Enseñando a programar en la orientación a objetos

Spigariol, Lucas
Universidad Tecnológica Nacional, F.R. Buenos Aires / F.R. Delta
Passerini, Nicolás
Universidad Tecnológica Nacional, F.R. Buenos Aires. Universidad Nacional de Quilmes.

Abstract
El presente trabajo consiste en una investigación pedagógica sobre la dinámica de enseñar a programar en el paradigma orientado a objetos en la universidad, centrada en la utilización de un software educativo denominado *LOOP (Learning Object Oriented Programming)*, desarrollado específicamente para este objetivo por un equipo de docentes. Con una mirada desde la pedagogía crítica, reflexiva sobre la práctica docente y atenta al contexto actual de la actividad profesional, utilizando técnicas cualitativas de análisis, se sistematiza el recorrido que fueron haciendo docentes y estudiantes en torno a dicha herramienta, desde su gestación hasta la evaluación de su impacto y las propuestas de actualización. El objetivo central de *LOOP*, que funciona dentro del ambiente del lenguaje Smalltalk, es realizar ejercicios prácticos utilizando la computadora ya desde el inicio de la cursada de la materia sin que el estudiante deba conocer todos los conceptos del paradigma que se requieren para usar el lenguaje normalmente. Se basa en la posibilidad de crear objetos y enviar mensajes interactivamente sin definir clases, permite ir aplicando progresivamente los conceptos de delegación, polimorfismo y colecciones, entre otros, a la vez que cuenta con una interfaz gráfica que permite visualizar a los objetos y sus referencias en el ambiente. La experiencia se desarrolla originalmente en la cátedra Paradigmas de Programación de la carrera de Ingeniería en Sistemas de la F.R. Buenos Aires de la Universidad Tecnológica Nacional y se extiende luego a la F.R. Delta (UTN) y a las Universidades Nacionales de Quilmes y de San Martín.

Palabras Clave
Programación orientada a objetos, Smalltalk, pedagogía crítica, software educativo, paradigmas de programación.

Introducción

Un grupo de docentes que forma parte de la cátedra Paradigmas de Programación, de la carrera de Ingeniería en Sistemas de Información (UTN - FRBA) desarrolló un software educativo, llamado inicialmente Object Browser y luego renombrado como *LOOP (Learning Object Oriented Programming)*, con el fin de utilizarlo para la enseñanza de algunos conceptos básicos de la programación orientada a Objetos.

Con sucesivas mejoras y actualizaciones su uso se fue extendiendo, por un lado pasando a ocupar un lugar destacado dentro de la planificación de la materia, y por otro siendo utilizado por todos los docentes de la cátedra y de materias afines de carreras de sistemas de otras universidades.

Entendemos a *LOOP* como software educativo, en tanto es un "programa de computación realizado con la finalidad de ser utilizado como facilitador del proceso de enseñanza"[1]. Por eso, para realizar una evaluación de calidad consideramos importante priorizar la mirada pedagógica, enfatizando criterios tales como la "facilidad de uso, la relevancia curricular, el enfoque pedagógico o la orientación hacia los alumnos".[2] En este sentido, realizamos una mirada reflexiva sobre la experiencia de utilizar *LOOP* en el aula, analizando e interpretando con criterios pedagógicos los motivos de la creación del software, su inserción dentro de una propuesta educativa

más abarcativa, la forma en que se lo utiliza por parte de los docentes, su sentido en función del contexto del ejercicio profesional, el impacto que tiene en los estudiantes y las posibilidades de ampliar su funcionalidad y alcance. Desde allí, pretendemos hacer un aporte teórico al campo de la enseñanza de la ingeniería, en particular, a la forma en que se puede enseñar a programar en el paradigma de objetos en la el ámbito universitario.

LOOP, sin llegar a ser un lenguaje en sí mismo, es una herramienta de desarrollo de software que se puede enmarcar en una amplia tradición de lenguajes didácticos, iniciada por el emblemático LOGO y la teoría construccionista de Papert [3] y continuada por Pascal y muchos otros, entre los que actualmente se encuentra el proyecto Scratch[4] o Alice[5]. En concreto, es una aplicación que se instala sobre el Smalltalk -la última versión está hecha sobre Pharo-, y que se integra como una opción más entre todas las que traen el IDE del lenguaje. Al activarla se abren una serie de ventanas para realizar cada una de las tareas (*Object Browser, workspaces*, diagrama de objetos, *tests*, etc). Su objetivo central es que los estudiantes puedan fácilmente realizar ejercicios prácticos de desarrollo de software utilizando la computadora y que puedan probarlos y hacer un seguimiento de su funcionamiento, sin que necesiten para ello comprender todos los conceptos del paradigma que se requieren normalmente para usar el lenguaje Smalltalk. Se basa en la posibilidad de crear objetos y enviar mensajes interactivamente, sin definir clases, permite ir aplicando progresivamente los conceptos de delegación, polimorfismo y colecciones, entre otros, a la vez que cuenta con una interfaz gráfica que permite visualizar a los objetos y sus referencias en el ambiente.

Una referencia teórica fundamental la constituyen dos textos escritos por los mismos desarrolladores de LOOP, en los que se presenta y justifica la herramienta. [6] [7] Así también, para una descripción más completa de la herramienta en sí, es ineludible el sitio web del que se puede descargar libremente el software, tutoriales y toda su documentación. [8]

Elementos del trabajo y metodología

Como punto de partida, explicitamos las convicciones que nos movilizaron a realizar este trabajo y el marco desde el cual lo concretamos. Partimos de entender a la universidad como un ámbito en el cual ni la impronta técnica profesional que se le imprime a la formación, ni la madurez de los jóvenes adultos que concurren, hace que pierda sentido el esfuerzo pedagógico de los docentes para llevar adelante su tarea educativa, sino todo lo contrario: la complejidad de los contenidos y lo cambiante del contexto profesional requiere una reflexión crítica permanente que permita establecer estrategias pedagógicas específicas. Por lo tanto, creemos fundamental prestar atención a estos aspectos:

- La comprensión de la gradualidad y complejidad del proceso de aprendizaje y del protagonismo del estudiante en dicho proceso.
- La importancia de asumir el rol docente en tanto mediador entre los contenidos y el estudiante y en ese sentido, promotor de todo tipo de recursos que favorezcan el proceso de aprendizaje.
- La necesaria, conflictiva y enriquecedora articulación entre teoría y práctica.
- Una lectura permanente del contexto de la actividad profesional desde el que se mira críticamente el curriculum pres-

cripto, la realidad de los estudiantes y la propia práctica docente.

Metodología

El trabajo se puede categorizar como un estudio de caso. Utilizamos métodos cualitativos de relevamiento de datos, tales como entrevistas a los docentes de la materia, a estudiantes y a profesionales de sistemas que trabajan con lenguajes de objetos. También realizamos observaciones participantes en ciertas clases claves de la materia, como las que se desarrollan en el laboratorio de computadoras utilizando *LOOP* y las que se destinan a que los estudiantes evalúen el desarrollo de la cursada. A su vez, hicimos un análisis del software y su documentación y una exploración bibliográfica en distintos campos teóricos relacionados.

Para sintetizar la concepción epistemológica que utilizamos, apelamos a la distinción entre los distintos métodos: "A diferencia de los métodos cuantitativos que se enmarcan en una concepción positivista, que aplican controles rígidos a situaciones 'artificiales' y en cuya aplicación el investigador intenta operar manteniendo cierta distancia y neutralidad, en los métodos cualitativos se actúa sobre contextos 'reales' y el observador procura acceder a las estructuras de significados propias de esos contextos mediante su participación en los mismos."[9] En este sentido, tratándose de un proceso tan complejo y multidimensional como es el aprendizaje, evitamos reducir el resultado de la investigación a una mera medición de resultados de aprobación de la materia o realizar comparaciones artificiales entre cursos que utilizan o no la herramienta, y afirmamos en coincidencia con investigaciones anteriores en evaluación de calidad de software educativo que "evaluar logros de dos grupos comparando resultados, significa considerar al alumno sin la influencia del entorno social, la motivación inherente de esta contextualización y perder de vista la expectativa social que lo conduce a adquirir ese aprendizaje". [10] No pretendemos establecer una afirmación de validez universal ni de extrapolar mecánicamente las conclusiones acerca del uso del software *LOOP* a cualquier otro contexto de enseñanza, pero sí poder interpretar la significatividad del uso de la herramienta y cuál es el aporte que hace al campo de la enseñanza de la ingeniería.

LOOP

Antes de seguir avanzando, es necesario un pequeño apartado para describir, aunque sea someramente, en qué consiste realmente el *LOOP*. Para ello recurrimos a cierto vocabulario técnico específico del paradigma de objetos y del lenguaje Smalltalk. La Figura 1 muestra la pantalla principal de la herramienta.

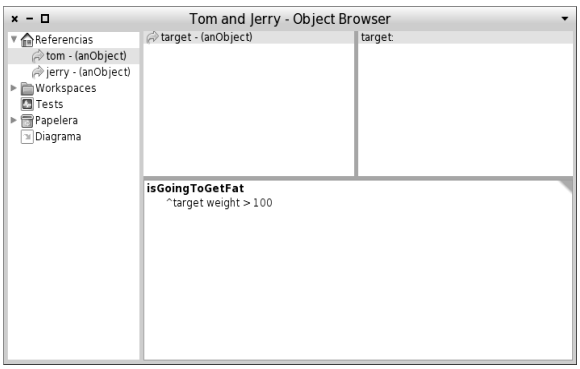

Figura 1. Dos objetos creados, uno de ellos con un atributo y un método.

En primer lugar, *LOOP* permite crear un objeto mediante un comando interactivo en el que basta con ingresar un identificador con el que se lo va a conocer en el ambiente. Dicho en otras palabras, no se requiere previamente definir la clase, explicitar la forma en que se la instancia ni mucho menos ubicar la clase en la jerarquía, como así tampoco tener que saber de antemano qué atributos o métodos necesita.

Luego, interactivamente se le pueden ir agregando los atributos y desarrollando los métodos, de manera que el objeto pueda responder a los mensajes que se le envíen desde el *workspace*. Para escribir el código de cada método, se despliega una ventana idéntica a la que cuenta el Smalltalk y se utiliza exactamente la misma sintaxis.

La herramienta permite crear colecciones, en forma interactiva y desentendiéndose de los detalles más complejos, que se enseñarán en una etapa posterior. También existe la posibilidad de definir *tests*. Una papelera de reciclaje permitir recuperar los objetos ya no referenciados, lo que resulta de gran utilidad para un alumno, que lógicamente en su proceso de aprendizaje y exploración puede cometer este tipo de errores.

Otra característica importante es que incluye un entorno gráfico de sencillo manejo, en el que se representa un diagrama de objetos, llamado también "grafo" de objetos (c.f. Figura 2). En él se ven las relaciones entre objetos, se exhibe el estado interno de los objetos y se va modificando sólo a medida que se evalúan los mensajes del *workspace* que provocan que el estado interno de los objetos se modifique.

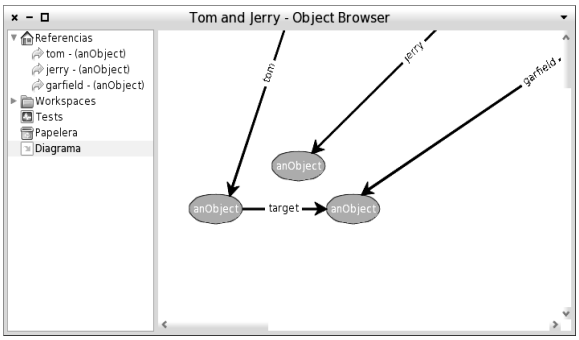

Figura 2. Diagrama de objetos, en los que se ven las relaciones entre ellos.

Al igual que en Smalltalk, para crear objetos polimórficos sólo se requiere que ambos entiendan el mismo mensaje. No se necesita aprender ningún otro concepto o construcción específica del lenguaje. Lo que no permite hacer, y de alguna manera determina el alcance del software, es la definición de clases, y en consecuencia, todo el manejo de herencia que trae aparejado. Sin embargo, ofrece entre sus opciones la posibilidad de clonar objetos, como otra forma de compartir código entre objetos.

La primera versión de *LOOP* utilizada en un curso de paradigmas fue en el año 2006. Entonces se llamaba *Object Browser* y funcionaba sobre Dolphin Smalltalk 6.0. Quienes llevaron adelante el proyecto fueron algunos de los docentes de la materia Paradigmas de Programación de la UTN-FRBA. En 2009, cuando la nueva versión de Dolphin eliminó la posibilidad de contar con licencias educativas, se decidió la migración a Squeak/Pharo, con el objetivo de seguir contando con una herramienta de código abierto. Finalmente, en 2011 se incluyeron los *tests*, la papelera de reciclaje y diagramas de objetos, llegando a una configuración similar a la actual. Esta versión fue presentada en la conferencia del *ESUG (European Smalltalk User Group)* en el año 2011, obteniendo el tercer lugar en los *8° Innovation Technology Awards*.

Durante ese tiempo, toda la cátedra se fue involucrando en el proyecto, utilizando cada vez más intensivamente la herramienta y se extendió a la Facultad Regional Delta (UTN) y a las Universidades Nacionales de Quilmes y de San Martín, en materias en las que también se enseña programación orientada objetos, aunque ubicadas de forma diferente en sus respectivos planes de estudios.

Análisis del software

El análisis de *LOOP* como software educativo se sistematiza de la siguiente manera:

- El contexto profesional actual de la programación utilizando tecnologías orientadas a objetos, que marca el horizonte de trabajo.
- La motivación de los creadores del software, quienes lo hicieron desde su ponderación, orientación y orden del programa de la asignatura.
- La experiencia de aprendizaje vivida por los estudiantes, que aporta un elemento significativo a la hora de analizar los resultados.
- La utilización de *LOOP* por parte de los docentes, con su propia mirada acerca del proceso de enseñanza.

Contexto profesional

Lejos de asumir una concepción mercantilista de la formación universitaria que la reduzca meramente a satisfacer las demandas presentes de la actividad privada, una mirada amplia y crítica de la actualidad profesional, y sobre todo de las tendencias que se vislumbran, es sumamente importante para poder orientar acerca de las capacidades que se pretenden desarrollar en los estudiantes de sistemas.

En la actualidad, es innegable la plena vigencia de la programación orientada a objetos dentro del panorama general de los diferentes paradigmas existente. Sin embargo, al acercarnos a las prácticas concretas, el principal problema que se vislumbra es la subutilización de las posibilidades que ésta ofrece. Notamos que es muy frecuente que "proyectos de IT que utilizan herramientas donde se pueden aplicar las ideas de objetos, el aprovechamiento de las potencialidades que brinda el trabajo con objetos resulta escaso".[11] En los últimos años puede decirse que han ocurrido algunas mejoras en cuanto a la utilización de tecnologías de objetos en el ámbito industrial, pero no han desaparecido los problemas.

Consideramos que uno de los principales problemas es una carencia en la comprensión del polimorfismo como herramienta para lograr encapsulamiento y flexibilidad. El ingeniero Franco Bulgarelli afirma que es habitual encontrar desarrollos en los que “las abstracciones son pobres y el polimorfismo inexistente". El ingeniero Fernando Dodino es contundente al expresar que la mayor dificultad es "buscar polimorfismo y manejarlo adecuadamente". En general, los profesionales consultados acuerdan en que los problemas frecuentes que perciben en quienes ingresan al mercado laboral es la falta de abstracción y la repetición de código, que muchas veces se resuelve con un uso adecuado del polimorfismo. En este sentido, el diagnóstico acerca de la dificultad del polimorfismo es coincidente con trabajos en otros países.[12]

Otro problema lo constituye la gran cantidad de diseños en los que encontramos una clara separación entre estado y comportamiento. Bulgarelli indica que esta separación lleva a una programación de tipo procedural, en la que las clases se utilizan alternativamente como módulos o como estructuras.

Esta subutilización de las tecnologías de objetos se ve potenciada por la popularización de algunas herramientas tanto teóricas como tecnológicas pensadas para ser utilizadas junto con lenguajes orientados a objetos, pero que promueven los diseños procedurales, por ejemplo presentando como un diseño deseable la separación entre objetos “que hacen” (con forma de bibliotecas de funciones”) y objetos “que son” (estructuras de datos sin comportamiento). Podemos citar como ejemplo algunas herramientas MVC en las que se sugiere que el comportamiento esté en el *Controlador* y el *Modelo* sólo contenga los datos. De la misma manera, algunas arquitecturas en capas su-

gieren tener una *capa de datos*, sin comportamiento alguno. Incluso algunas herramientas tecnológicas de amplia adopción en la industria imponen este tipo de diseños. Estas ideas y herramientas, presentadas como mejoras o como "desacoplamiento" difunden visiones que no aprovechan las posibilidades del paradigma y, por el contrario, detienen la evolución del conocimiento del mismo, retrasando la posibilidad de una adopción industrial más profunda.

Por otro lado, las herramientas que se utilizan en proyectos basados en tecnología de objetos pero que no permiten un aprovechamiento de sus posibilidades llevan, en muchos casos, a una frustración en muchos estudiantes e ingenieros jóvenes que, habiendo aprendido objetos en el ámbito académico, intentan llevar ese conocimiento a su experiencia laboral. Asimismo, encontramos casos en los que la imposibilidad de aplicar el paradigma debido a las limitaciones de herramientas desactualizadas o pobremente concebidas, constituye un desafío profesional en el que desde la universidad se puede hacer un aporte significativo a la industria. Es fundamental tener presente que la universidad no está simplemente a merced a los vaivenes del mercado, sino que es un actor central que influye en cómo se va configurando el ambiente profesional de la actividad. Por lo tanto, el conocimiento de objetos según se enseña en la Universidad tiene un ámbito de aplicación que no se limita a la academia, sino que tiene sentido su uso en proyectos industriales.

Finalmente, son notorias las coincidencias en que es frecuente ver una exagerada orientación a clases, es decir, por ejemplo, que muchas veces los programadores recurren a la subclasificación como única manera de tener comportamiento diferenciado, cuando bien podrían encontrar obtener distintas combinaciones de comportamiento empleando composición. Es oportuno tener en cuenta que existen otras investigaciones que abordan la relación entre objeto y clase, y que hay lenguajes de programación que manteniendo la idea central de objetos entienden a las clases de otra manera o incluso borran la diferencia entre objeto y clase, que introducen discusiones interesantes y que han sido propuestos como herramientas educativas, como Self[13] o JavaScript[14].

Motivaciones iniciales

La *dicotomía entre teoría y práctica*, de la que habla Paulo Freire respecto de los sistemas educacionales de América Latina, "dicotomía que tiene que ver de inmediato con una comprensión mecanicista de lo que es enseñar y de lo que es aprender"[15], no parece haber perdido vigencia. Retomando lo que llamó *educación bancaria* [16], muchas veces en la actualidad universitaria se tiene una concepción de enseñar que se reduce a transferir el concepto vacío de contenido y no transmitir la comprensión precisa de este contenido. Retomando a Freire, aprender y enseñar debieran ser entendidos como producción de saber, como producción de conocimiento.

Siguiendo este razonamiento, es evidente que para la enseñanza de la programación es conveniente desarrollar una dinámica que combine lo teórico con lo práctico. Fácilmente puede ser aceptado como un principio básico, pero lo interesante para analizar es que al contextualizarlo en la práctica docente de una asignatura en particular surgen las inevitables –y bienvenidas- preguntas acerca de qué teoría estamos hablando, qué entendemos por práctica y, por sobre todo, de qué se trata esto de "combinar". Una primera aproximación es que la práctica permita comprender mejor el sentido de los conceptos teóricos y a la vez

que dichos conceptos permitan plasmar soluciones prácticas adecuadas a problemas reales.

El hecho de enfrentarse de pronto a la computadora y tener que no sólo entender sino también expresarse en ese lenguaje formal con toda la complejidad de su sintaxis y lo abstracto de su simbología, manipular entornos de desarrollo con múltiples opciones, poder interpretar y depurar errores, realizar seguimientos, validar resultados y todo lo que implica lograr que un programa funciones y lo haga correctamente, presenta evidentemente un grado alto de dificultad. Este abismo que en definitiva separa al estudiante de la programación, no es menor ni trivial; tampoco es una dificultad insalvable: requiere de mediaciones pedagógicas, y *LOOP* apunta en esa dirección. Tratándose de conceptos que se constituyen como herramientas para desarrollar software, la forma más evidente de ponerlos en práctica es utilizando algún lenguaje de programación en particular, que en este caso se trata de Smalltalk.

Los docentes veían en los estudiantes que el uso del lenguaje representaba una curva de aprendizaje abrupta en los primeros momentos de la materia ya que requieren del manejo de una cantidad amplia de conceptos antes de poder realizar algo relativamente sencillo. Esto llevaba, en algunos casos, a demorar las instancias de práctica concreta con los lenguajes hasta tener una base conceptual más sólida, o en otros, a la incorporación desmedida de conceptos no interiorizados adecuadamente para poder acceder más rápidamente a una práctica concreta. En cualquiera de ambos extremos, la disociación entre teoría y práctica que se generaba era ciertamente contraproducente y dificultaba el proceso de aprendizaje.

Por otra parte, también señalaban otros problemas, como la falta de capacidad para construir software que efectivamente aproveche las características del modelado con objetos. Entre ellas señalan la dificultad de que al codificar un método, se tenga en cuenta que "se va a evaluar cuando se envíe un mensaje a un objeto concreto, y que por lo tanto el código del método se evalúa en el contexto del objeto receptor".[17] También advierte sobre limitaciones al entender el concepto de variable como una referencia a un objeto y darse cuenta de que un objeto necesita conocer a otro para poder enviarle mensajes, de donde surge la idea de representar visualmente el grafo de las relaciones de conocimiento entre los objetos. Por último, remarca la "confusión entre una clase y sus instancias, que lleva a los alumnos a usarlas indistintamente, enviando un mensaje a una clase cuando lo que corresponde es ubicar a la instancia concreta que debe recibir el mensaje y enviar el mensaje a esa instancia". [18]

La experiencia de los estudiantes

En un marco en el que *LOOP* es valorado ampliamente por los estudiantes, hay algunas características en las que se hace más evidente y otras en las que se lo usa con una naturalidad que no se percibe el impacto de su uso. Asimismo también se expresan algunas dificultades del software y se reflejan los diferentes perfiles de estudiantes.

En las evaluaciones que los estudiantes hacen sobre la materia al terminar la cursada, se destaca en primer lugar como positivo la posibilidad de practicar utilizando directamente las computadoras para construir y probar las soluciones y en relación directa con el *LOOP* se resalta la interfaz gráfica del diagrama de objetos. "El respaldo gráfico ayuda a entender lo que está pasando adentro", expresa uno de los estudiantes. Una afirmación que se reitera para

explicar en qué medida ayudó mucho a entender el paradigma, gira en torno a que "una cosa es entender la explicación del profesor pero otra cosa es verlo cómo funciona en la práctica". Otra estudiante expresa: "Me ayudó a comprender los conceptos de delegación y encapsulamiento, ver cómo no tenía que poner un comportamiento extenso en un solo objeto sino que tenía que ir delegándolo en los distintos objetos". "Facilita mucho la diferenciación de los conceptos de referencia y objeto, ayuda a concentrarse en los objetos y la interfaz que exponen, y no tanto *dónde* defino el comportamiento", explica Clara Allende una estudiante que luego se integró como docente de la materia.

Desde su experiencia personal como alumno de la materia, cuando aún no se utilizaba el *LOOP*, el ingeniero Alfredo Sanzo, uno de los actuales docentes, recuerda que "el alumno que no poseía conocimientos básicos de objetos encontraba una barrera inicial conceptual muy difícil de franquear: la creación de objetos mediante clases" y como en su planificación el tema *clase* estaba semanas después lo resolvía retrasando la práctica del lenguaje Smalltalk en las máquinas, "lo cual es pedagógicamente incorrecto en una materia de programación", concluye.

La dificultad señalada con mayor frecuencia es el pasaje de dejar de usar *LOOP* para enfrentarse al Smalltalk. Si bien el *LOOP* está integrado al IDE del lenguaje e intenta mantener cierta analogía, la experiencia de los estudiantes muestra que aún le falta suavizar didácticamente ese salto.

Otra situación que se produce es que en la heterogeneidad de conocimientos previos que traen los estudiantes, hay quienes ya tienen algún tipo de experiencia en la programación orientada a objetos. Entre ellos, se perciben a su vez tres modos diferenciados: para algunos "olvidarse" de las clases y pensar en objetos representa una dificultad más que una mediación didáctica y se muestran disconformes con su uso; otros tienen una actitud más neutra, asumiendo que si bien a ellos no les aporta lo mismo, entienden que puede ser de utilidad para sus compañeros; por último, los casos más interesante son los de quienes teniendo conocimientos previos, tener que volver a empezar y por ejemplo focalizarse en los objetos más que en las clases fue lo que les permitió "desaprender" para volver a aprender de otra manera: "Yo creía que sabía objetos..." se sincera uno de los estudiantes.

La utilización de LOOP

Lejos de pensar que se trata de atributos intrínsecos que por sí solos garantizan buenos resultados, como sucede con todo tipo de herramienta, hay que ver quién, cómo y para qué se la utiliza. Y es oportuno denominarla "herramienta" recuperando el valor artesanal que tiene su utilización. Apelando a su sentido originario como continuidad de la mano del obrero, del campesino o del artista, el *LOOP* es herramienta en tanto que depende de cómo, cuándo y para qué la utilice, en este caso el trabajador de la educación, que tiene a su cargo la conducción del proceso de enseñanza.

Por lo tanto, otro aspecto de este trabajo es analizar cómo se ubica *LOOP* en tanto software educativo dentro de la estrategia didáctica de cada docente. Aquí, en primer lugar, se ve que hay una tendencia común que presenta matices según qué docente la lleva adelante o cómo es el grupo de estudiantes. En términos generales, su uso va desde explicaciones en el aula con el proyector como apoyatura de la explicación, mostrando ejemplos y viendo sus resultados hasta prácticas de laboratorio en las que cada estudiante trabaja con su máquina y

resuelve a su manera algún problema planteado. Complementando su uso en el tiempo de clase, todos los docentes coinciden en pedir trabajos prácticos que los estudiantes deben realizar individualmente o en grupos en sus domicilios.

Quienes fueron los primeros en utilizarlo, en su doble rol de docentes y desarrolladores, afirman que el principal sentido del software es lograr que "los alumnos puedan experimentar rápidamente la aplicación de los conceptos de objetos, mensajes y relaciones, haciéndolos andar en una computadora y pudiendo aprender de los resultados que obtienen"[19]. En la actualidad, la mirada de los docentes acerca del sentido pedagógico de *LOOP* es coincidente. Desde el punto de vista de los contenidos, la característica principal que valoran es que sin necesidad de conocimientos previos en programación, permite crear, ver y entender qué es un "objeto", -el concepto fundante del paradigma- e inmediatamente enviarle "mensajes" -concepto que muestra el sentido, el "para qué" de dicho objeto. Con sólo estas dos ideas ya es posible avanzar en prácticas concretas, para poder ir incorporando otros conceptos, principalmente el de polimorfismo -tema central del paradigma de objetos- que de esta manera se lo puede empezar a comprender y poner en práctica casi desde el principio del dictado de la materia. También, progresivamente se pueden ir abordando otros conceptos, algunos más básicos, como el uso de variables, métodos, constantes y otros elementos sintácticos, y otros más profundos teóricamente hablando, como el manejo de referencias, la noción de estado, el mecanismo de clonación y la recolección de basura.

Lo que claramente no se puede hacer utilizando el *LOOP*, pero lejos de ser visto como una limitación es la frontera que le da sentido, es definir "clases". No se trata de una decisión arbitraria ya que hay un convencimiento de que "el objeto y el mensaje son los conceptos centrales en vez de las clases"[20], sino que se asume que una vez transcurrido cierto tiempo del desarrollo de la materia, habiendo entendido, madurado y puesto en práctica una serie de conceptos sin la necesidad de recurrir a las clases, el estudiante ya está en condiciones de dejar de lado el *LOOP* y pasar a usar el entorno del Smalltalk tal cual es, y en él empezar a definir clases para luego continuar con "herencia" y otros conceptos del paradigma.

Otro elemento destacado mayoritariamente es la interfaz gráfica y el diagrama de objetos. "Tener la representación visual ayuda a fijar el concepto de variable y de referencia" afirma Gisela Decuzzi, ingeniera y docente de la materia. Los diagramas son muy importantes en el primer acercamiento a la programación orientada a objetos, para que los estudiantes vean qué es lo que está sucediendo internamente. "Es muy valioso el diagrama de objetos que se actualiza dinámicamente en base al código ejecutado" apunta en la misma dirección Mariana Matos, también ingeniera y docente. En otros términos, es un claro ejemplo del concepto metafórico de "andamiaje" de Bruner [21], en tanto medio de brindar apoyo para que el estudiante pueda adquirir un nuevo conocimiento que en caso de no tenerlo le sería imposible, lo que va en profunda consonancia con la noción vygotskiana de *zona de desarrollo próximo*. Es un medio que además responde al carácter provisorio que señalan al respecto los pedagogos constructivistas, ya que cuando el estudiante comprende la dinámica de las referencias entre objetos y es capaz de aplicarlo en su práctica ya no tiene sentido seguir utilizando los diagramas de *LOOP* como recurso pedagógico.

Mirando lo sucedido en los últimos años en comparación con los anteriores o con lo que sucede en otras instituciones educativas, los docentes de la asignatura coinciden que con este recurso se llega a tener código funcionando mucho más rápido que por las formas tradicionales. Si bien la programación requiere de bases teóricas, conviene combinar ese aprendizaje teórico con una fuerte experiencia práctica, aprender la teoría sin poder aplicarla es mucho más complejo. Por eso, que el estudiante pueda construir programas sencillos desde la primera clase permite una asimilación mucho más rápida de los conceptos teóricos, ya que los va incorporando en sus programas inmediatamente.

El uso de *LOOP* tiene rasgos comunes entre los diferentes cursos y docentes: se lo utiliza desde el primer día en el que se enseña el paradigma de objetos, hasta alrededor de la quinta clase, sobre entre 10 y 12 dedicadas al paradigma de objetos. En ningún curso baja de tres clases y a lo sumo llega a la séptima en manos de los docentes que más han participado en el desarrollo del software. La diferencia más significativa de uso pasa por quienes llegan a utilizar la posibilidad de clonación de objetos, y por lo tanto dedican alguna clase más para ello, y quienes prefieren pasar directamente al concepto de clase como forma de generalizar comportamiento. Otro matiz surge de quienes hacen un uso mayor de las colecciones que incluye *LOOP* y quienes pasan más rápido por ese tema y se explayan luego utilizando el IDE de Smalltalk. Por otra parte, los casos de menor uso del *LOOP* suelen coincidir con los cursos nocturnos en los que aumenta la proporción de estudiantes con mayor experiencia laboral y conocimientos previos de programación.

Resultados

En primer lugar, los docentes de la materia, incluidos los desarrolladores del *LOOP*, coinciden en evaluar positivamente la herramienta -lo cual puede parecer obvio desde el momento en que la siguen utilizando- pero es importante analizar sus justificaciones.

El principal indicador que se observa y que da cuenta de un aprendizaje significativo por parte de los alumnos es que la respuesta que se va viendo en los estudiantes permite aumentar el nivel de complejidad de trabajos prácticos y exámenes, sin que ello signifique la fractura del proceso de aprendizaje, en términos generales. Un docente con años en la cátedra señala que "el nivel de complejidad de los ejercicios que podemos tomar es mucho más interesante, incluso incorporando decisiones de diseño de complejidad media o baja".

A su vez, el tipo de problemas que surge de la corrección de los exámenes es diferente, ya que hay ciertos errores que anteriormente eran más frecuentes y ahora lo son menos. "El impacto es positivo en general, aumentando el nivel de compromiso inicial y de esta manera cumple lo que se propone" dice Sanzo. "Los alumnos adquieren notablemente un conocimiento más preciso de conocimientos, lo cual permite incluir en el programa conceptos más avanzados", agrega, lo que es consistente con las modificaciones en las planificaciones de los últimos años respecto de los anteriores, en las que se incluyen temas que antes no aparecían.

La experiencia usando *LOOP* muestra que los estudiantes aprenden a programar en objetos más fácilmente y con un mayor nivel de utilización de las posibilidades del paradigma. En ese sentido, contar con un propio entorno de programación "nos permite seleccionar los conceptos de progra-

mación que queremos introducir en cada paso del proceso de aprendizaje, proporcionando una excelente terreno para la introducción gradual de los conceptos."[22]

Las opiniones de los estudiantes van en la misma dirección que el análisis efectuado por los docentes respecto de la aplicación de esta herramienta informática, afirmando que facilita el aprendizaje. Se percibe que los estudiantes están más cerca de comprender los conceptos del paradigma con una profundidad suficiente para poder aplicarlos efectivamente en situaciones concretas, y de esta manera se suaviza la curva de aprendizaje. "A su vez, esto acerca a los alumnos a tener la confianza suficiente para decidir usar los conceptos y técnicas incorporados en la materia en su práctica profesional posterior"[23]. En este sentido, también hay consenso en que un sólido conocimiento de los conceptos facilita una transición posterior a cualquier otro lenguaje de programación orientado a objetos.

En relación al contexto, es prematuro y excede los alcances de este trabajo, pero lo que realmente constituye el objetivo más preciado de la materia y en particular de esta herramienta, que es que el impacto se traduzca en una mejor calidad de software en el campo del ejercicio profesional de desarrollo de sistemas, como aporte significativo de la Universidad a la sociedad.

Discusión

Por otra parte, sin perder una mirada crítica sobre la propia práctica, los docentes son conscientes de los límites de la herramienta y perciben nuevas dificultades que dan pie a inquietudes de cambio y desarrollo de nuevas versiones de *LOOP*.

Respecto de lo engorroso que pueda ser la herramienta para aquellos que ya conocen de programación orientada a objetos y tienen experiencia con otras tecnologías, no se puede hacer demasiado con esta herramienta, ya que apunta a un perfil promedio de desconocimiento de dichos conceptos y en particular la prioridad para un uso más intensivo de *LOOP* la tiene el estudiante que presenta mayores dificultades.

Como se ha mencionado anteriormente, una dificultad que todos perciben es la transición del uso de *LOOP* al entorno clásico del Smalltalk que aún no termina de dejar tranquilos a los docentes. Uno de los desafíos es poder lograr que un mismo ejemplo que arranca en las primeras clases con objetos y prototipado se pueda continuar más adelante usando clases, "mediante algún *feature* que permita automatizar este cambio de modelo y no tener que volver a hacer el trabajo" señala el ingeniero Pablo De Haro, docente de la materia. En concreto, resolver mejor esta transición es un requerimiento que ya está presente en los planes para una nueva versión de la herramienta.

Otro aspecto que se analiza como posibilidad de inclusión al software, o al menos como una temática asociada que amerita una investigación más profunda, es el aprovechamiento de las nociones de *mixins*, *traits* o categorías como complemento al modelo de clases con herencia simple.

Conclusiones

La constatación que primero hacemos es que, mirando específicamente el software como tal, presenta características didácticas que facilitan el aprendizaje de los lenguajes de programación del paradigma de objetos, tales como la interfaz gráfica, la sencillez de la interacción, la posibilidad de utilización sin conocimientos previos en el paradigma.

Luego, lo que consideramos que brinda una mayor utilidad a la herramienta es el

momento, el enfoque y la forma de uso que se hace de ella, acorde a una ponderación de la importancia relativa de los temas de la asignatura desde la percepción del campo profesional, a las decisiones acerca del orden y gradualidad en que se presentan los temas y a la importancia que se le da a la resolución de problemas reales utilizando un lenguaje concreto como articulación entre teoría y práctica. Es desde este conjunto de opciones pedagógicas que tiene sentido utilizar *LOOP* como recurso para enseñar a programar en objetos, y es lo que posibilita el uso de esta herramienta en otros contextos educativos.

Por último, creemos que es destacable para valorar como experiencia replicable en otros contextos universitarios, el mismo ejercicio reflexivo y creativo de mirar la propia práctica, detectar una dificultad, construir una herramienta concreta -en este caso de software- que intenta superarla, ponerla en práctica y revisar su utilización para realizar nuevas actualizaciones.

Agradecimientos

A todos los integrantes del equipo docente de *Paradigmas* que colaboraron y participaron de la investigación, muchos de ellos anónimamente.

Referencias

[1] Cataldi, Zulma. "Metodología de diseño, desarrollo y evaluación de software educativo". Tesis de Magister en Informática. UNLP.
[2] Marques, Pere: (1998): *La evaluación de programas didácticos. Comunicación y Pedagogía*, Nº 149. Barcelona.
[3] Papert, Seymour. (1999): *Logo Philosophy and Implementation.* LCSI.
[4] Resnick, Mitchel. "Reviving Papert's Dream" en *Educational Technology*. Vol 52. N° 4. Julio-Agosto 2012.
[5] Utting S. y otros *Alice, greenfoot, and scratch – a discussion.*
http://web.media.mit.edu/~jmaloney/papers/AliceGreenfootScratch.pdf (Consultado 01/07/2013)
[6] Lombardi, Carlos. Passerini, Nicolás. Cesario, Leonardo (2007) "Instancias y clases en la introducción a la programación orientada a objetos". Smalltalks 2007 – Primera Conferencia Argentina de Smalltalk, 2007.
[7] Griggio, Carla. Leiva, Germán. Polito, Guillermo. Decuzzi, Gisela. Passerini, Nicolás. (2011) *A programming environment supporting a prototype-based introduction to OOP*
[8] https://sites.google.com/site/objectbrowsertool/ (consultado el 01/07/2013)
[9] Vasilachis de Gialdino, Irene (1992) *Métodos Cualitativos I. Los problemas teórico-epistemológicos.* Buenos Aires, Centro Editor de América Latina.
[10] Cataldi, Zulma. op cit.
[11] Lombardi, C y otros. Op cit.
[12] Liberman Neomi. Beeri, Catriel. Kolikant, Yifat Ben-David. *Difficulties in Learning Inheritance and Polymorphism*. Básicamente coincide en los problemas identificados, aunque la forma sugerida de resolverlo varía bastante, ya que asocia mucho más fuertemente el concepto de polimorfismo al de herencia, mientras Loop va a prescindir totalmente de la herencia para proponer una buena comprensión y utilización del polimorfismo.
[13] Ungar, David. Smith, Randall. *Self the power of simplicity.* Explica que los objetos simplifican las relaciones, ya que en un lenguaje con clases, además de las relacione de conocimiento entre objetos se tiene la relación de "es un" entre un objeto y su clase y otras para representar la herencia, en un lenguaje sin clases como self, todas las relaciones son de "conoce a". Esta idea es coincidente con el enfoque de Loop, salvando la distancia de que la presente herramienta no pretende ser un lenguaje por sí mismo sino un paso intermedio para finalmente incorporar la idea de clase.
[14] Wilkens, Linda. *Objects with Prototype-Based Mechanisms.* La autora valora la posibilidad de tener menos conceptos para facilitar el aprendizaje y señala que sin las clases igual se pueden lograr abstracción, encapsulamiento y modularidad.
[15] Freire, Paulo (2006) *Pedagogía de la tolerancia.* Fondo de Cultura económica. Compilación de Ana María Araujo Freire.
[16] Freire, Paulo (2009) *Pedagogía del Oprimido* Siglo XXI Editores. La "educación bancaria" es uno de los conceptos básicos de su pedagogía. Fue acuñado en esta obra, editada por primera vez 1970.
[17] Lombardi, C y otros. Op cit.
[18] Lombardi, C y otros. Op cit.
[19] Lombardi, C y otros. Op cit.
[20] Griggio y otros. Op Cit.
[21] Bruner, Jerome, (1997). *Celebrating Divergence: Piaget and Vygotsky.*
[22] Griggio y otros. Op Cit..
[23] Lombardi, C y otros. Op cit.