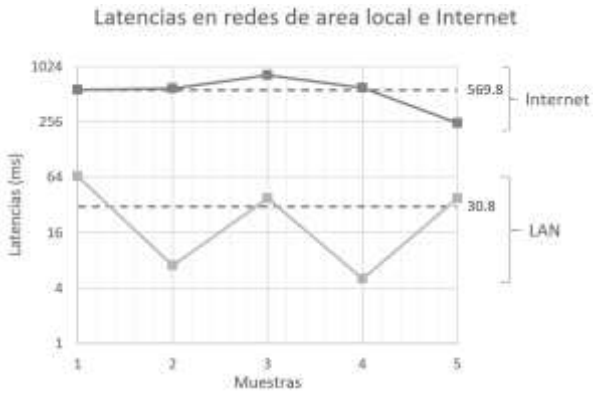


**Figura 4.** Comparativa de latencias en sincronización de archivos entre Internet y redes locales

mientras que para la red local 31 milisegundos, es decir, alrededor de 18 veces más lento en el primer caso. Hay que tener en cuenta que las pruebas realizadas en el primer escenario se han llevado a cabo con un acceso contratado a un proveedor externo, es decir, en un entorno común de una red doméstica con acceso a Internet.

Aunque COLLECE 2.0 proporciona una edición en tiempo real distribuida, nada impide a los usuarios bloquear completamente el archivo a editar para de esta forma adoptar un enfoque por turnos, o más específicamente de programación en pareja de forma distribuida, donde un usuario toma el rol de controlador para implementar la solución, mientras el otro adopta el de navegador para revisarla en tiempo real. Si por el contrario los usuarios quisieran adoptar un enfoque menos tradicional, podrían simplemente organizarse para implementar distintas partes del problema en el mismo archivo, sin que se produzcan conflictos entre sus soluciones.

Finalmente, puede darse el caso de que un usuario edite el archivo desde otro editor de texto, mientras está conectado a la sesión. Para este supuesto, COLLECE 2.0 implementa una tarea que se ejecuta cada cierto tiempo con el propósito de asegurar la consistencia entre la copia maestra de los archivos (la existente en el servidor) y la del resto de clientes. Esto se realiza comprobando diversos metadatos del archivo, como su tamaño y fecha de modificación, para finalmente comprobar que el *hash* o resumen del archivo del servidor coincide con el del resto de clientes. En caso contrario, los clientes con una copia conflictiva solicitan una copia del archivo al servidor.

### 3.4. Sincronización de proyectos

Sincronizar documentos individuales supone resolver cierta problemática relacionada con las posibles inconsistencias existentes entre clientes, como hemos visto en el apartado anterior.

Sin embargo, dicho caso se engloba dentro de una problemática aún mayor: la de sincronizar proyectos completos, es decir, conjuntos de archivos que conforman la solución al problema de la sesión.

El sistema planteado, permite que todos los clientes dispongan de la última versión de los proyectos. El servidor, por su parte, se encarga de gestionar y preparar esta sincronización. En la Figura 5 se puede observar un esquema de este sistema.

En primer lugar, un usuario ejecutaría el servidor encargado de gestionar el sistema (1). La primera tarea que éste realizaría sería actualizar los repositorios asociados a las sesiones de trabajo (2). Si una copia del repositorio no existiera en el servidor, además de obtenerla del repositorio remoto, deberá crear una rama de trabajo (de nombre «collece») tanto de forma local como remota. Trabajar con una rama dedicada permite a la herramienta no tener que preocuparse de resolver conflictos que puedan surgir de otros usuarios trabajando en el repositorio sin COLLECE 2.0.

Cuando un usuario se conecta al servidor (4), éste último actualiza la rama «collece» del repositorio remoto con los cambios que otros clientes hubieran podido producir (5).

Una vez que el servidor ha actualizado el repositorio remoto, éste notifica al cliente recién conectado –y al resto de clientes que se encuentren conectados, (6) que actualice su copia local del repositorio desde la rama «collece» remota (7).

Emplear un repositorio remoto para mantener la copia del proyecto siempre actualizada, aporta ciertas ventajas frente a una implementación propia en la que el servidor sea el que ofrezca la copia del proyecto de forma remota:

- Los usuarios se encuentran comúnmente familiarizados con sistemas de control de versiones, además de alojar y compartir sus proyectos en servicios *cloud* como los citados anteriormente.
- La ejecución de un servidor no se complica para los usuarios al no tener que desplegar un servidor externo de sistemas de control de versiones, con lo que ello conlleva.
- Los usuarios son los propietarios finales del repositorio, y son los que deciden en última instancia si los cambios aportados por las sesiones de trabajo en la rama «collece» deben ser mezclados en su rama «master» o no.
- Se permite a los usuarios trabajar de forma externa sin tener que involucrar una instancia del servidor de COLLECE 2.0 para ello.
- Integración de tareas de aprendizaje y de desarrollo o producción sobre el mismo entorno y/o proyecto.

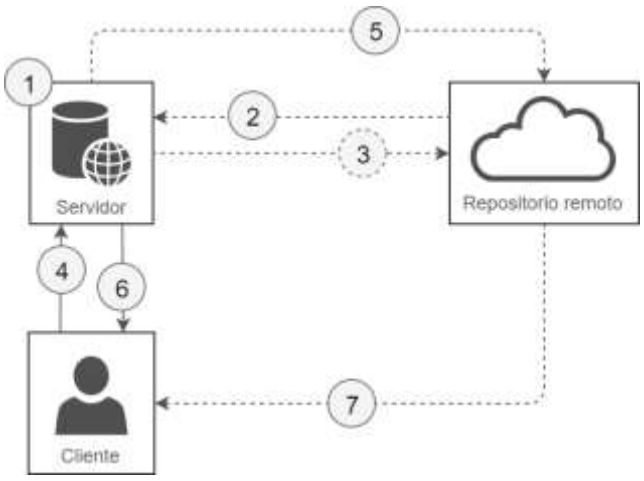


**Figura 5.** Sincronización de proyectos entre clientes y servidor; la secuencia de operaciones se encuentra explicada en el texto

## 4. Comentarios finales y trabajos futuros

La herramienta desarrollada está orientada a facilitar el aprendizaje realista de la programación mediante un entorno familiar y sencillo de usar para los usuarios. El enfoque basado en la resolución de problemas de forma colaborativa contribuye a proporcionar a los usuarios un sistema de aprendizaje orientado al mundo real.

Gracias a los mecanismos de percepción del entorno implementados en la herramienta, los usuarios nunca

pierden el flujo de trabajo de la tarea que pretenden abordar.

El planteamiento de la herramienta cuenta con ciertas limitaciones que conviene destacar:

- La sincronización de proyectos mediante repositorios obliga al propietario del repositorio a proporcionar acceso a la herramienta mediante un usuario previamente creado que será el que use COLLECE 2.0 para actualizar el repositorio remoto. Esto, sin embargo, es relativamente sencillo de abordar sustituyendo el sistema de credenciales clásico de usuario y contraseña por un sistema de *tokens* basado en *OAuth2*, actualmente soportado por la mayoría de servicios *cloud* de sistemas de repositorios existentes.
- Actualmente COLLECE 2.0 es soportado por las versiones Mars y Neon de Eclipse. La futura versión, llamada Oxygen, presenta algunos problemas al haber actualizado la versión de ECF, aunque no deja de ser un problema menor al poder incrustar las versiones específicas de las dependencias en el propio plugin.
- Algunas características de Eclipse, como la auto-inserción de paréntesis, llaves y corchetes comprometen la gestión del editor de archivos por parte del plugin, resultando en algunas anomalías como duplicidad de caracteres a la hora de sincronizar archivos entre clientes. Hasta el momento, esto puede resolverse desactivando dichas características en la configuración de Eclipse o esperar a que dicha desincronización se resuelva automáticamente mediante la tarea periódica de garantía de consistencia.

El futuro de la herramienta pasa por ser ampliada para soportar ciertas tecnologías emergentes como el uso de dispositivos móviles y técnicas de realidad aumentada para visualizar las características estructurales de los programas, visualizar la ejecución de las soluciones implementadas, compartir la ejecución gráfica de dichas soluciones entre los distintos clientes de la sesión, proporcionar una pizarra compartida que los usuarios puedan utilizar para expresar sus ideas, un sistema de votaciones que en sesiones de trabajo con múltiples usuarios permita definir la próxima tarea a implementar, o la visualización de estadísticas relacionadas con el número de compilaciones, ejecuciones, líneas de código escritas por cada usuario, etc.

Sin duda, la herramienta se ha planteado como una plataforma lo más escalable posible que permita ser extendida fácilmente para soportar otras características relacionadas con el aprendizaje de la programación. Se espera, sin embargo, que la próxima tarea a abordar sea la realización de un estudio de campo con usuarios reales que cursen estudios relacionados con la programación para observar cómo afecta realmente COLLECE 2.0 en su proceso de aprendizaje.

## Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía, Industria y Competitividad, y el Fondo Europeo de Desarrollo Regional a través del proyecto TIN2015-66731-C2-2-R.

## Referencias

[1] H. Fan y C. Sun, "Achieving integrated consistency maintenance and awareness in real-time collaborative programming environments: The CoEclipse approach," *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2012*, Wuhan, China, 2012.

[2] H. Fan y C. Sun, "Dependency-based automatic locking for semantic conflict prevention in real-time collaborative programming," *Proceedings of the ACM Symposium on Applied Computing*, Trento, Italia, 2012.

[3] H. Shen y C. Sun, "RECIPE: a prototype for Internet-based real-time collaborative programming," *Proceedings of the 2nd Annual International Workshop on Collaborative Editing Systems*, Filadelfia, Pensilvania, EEUU, 2000.

[4] C. Bravo, R. Duque, y J. Gallardo, "A groupware system to support collaborative programming: Design and experiences," *Journal of Systems and Software*, vol. 86, pp. 1759-1771, 2013.

[5] C. Gutwin y S. Greenberg, "Workspace awareness in real-time distributed groupware," 1995.

[6] Á. Fidalgo-Blanco, F. García-Peñalvo, y M. Sein-Echaluce Lacleta, "Aprendizaje Basado en Retos en una asignatura académica universitaria," 2017.

[7] V. Jadzgevičiene y J. Urboniene, "Cooperation in programming learning," *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, Koli, Finlandia, 2012.

[8] W. Y. Hwang, R. Shadiev, C. Y. Wang, y Z. H. Huang, "A study of cooperative computer programming learning behavior and its influence on learning performance," *International Conference on e-Learning*, Kidmore End, Inglaterra, Reino Unido, 2012.

[9] S. Salinger, C. Oezbek, K. Beecher, y J. Schenk, "Saros: an eclipse plug-in for distributed party programming," *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, Ciudad del Cabo, Sudáfrica, 2010.

[10] A. I. Molina, J. Gallardo, M. A. Redondo, y C. Bravo, "Evaluating the awareness support of COLLECE, a collaborative programming tool," *Proceedings of the XV International Conference on Human Computer Interaction*, Puerto de la Cruz, Tenerife, España, 2014.

[11] A. Ellis y S. J. Gibbs, "Concurrency control in groupware systems," *Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, Portland, Oregon, EEUU, 1989.

[12] A. Nichols, P. Curtis, M. Dixon, y J. Lamping, "High-latency, low-bandwidth windowing in the Jupiter collaboration system," *Proceedings of the 8th annual ACM symposium on User interface and software technology*, Pittsburgh, Pensilvania, EEUU, 1995.

[13] C. Sun y C. Ellis, "Operational transformation in real-time group editors: issues, algorithms, and achievements," *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, Seattle, Washington, EEUU, 1998.

[14] A. Aguilar, F. Jurado, y M. A. Redondo, "Un algoritmo de ordenación causal de mensajes para aplicaciones educativas colaborativas que utilizan dispositivos móviles con redes inalámbricas," *Proceedings of 1st Spanish Congress of Computer Science (CEDI-Sintice)*, Granada, España, 2005.