e-Book

Actas

Actas II Jornadas Argentinas de Didáctica de la Programación

Editores:

Araceli Acosta
Belén Bonello
Cecilia Martínez
Sonia Permigiani
Nicolás Wolovick

UNC FAMAF ffyh UNRC

ACTAS

II JORNADAS ARGENTINAS DE DIDÁCTICA DE LA PROGRAMACIÓN

Actas II Jornadas Argentinas de Didáctica de la Programación / Alejandro Iglesias... [et al.] ; editado por Araceli Acosta... [et al.].- 1a ed.- Córdoba : Universidad Nacional de Córdoba. Facultad de Filosofía y Humanidades, 2020.
Libro digital, PDF

Archivo Digital: descarga y online
ISBN 978-950-33-1600-9

1. Didáctica. 2. Lenguaje de Programación. 3. Formación Docente. I. Iglesias, Alejandro. II. Acosta, Araceli, ed.
CDD 004.071

COMITÉ ACADÉMICO

EDITORES

ILUSTRACIÓN DE TAPA

Manuel Coll – Área de Comunicación Institucional – FFyH – UNC

Wollok: Un entorno de aprendizaje de Programación Orientada a Objetos

Lucas Spigariol[1]

Introducción

Wollok es un software educativo para aprender a programar basado en el paradigma de la Programación Orientada a Objetos, desarrollado por docentes e investigadores de Universidades Públicas de Argentina, en el marco de la Fundación Uqbar. Desde una mirada tecnológica, se trata de un *entorno de desarrollo de software* que incluye un lenguaje de programación propiamente dicho, con un alcance conceptual similar a los lenguajes de alto nivel de uso profesional en la actualidad. Desde una mirada educativa, es un *entorno de aprendizaje* que propone una manera particular de llevar adelante un proceso educativo en programación [PFTD17], basándose en una herramienta concreta que permite articular fecundamente teoría y práctica.

La decisión medular que llevó a su desarrollo y utilización parte de priorizar las opciones pedagógicas por sobre las limitaciones tecnológicas. En otras palabras, en vez de diseñar itinerarios formativos que se adapten a las características que tienen las herramientas de desarrollo de software de uso industrial, se optó por desarrollar una herramienta tecnológica que se adapte a una concepción de qué, cómo y por qué enseñar a programar. Las principales características que presenta apuntan a abarcar un abanico de situaciones que van desde estudiantes sin experiencia previa a quienes se quiere introducir en el pensamiento computacional (interfaz gráfica interactiva, diagramas de objetos, abstracciones simples, estructuras de control básicas, objetos autodefinidos, reportes de errores en español, etc.) hasta estudiantes que tienen como meta su inserción laboral en el ambiente de desarrollo de software (tests unitarios, integración con *Git*, manejo de clases, herencia y mixins, sistema de inferencia de tipos, etc.)

Las forma recomendada de usar Wollok es como aplicación de escritorio, aunque existe una alternativa de ejecutarlo por línea de comando de reciente creación (Wollok CLI) y también es posible usarlo desde la plataforma educativa on line *Mumuki*, donde hay guías interactivas de ejercicios, problemas y desafíos que ayudan a sostener el proceso de aprendizaje.

1 Universidad Tecnológica Nacional, Universidad Nacional General San Martín, Fundación Uqbar
lspigariol@gmail.com

Origen y fundamentación

A pesar de ser la forma de construir software más utilizada en el ambiente profesional, la Programación Orientada a Objetos no suele ocupar un lugar significativo en las propuestas pedagógicas sobre programación. Esta decisión puede tener diferentes motivaciones, entre ellas la percepción de que es difícil de enseñar [BC04, Jen02, Uys], porque requiere del estudiante la capacidad de manejar muchos conceptos incluso para hacer soluciones sencillas, o que suele ser necesario aprehender un volumen considerable de teoría antes de poder hacer una práctica concreta [KQPR03, SGURGH15], lo cual lleva a considerarlo como un contenido lejano a la educación básica y media, e incluso en los trayectos de educación superior se la suele presentar como un contenido más avanzado.

La convicción que condujo a crear *Wollok* es que la dificultad no está en el paradigma en sí, sino en la elección de recorridos didácticos que no permiten una incorporación gradual de sus nociones centrales a lo largo del curso y en cambio obligan a que un estudiante sea capaz de manejar un amplio abanico de conceptos teóricos antes de poder producir su primer programa [CO16]. Estos recorridos, sin embargo, están frecuentemente limitados por la utilización en el ámbito del aprendizaje de lenguajes de programación y otras herramientas pensadas para profesionales, que naturalmente priorizan maximizar el rendimiento de un experto por sobre facilitar el acceso a un principiante [ACS02, GF03].

Frente a esta dificultad, la reflexión pedagógica fue que la secuencia didáctica debía articular teoría y práctica desde un primer momento y que no debía restringirse debido a las herramientas tecnológicas existentes, sino al revés: que el lenguaje de programación sea el que vaya acompañando y sosteniendo el itinerario formativo.

Un conjunto de asignaturas de programación en carreras universitarias de informática, donde confluyen la percepción de la dificultades mencionadas en los procesos de aprendizaje de los estudiantes con la capacidad profesional de los docentes de poder crear los propias herramientas tecnológicas, constituyó un espacio fecundo para el surgimiento de *Wollok*, fue un ambiente contenido para probar su funcionamiento, es el origen de una lluvia de ideas para mejorarlo permanentemente y sigue siendo un semillero de vocaciones de investigación que creen que tiene sentido apostar a un desarrollo tecnológico autónomo y de calidad desde la Universidad y el Estado.

En su origen, el perfil para el que se pensó la herramienta es el de un estudiante universitario sin experiencia previa en programación orientada a objetos. La realidad de los planes de estudios de las carreras universitarias en las que se comenzó a utilizar *Wollok* condiciona que quienes llegan a las asignaturas donde se aprende a programar en el paradigma de objetos tienen como correlativas previas materias donde se enseñan nociones básicas de programación, con mayor o menor intensidad y utilizando diferentes lenguajes según la institución, pero en todos los casos en términos de programación procedural. De esta manera, conceptos habitualmente considerados iniciales como por ejemplo estructuras de control condicionales, manejo de variables y constantes o paso de parámetros, que sin

duda se podrían explicar utilizando *Wollok* si el contexto fuese otro, no se lo hace porque no es necesario y los docentes se limitan a especificar la sintaxis con la cual se lo expresa. Incluso más recientemente, con la incorporación de *Wollok* en escuelas secundarias técnicas, se da una situación similar ya que si bien son estudiantes de menor edad, también pasan previamente por materias de otras formas de programar antes de abordar el paradigma de objetos.

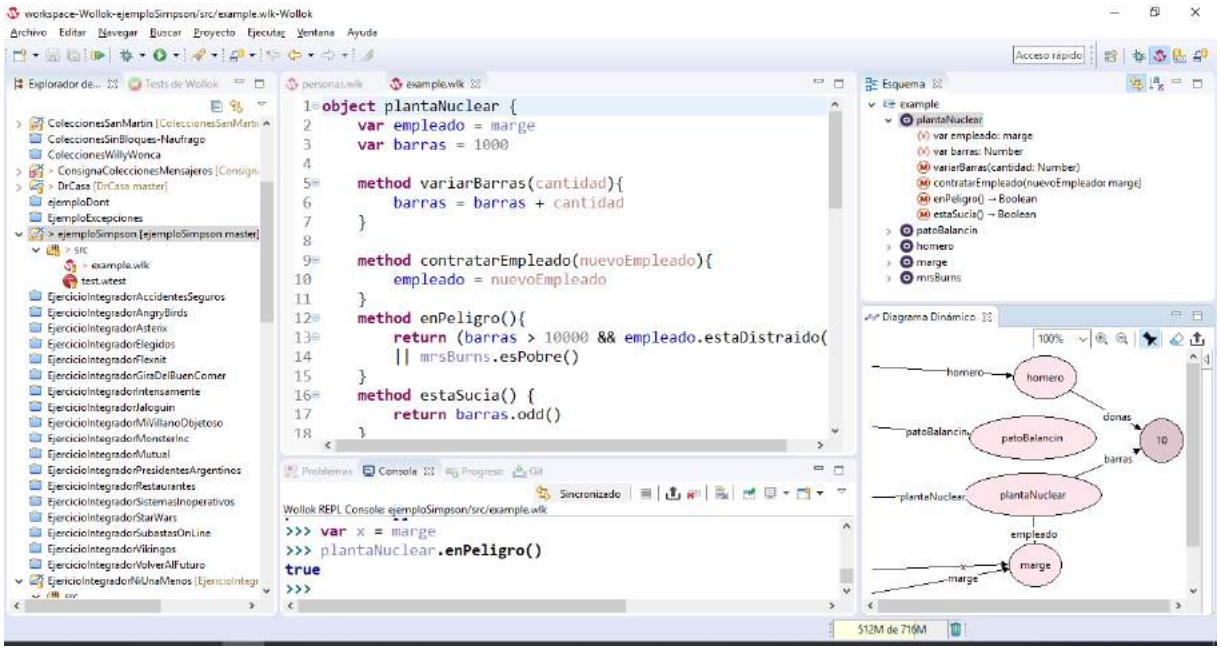
No es el objetivo de este trabajo establecer un discusión acerca de si la mejor forma de introducir a los estudiantes en el mundo de la programación es en términos de la programación procedural o de objetos -o por qué no utilizando conceptos de otros paradigmas como la programación funcional o lógica-, pero sí se busca que, para quien quiera experimentar que la primera aproximación a la programación sean en con objetos, conozca una herramienta concreta que le permita hacerlo, evitando las complejidades que tienen las herramientas profesionales de desarrollo de software en este paradigma, que están diseñadas para aumentar la eficiencia de quien ya sabe y no para facilitar el aprendizaje como se propone *Wollok*.

Wollok se viene utilizando desde hace cinco años en materias iniciales de programación en diferentes universidades (actualmente en UTN, UNGSM, UNQ, UNDAv, UNaHur) y también en algunas escuelas medias de orientación técnica.

Como antecedente directo, se desarrolló y utilizó durante unos años también una herramienta de software llamada Ozono, cuyo objetivo y fundamentos pedagógicos eran similares, pero que en vez de ser un lenguaje en sí mismo se ejecutaba dentro del ambiente de desarrollo de Smalltalk y se preveía su uso durante parte del curso para luego dejarlo y continuar con Smalltalk propiamente dicho. Su uso fue satisfactorio [Spi15], tuvo varias denominaciones a medida que cambiaba de versión (Object Browser, Loop), pero se discontinuó su uso ante la existencia de *Wollok.*

Descripción de la herramienta

Como primera aproximación, *Wollok* es un lenguaje de programación en el sentido típico del término: Con él se pueden escribir programas, se utiliza una determinada sintaxis, su código se ejecuta, se modelan soluciones y se obtienen resultados. Técnicamente es un intérprete y corre sobre diferentes sistemas operativos; desde una mirada docente, abarca los conceptos centrales del paradigma de objetos (objetos, mensajes, atributos, clases, colecciones, polimorfismo, herencia, redefinición) como también otros que se pueden considerar como más avanzados o que rondan los límites del paradigma (mixins, interfaces) y a la vez incluye elementos transversales a los diferentes paradigmas de programación (tests unitarios, manejo de excepciones, interfaz gráfica).



Vista general de Wollok, en la que se ve una porción de código con objetos y métodos, la consola REPL y el diagrama dinámico.

Hilando más fino, *Wollok*, más que un *lenguaje de programación*, es un *entorno de aprendizaje y desarrollo de software*. Existe una concepción en ciertos ambientes académicos que asocian casi unívocamente la tarea de programar al uso de un lenguaje de programación; en otras palabras, que ve al programa escrito como un producto y a su proceso de desarrollo como una tarea de escritura. En este enfoque, el ambiente de desarrollo es un simple editor de texto -que puede ser fácilmente sustituible por el pizarrón en una clase y por una hoja de papel en un examen- y que en el mejor de los casos viene complementado con el compilador adecuado que a modo de caja negra transforma ese código en un programa ejecutable. En caso que el código sea correcto, se ejecuta el programa y se observan los resultados, en caso que haya errores, el compilador advierte sobre su ubicación y naturaleza para que el programador reescriba los cambios necesarios y vuelva a compilar. En la dinámica moderna de desarrollo de software, se destacan los entornos integrados de desarrollo (IDEs) que no sólo permiten escribir código, sino que incluyen diversas herramientas que facilitan la tarea de programar. En este sentido, *Wollok* viene integrado en el entorno Eclipse, con una configuración especialmente diseñada en la que se aprovechan algunas de sus características, se ocultan otras para evitar complejidad innecesaria y a la vez se incorporan las específicas del lenguaje.

Puntualmente, algunas de estas características que facilitan el aprendizaje, y que lo diferencian de otras herramientas, son:

- Detección de errores, advertencias y sugerencias a medida que se escribe el código, lo que brinda un feedback instantáneo.

- Consola REPL para consultas interactivas, sin necesidad de salir del entorno de desarrollo.
- Integración con herramienta para definir y ejecutar pruebas unitarias.
- Diagramas que se generan automáticamente: El estático, que se basa en el código del programa escrito y muestra la estructura de los objetos y clases, sus atributos y métodos, las relaciones de herencia, entre otras características. El dinámico, que se va modificando conforme se van ejecutando diferentes partes del programa desde la consola interactiva. Ambos diagramas son representaciones gráficas de conceptos fundamentales en el paradigma, que acompañan y refuerzan su comprensión.
- Interfaz gráfica para hacer videojuegos.
- Sistema de inferencia de tipos, que permite realizar sugerencias y detectar inconsistencias entre la definición y uso de los diferentes módulos del programa, sin necesidad de un tipado explícito.

Pero para dar un paso más acerca de lo que es *Wollok*, y el sentido que tiene contar con estas herramientas para llevar adelante un proceso de aprendizaje, cabe hacerse algunas preguntas sobre qué y cómo enseñar.

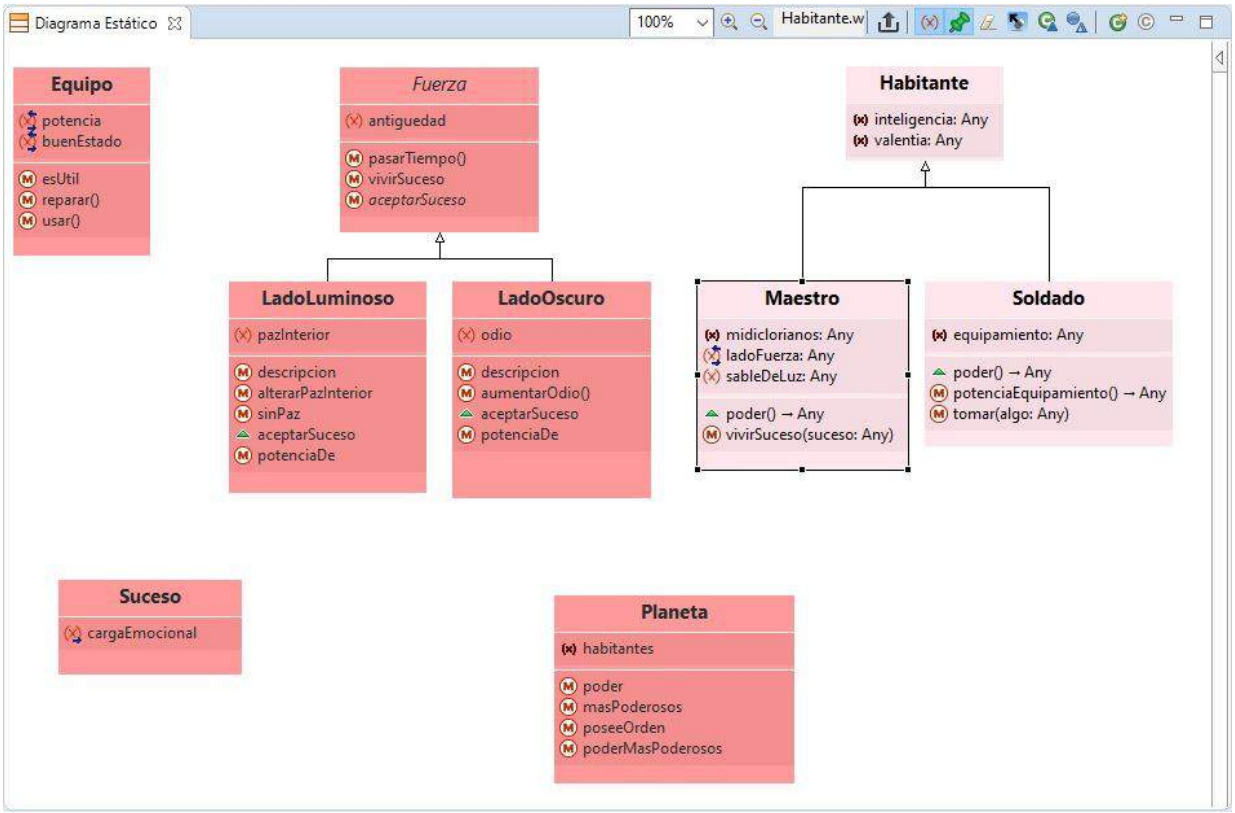
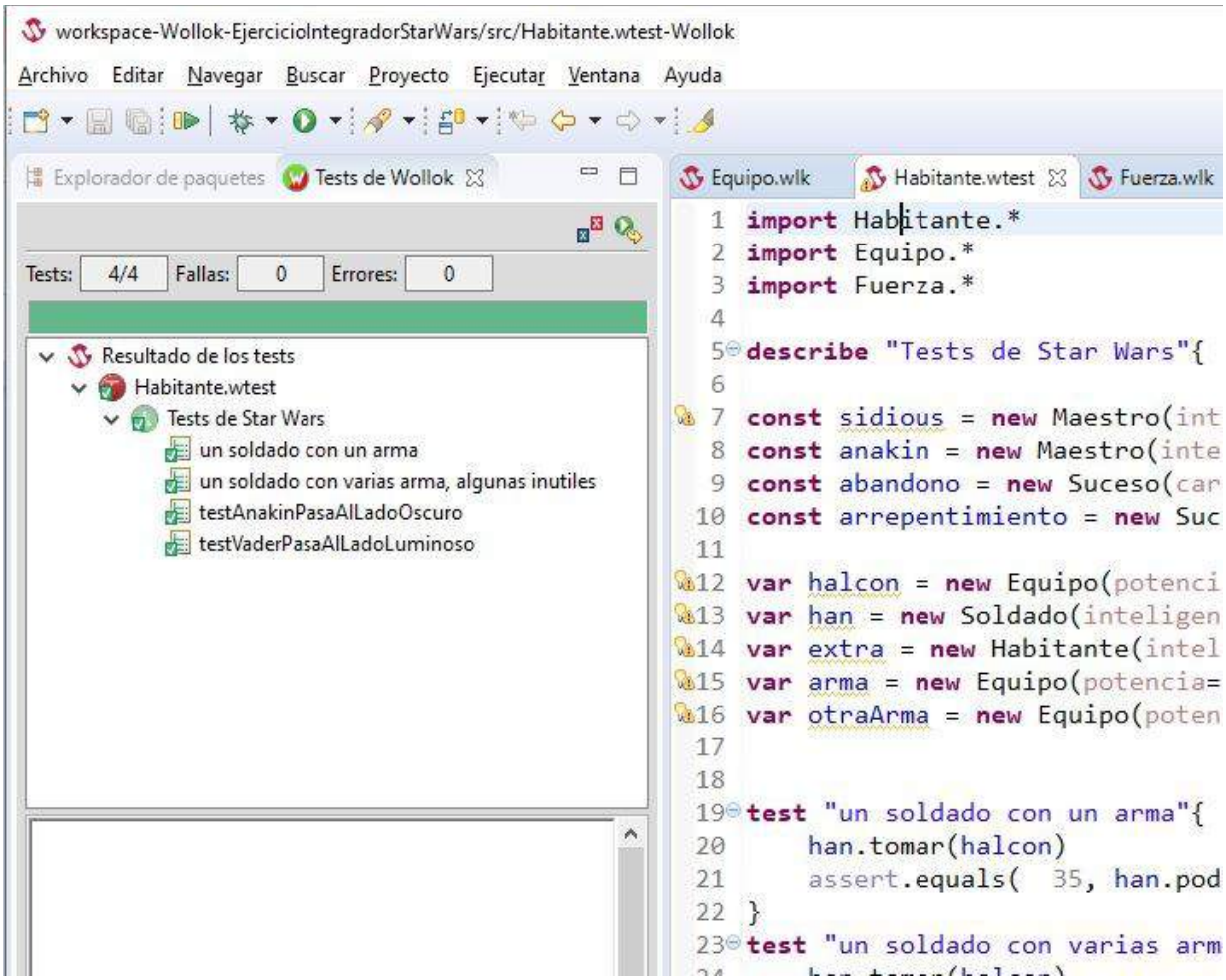


Diagrama estático de un ejercicio, donde se ven las clases con sus atributos y métodos y las relaciones de herencia.



Herramienta para pruebas unitarias. Resultado de la ejecución y porción de código de su definición.

¿Qué enseñar?

Dentro del amplio abanico de nociones y destrezas propias de la programación, tanto de las cuestiones típicas de la programación orientada a objetos como de otras formas de programar, se realiza una cierta selección y ponderación.

Se destaca una de las intuiciones originarias del paradigma de objetos de disminuir el *gap semántico* entre la realidad y el modelo, es decir, desde la perspectiva del estudiante, que las entidades que están en el programa le resulten lo más cercanas y representativas posible a las entidades del mundo real en que se basan. Por esto, se presenta al *objeto* como elemento central, que más que una entidad inerte o simple contenedora de información se lo interpreta como *sujeto*, con comportamiento e identidad, que interactúa con otros, que *conoce* mediante sus atributos, *pregunta* al enviar mensajes y *colabora* al ejecutar un método. A su vez, al objeto se lo rescata en su originalidad e individualidad, sobre la que eventualmente en un paso posterior se puede generalizar a partir de descubrir similitudes con otros, antes que pensarlo como un caso particular de una generalización abstracta de existencia previa y alcance predeterminado.

Pensar a un programa de computación como un conjunto de *objetos/sujetos* que interactúan, es una forma de modelar la realidad que, además de constituir una metáfora de la sociedad que dispara miles de preguntas y conecta con otras disciplinas, resulta relativamente sencilla de comprender y constituye un forma de pensamiento computacional con diferentes niveles de complejización.

En concreto, *Wollok* permite programar fluidamente con *objetos autodefinidos* sin necesidad de recurrir inicialmente al concepto de *clase,* como suele suceder en las propuestas pedagógicas existentes. Esto va de la mano de la intención de mostrar las virtudes del paradigma de objetos sin caer en las inercias procedurales que generalmente provienen de entender la *clase* como una simple estructura de datos. Además, que los objetos tengan un identificador conocido permite trabajar con ellos sin preocuparse aún por el manejo de referencias y el uso de variables, como sucede en lenguajes más tradicionales como Smalltalk donde no se puede interactuar con un objeto sin antes tener una referencia.

Otra de las premisas de *Wollok* es la de un tipado dinámico que permite un manejo de *polimorfismo* independientemente de la *herencia*, *interfaces*, *clases* ni ninguna definición de tipos de variables. La convicción acerca de la importancia del *polimorfismo* como forma de entender las interacciones entre objetos, tanto como metáfora de la no discriminación y la capacidad de ponerse en el lugar del otro, como de su centralidad a la hora de construir soluciones informáticas profesionales, lleva a que se permita su utilización con un mínimo de elementos conceptuales previos y se lo ponga inmediatamente en práctica. A su vez, la facilidad que representa para un estudiante no tener que definir previamente el *tipo* de todo elemento del código, se complementa con un poderoso s*istema de inferencia de tipos* que informa sobre la formas de uso correctas y advierte posibles errores.

Aún sin ser lo que sucede dada la utilización actual de *Wollok* en situaciones donde se busca otra cosa, características tales como la forma de definir y asignar en variables, la posibilidad de tener una lista de elementos mediante una sintaxis sencilla y poder mezclarlos sin estar condicionados por su tipo, la existencia de un *if* y operadores lógico y matemáticos, también habilita su uso para quien no quiera profundizar en conceptos más específicos del paradigma de objetos.

¿Cómo enseñar?

Hay varios tópicos en los que se podría dar una respuesta. Se mantienen criterios de simplicidad en la sintaxis y facilidades de usabilidad en el entorno de desarrollo, se establece una fuerte asociación entre conceptos y palabras reservadas, se cuida la expresividad y localización de los mensajes de error que se suponen frecuentes en un estudiante que da sus primeros pasos, entre otras. Pero en particular se destacan dos características, que a su vez habilitan a diferentes formas de uso: el ordenamiento de la secuencia didáctica y la utilización de herramientas gráficas.

Secuencia didáctica

Desde una perspectiva de itinerario, el recorrido propuesto para el cual *Wollok* brinda facilidades, se organiza en cuatro etapas con temas centrales de complejidad creciente (*objetos, colecciones, clases y herencia*) y en todas ellas hace énfasis en la importancia del *polimorfismo*.

Una facilidad, sobre todo para los primeros ejercicios en objetos con los que se pretende aprender a modelar *objetos* y *mensajes*, es la posibilidad que los objetos tengan valores iniciales para sus variables sin que se les envíe ningún mensaje previamente. Permite además que los primeros mensajes que se usen sean sin *parámetros* y que en un posterior ejemplo se los incorpore. También, se ve la secuenciación en el manejo de *colecciones*. En otros lenguajes, para tener una *colección vacía* a la que se le pueda ir agregando elementos se debe manejar el concepto de *clase*, instanciarla indicando su nombre y enviándole un *mensaje de clase* llamado *new*. Lo que permite *Wollok* es una inicialización de colecciones sin necesidad de *métodos*, detallando los valores necesarios junto con la definición de las variables, y una sintaxis sencilla que recuerda a la notación de listas de otros lenguajes y conjuntos matemáticos. De esta manera, si el docente así lo prefiere, puede trabajar tranquilamente con *colecciones* antes que explicar el concepto de *clase*.

En muchas situaciones como esta, el docente se encuentra en la tensión entre adelantar la presentación de otros conceptos aún para hacer cosas simples, o a requerir de los estudiantes un acto de fe: por más que no entienda qué es o qué significa una cierta expresión, "hay que escribirla así para que funcione y listo". Frente a esto, *Wollok* evita incluir en su formulación expresiones arbitrarias o innecesarias, mantiene una marcada economía sintáctica sin sacrificar expresividad y, sobre todo, sugiere que en su código no se anticipen respuestas a problemas que aún no hayan surgido. Con criterios similares, el *polimorfismo* no está condicionado por la *herencia*, la *instanciación* no requiere de un *constructor* definido previamente, los objetos se pueden autorreferenciar sin necesidad de usar *self* o similar, y hay variantes para expresar el manejo de *excepciones* que permite introducir el concepto en diferentes momentos de la secuencia didáctica, entre otras decisiones.

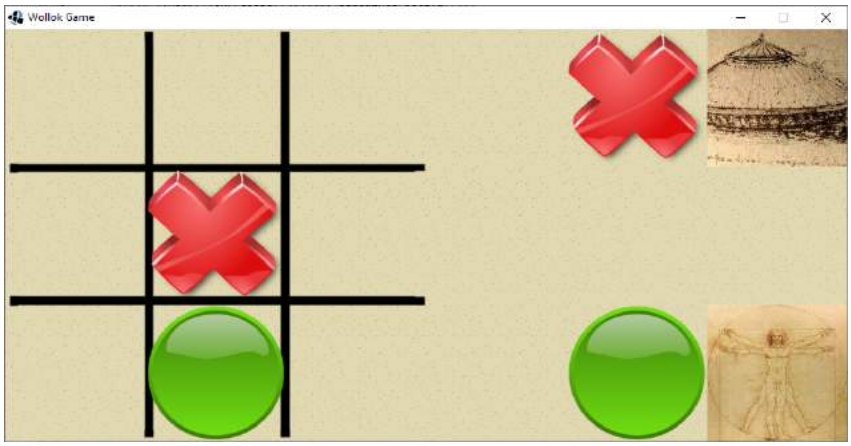
Elementos gráficos

Desde una perspectiva de proyectos, otra forma de utilización es mediante *Wollok Game*, una interfaz gráfica que viene integrada en la herramienta, que permite construir juegos y aplicaciones interactivas de menor o mayor complejidad y desde allí ir descubriendo y aplicando los diferentes conceptos del paradigma. Se basa en la asociación de los objetos presentes en el código con imágenes y posiciones en la pantalla de manera que la ejecución del programa sea visible a la vez que permite al estudiante interactuar con su creación como un usuario, vinculando los eventos con el código. Incluye facilidades propias de los

videojuegos tales como los desplazamientos y colisiones de elementos visuales, el manejo del tiempo, la interacción con el usuario final, la incorporación de música y sonido.

Otro aspecto visual importante, son los ya mencionados diagramas estáticos y dinámicos, que se generan automáticamente desde el código. El diagrama dinámico muestra el estado del sistema en todo momento reflejando las relaciones entre objetos y su evolución en el tiempo. Permite, por ejemplo, ver cómo las variables son referencias que apuntan a otros objetos, cómo al asignarle un valor la flecha del diagrama se mueve, cómo un objeto puede ser referenciado desde muchos lugares a la vez. Complementariamente, el diagrama estático muestra las clases y objetos definidos, permite que el estudiante visualice de un pantallazo general la organización de lo que está escrito en el código, comprenda mejor lo que está haciendo y pueda modificarlo con mayor agilidad. Se basa en el estándar de UML, refleja las relaciones definidas explícitamente, como las de herencia, y la potencialidad -es un feature en desarrollo- de integrarse con el sistema de tipos e inferir otras relaciones, como las de asociación o composición.

Ventana de un juego sencillo, hecho con Wollok Game por un grupo de estudiantes.



Una de las ventanas de un juego más complejo, hecho con Wollok Game por estudiantes, con diferentes niveles, personajes, herramientas y una lógica compleja.

Resultados

El feedback que se recibe de docentes y estudiantes es significativo y variado. En primer lugar están los reportes de *bugs,* que no sólo son útiles para resolver los problemas de implementación específicos a los que hacen referencia, sino que en cierto modo son un termómetro de la intensidad de su uso.

Más interesantes son las propuestas de mejoras y pedidos de nuevos features, como así también los comentarios sobre la aceptación o desacuerdo con las funcionalidades que se van agregando año tras año, que enriquecen la dinámica de desarrollo del proyecto y permiten ver la ampliación de docentes e instituciones que deciden utilizar o al menos probar *Wollok*.

Las evaluaciones y encuestas de los estudiantes sobre el desarrollo de los cursos en los que se enseña a programar con *Wollok* son generalmente muy buenas. Esto obviamente habla más de los respectivos docentes y su forma de llevar la materia que del lenguaje en sí, pero al menos muestra que su uso no es un motivo fuerte de desaprobación que arrastre la valoración de la asignatura. Cuando se trata de instrumentos elaborados en forma genérica por la institución educativa no es posible, pero en algunos casos en que se trata de una iniciativa del mismo docente y el formulario incluye alguna pregunta directa sobre *Wollok*, -o cuando se hace un balance de la cursada en forma oral- el resultado es ampliamente positivo. Las críticas que se encuentran, dejando de lado ciertos comentarios en referencia a bugs o situaciones puntuales, generalmente van por el lado de que se hubiese preferido un lenguaje comercial/industrial/profesional. La opinión de los mismos estudiantes como sujetos de su propio proceso de aprendizaje es valiosa como indicador de la utilidad de la herramienta pero no determinante por sí solo. Saliendo de los extremos de un eventual rechazo mayoritario o de una inimaginable actitud de admiración que no se constatan en la realidad, lo más valioso del feedback de los estudiantes es la posibilidad de hacer mejoras o ajustes dentro de la misma propuesta. Sin descartar que algunos con un poco más de experiencia lo puedan hacer, no se espera que el estudiante promedio tenga la suficiente distancia para analizar cómo hubiera sido el desarrollo de la materia utilizando otro lenguaje y sacar una conclusión al respecto. (Al menos al momento de concluir la materia, sería interesante su punto de vista años después, ya inmerso en el mundo profesional, habiendo tenido experiencias de otras herramientas de desarrollo).

Otro camino posible para validar científicamente el impacto de *Wollok* en el proceso de aprendizaje es entrar en el terreno cuantitativo de los análisis comparativos. Aún reconociendo las limitaciones de su implementación, lo discutible de sus eventuales resultados y ciertas discrepancias epistemológicas acerca de este tipo de metodologías en el ámbito educativo, hay algunos estudios en marcha de los cuales no se disponen resultados, pero que se espera puedan enriquecer el análisis en un futuro.

Teniendo en cuanta las consideraciones, posibilidades y limitaciones mencionadas, para el presente estudio, los indicadores que se consideran prioritarios a la hora de plasmar el

resultado de utilizar *Wollok* como herramienta que da soporte a la propuesta formativa son la convicción de parte de los docentes y el aumento de complejidad de las producciones de los estudiantes y el aumento de exigencia de las evaluaciones sin disminuir los niveles de aprobación. Ambos combinan un componente cualitativo importante sujeto a la intepretación con datos mensurables.

Convicción docente

De todos los docentes que habiendo utilizado antes otras herramientas pasaron a usar *Wollok*, ninguno volvió atrás o buscó otras alternativas superadoras, sino que aún reconociendo limitaciones apuestan a mejorarlo. Para dimensionar el escenario, la institución que más docentes -y estudiantes- tiene utilizándolo es la Facultad Regional Buenos Aires de la Universidad Tecnológica Nacional. Cada año, son alrededor de 15 comisiones las que se llevan adelante, cada una con su respectivo docente a cargo más un conjunto de ayudantes, que juntos conforman un equipo docente de más de un centenar de integrantes; sumando a todas las demás instituciones educativas, el volumen total aproximadamente se duplica. Más allá de los pormenores, como que la decisión de comenzar a usar *Wollok* no fue exenta de discusiones y objeciones iniciales, que en algunas instituciones fue tomada antes que en otras, que hay algunos docentes que trabajan en más de una institución a la vez, que en los equipos docentes hay una minoría que participa en el desarrollo mismo de *Wollok* y una mayoría que sigue el proceso a la distancia, con mayor o menor grado de cercanía, que los plantes de estudios vigentes no estipulan qué lenguaje utilizar o no, que casi la totalidad de los docentes trabaja a su vez en la industria del desarrollo de software y conoce de primera mano otros lenguajes de programación orientada a objetos, que existe la libertad de cátedra en todas las instituciones como para dar marcha atrás o tomar otros caminos, lo que sucedió es que todos los docentes involucrados lo continúa utilizando.

Complejidad de las producciones de los estudiantes

Se constata un aumento significativo de la complejidad de los exámenes y trabajos prácticos que realizan los estudiantes, lo cual va de la mano de la ampliación de los alcances temáticos de las planificaciones de las asignaturas, sin aumentar el tiempo neto de clase ni variar significativamente los porcentajes de aprobación.

Comenzando por esto último, es frecuente ver en las planificaciones actuales clase enteras destinadas a discusión sobre modelado en objetos y elementos de diseño, con temas tales como "Herencia vs Composición", "Clase abstracta e Interfaz", "Mutabilidad. Igualdad e identidad", "Template method", "Acoplamiento", "Introducción al Strategy", que antes no estaban presentes o eran mencionados lateralmente. Se agrega también como tema importante el testeo automatizado y el manejo de excepciones, con clases destinadas a "Testeo unitario automatizado avanzado", "Fixture y Clases de equivalencias " o "Lanzar y capturar excepciones".

Sobre los límites del paradigma, otros temas que aparecen al final del recorrido del curso, y con diferencias según el docente, el año o la institución, apuntan a hacer comparaciones con lenguajes industriales, establecer vínculos con otros paradigmas de programación, introducir a conceptos más novedosos dentro del mundo de objetos -por ejemplo, mixins, que si bien *Wollok* los implementa, habitualmente no forma parte del contenido curricular de las asignaturas- o profundizar algún concepto puntual como esquemas de tipado. Cabe señalar que este tipo de clases ya se realizaban en algunos casos antes de que se utilice *Wollok*, pero en mucho menor cantidad que los que sucede en la actualidad.

Esta profundización y ampliación de los alcances temáticos, que en buena medida es el correlato del menor tiempo requerido para que se comprendan las nociones básicas, se traduce en exámenes, ejercicios y trabajos prácticos de mayor complejidad. Siendo una variable de análisis de dificultosa cuantificación, el criterio elegido es la interpretación de los propios docentes, que en definitiva son quienes hacen un ejercicio subjetivo similar a la hora de definir la aprobación de cada uno de ellos. Un conteo de líneas de código en los repositorios donde se suben los trabajos o de la cantidad de clases y objetos que intervienen en la solución podría hacerse, y probablemente mostraría que el volumen aumentó, pero es más valioso analizar la complejidad del modelo implementado y los criterios de validación. Con mirada retrospectiva, lo que mayoritariamente ellos constatan es que los exámenes son realmente más difíciles, incluyen más temas o son más exigentes a la hora de establecer el mínimo de aprobación, prestando mayor atención al modelado, a las abstracciones formuladas, la expresividad del código o la economía del código. Una minoría no percibe mayores diferencias, pero en ninguno de los casos consultados se manifiesta una disminución. Otro aspecto que algunos señalan, sobre todo en los trabajos prácticos con consignas más abiertas de desarrollo, es la grata sorpresa de encontrarse con producciones de gran nivel.

En paralelo, y considerando en esta oportunidad lo datos estadísticas de porcentajes de aprobación -tomados por institución y por docente-, no han variado significativamente desde que se usa *Wollok* en relación a lo que sucedía antes. Los pequeños matices se distribuyen de manera irregular por lo que se atribuyen a otros factores y no al uso del lenguaje.

Conclusiones

Sin dejar de ser un *lenguaje de programación*, o un *entorno de aprendizaje y desarrollo de software Wollok* es una propuesta pedagógica para la enseñanza de la programación orientada a objetos. En otras palabras, es el *andamiaje* que sostiene y posibilita un proceso de construcción de conocimiento. Ciertamente, existen numerosos lenguajes y herramientas con las que se programa profesionalmente en objetos y todas ellas se pueden utilizar para enseñar. Pero ante las preguntas planteadas anteriormente de *qué* y *cómo* enseñar, sumando el *para qué* hacerlo, las posibilidades o facilidades que ofrece cada uno de los lenguajes existentes es diferente. Precisamente, lo que da sentido a *Wollok* es una respuesta pedagógica que es previa a la creación de la herramienta y constituye su inspiración.

Cuando se piensa la enseñanza del desarrollo de software como uno de los ejes centrales de una formación profesional, una disyuntiva que aparece, y que condiciona los objetivos y las herramientas para abordar la programación orientada a objetos en particular asumiendo que es una etapa dentro de un recorrido más amplio, es entre tener una propuesta de formación relativamente más reducida que apunte una salida laboral rápida o una propuesta más extensa que permita transitar con mayor detenimiento los temas y conceptos involucrados. Más allá que sea una discusión con mucha aristas y que en la actualidad conviven trayectos académicos que se orientan en uno y otro sentido, no deja de ser una discusión que suele aparecer en los equipos docente a la hora de definir una planificación y elegir las herramientas a utilizar, y muchas veces no tiene que ver directamente con que sea una licenciatura, ingeniería o tecnicatura, o que la institución que la lleva adelante sea una universidad, un instituto terciario o un secundario.

Si lo que se busca es que el estudiante tenga rápidamente las mínimas herramientas para buscar una posibilidad de inserción laboral, probablemente *Wollok* no sea una buena opción, difícilmente un seleccionador de personal valore ese ítem en un CV y el estudiante mismo sea consciente que aún habiendo aprendido conceptos de programación le falte hacer una cierta transición e incorporar otros elementos para poder aplicarlo de manera inmediata.

La finalidad detrás de *Wollok* es formar en competencias, habilidades y conceptos de suma importancia en la industria del desarrollo de software, con una propuesta que prioriza los conceptos, el razonamiento y los fundamentos necesarios para la formación de un profesional de calidad para desempeño en la industria. Partiendo de estas ideas, sin dudas que los caminos son numerosos, diferentes y que las herramientas disponibles en el mundo de sistemas son variadas. Por un lado, lo que generalmente se hace -y quienes actualmente utilizan *Wollok* también antes lo hicieron con otros lenguajes- es habiendo elegido un lenguaje de programación de uso profesional, adaptar el recorrido temático en la medida que dicha herramienta lo permita, con las limitaciones o complicaciones que ello conlleva. En un abanico de lenguajes -y a riesgo de simplificar en exceso- que va desde un Smalltalk, más puro de objetos y menos comercial hasta un ampliamente difundido JavaScript con conflictiva identificación con el paradigma de objetos, pasando por Ruby, Java, C++ o Python, por nombrar algunos, la elección del lenguaje está dada por encontrar un equilibrio razonable entre diversos factores: algunos más pedagógicos y otros más profesionales, sin ignorar otros planos que conforman el contexto de decisión, como el deslumbramiento o recelo a las nuevas tendencias, la disponibilidad de documentación, los conocimientos previos de los docentes, las inercias institucionales, etc. Otra alternativa, que es la que dio origen a este lenguaje, es partiendo de una cierta selección, ponderación y organización de una propuesta educativa, construir un lenguaje que la sostenga, acompañe y facilite. Retomando lo anterior, el equilibrio pretendido surge de ponderar fuertemente el aspecto didáctico por sobre otros, con un guiño a las nuevas tendencias en materia de desarrollo de software que *Wollok* implementa.

Por otra parte, pensando la enseñanza de la programación como parte de una formación integral que pretende que toda persona adquiera formas de modelar la realidad, pueda

desarrollar un pensamiento abstracto y sea capaz de apropiarse de la tecnología que lo rodea, *Wollok* propone un espacio fecundo y a la vez contenido para la experimentación, el juego y desarrollar el potencial creativo y creador de los estudiantes. Si bien está aún muy lejos herramientas educativas como puede ser Scratch o Pilas Bloques, por citar algunas que tienen un notable esfuerzo de acercamiento al lenguaje y a la psicología infantil, podría analizarse la pertinencia de utilizarla con estudiantes con un poco más de experiencia, no en programación estrictamente, pero sí en utilización de software, con más autonomías, con mayor capacidad de lectura e interpretación. En este sentido, lo que aporta *Wollok* de diferente es la metáfora de los objetos que se envían mensajes como forma de representar la realidad -que es la esencia del paradigma de objetos- de una forma más accesible y sencilla que lo que tal vez se imagina quien conoce los lenguajes profesionales de objetos y no quiere eso para sus estudiantes.

Los resultados reflejan que los colectivos docentes estudiados han encontrado en la herramienta una respuesta satisfactoria para sus búsquedas educativas y allí radica su motivación para continuar utilizando *Wollok* y mejorándolo. Su posible extrapolación a otros contextos educativos o eventualmente su apropiación por parte de otros docentes e instituciones educativas, si bien obviamente está mediada por la posibilidad de conocer la herramienta y tener algún antecedente favorable comprobable como la que este trabajo presenta, dependerá principalmente de cuáles son las preguntas pedagógicas que dichos sujetos construyan y cómo interpretan la realidad de sus estudiantes y lo que buscan para ellos.

Referencias

[ACS02] E. Allen, R. Cartwright, and B. Stoler. Drjava: A lightweight pedagogic environment for java. In ACM SIGCSE Bulletin, volume 34, pages 137–141. ACM, 2002.

[BC04] J. Bennedsen and M. E. Caspersen. Teaching object-oriented programming – towards teaching a systematic programming process. In Eighth Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts. Affiliated with 18th European Conference on Object-Oriented Programming (ECOOP 2004), 2004.

[CO16] W. Cazzola and D. M. Olivares. Gradually learning programming supported by a growable programming language. IEEE Transactions on Emerging Topics in Computing, 4(3):404–415, 2016.

[GF03] K. E. Gray and M. Flatt. Professorj: a gradual introduction to java through language levels. In Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, pages 170–177. ACM, 2003.

[GLPDP11] Griggio, C. Leiva, G. Polito, G. Decuzzi, G. Passerini, N. “A programming environment supporting an prototype-based introduction to OOP”. European Smalltalk User Group, 2011.

[Jen02] T. Jenkins. On the difficulty of learning to program. In Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, volume 4, pages 53–58. Citeseer, 2002.

[LP07] Lombardi, Carlos. Passerini, Nicolás. Cesario, Leonardo. "Instancias y clases en la introducción a la programación orientada a objetos". Smalltalks 2007 – Primera Conferencia Argentina de Smalltalk, 2007.

[KQPR03] M. Kölling, B. Quig, A. Patterson, and J. Rosenberg. The bluej system and its pedagogy. Computer Science Education, 13(4):249–268, 2003.

[PFTD17] Passerini, Nicolás. Lombardi, Carlos. Fernandes, Javier. Tesone, Pablo. Dodino, Fernando. "Wollok: Language+ IDE for a gentle and industry-aware introduction to OOP." 2017 Twelfth Latin American Conference on Learning Technologies (LACLO). IEEE, 2017.

[PFTF15] Passerini, Nicolás. Fernandes, Javier. Tesone, Pablo. Fortini, Débora. “Wollok - Relearning How To Teach Object-Oriented Programming”. CONAIISI 2015.

[SGURGH15] J. E. Sánchez-García, M. Urías-Ruiz, and B. E. Gutiérrez-Herrera. Análisis de los problemas de aprendizaje de la programación orientada a objetos. Ra Ximhai, 11(4):148–175, 2015.

[SP13] Spigariol, Lucas. Passerini, Nicolás. “Enseñando a programar en la orientación a objetos”. CONAIISI, 2013

[Spi15] Spigariol, Lucas (2015) “Estrategias pedagógicas para la enseñanza de la programación”. Tesis de Magister en Docencia Universitaria. UTN FRBA.

[Spi16] SPIGARIOL, LUCAS (2016) IEEE Congreso Argentino de Ciencias de la Informática y Desarrollos de investigación. (CACIDI). UNSAM/Universidad CAECE/Universidad Central de Chile. Argentina. Diciembre 2016. "A pedagogical proposal for teaching object-oriented programming: Implementation through the educational software Wollok," Publicado: Electronic ISBN: 978-1-5090-2938-9

[Uys] M. P. Uysal. The effects of objects-first and objects-late methods on achievements of oop learners.

Sitio oficial de Wollok: https://www.wollok.org/

Repositorio del proyecto: https://github.com/uqbar-project/wollok-language

Repositorios de ejemplos de código: https://github.com/wollok

Fundación Uqbar: http://www.uqbar.org/