Experiencias con Da Vinci Concurrente en la enseñanza inicial de la programación y la programación concurrente

Beatriz O. Depetris, Daniel Aguil Mallea, Horacio Pendenti, Germán Tejero, Guillermo Feierherd
Instituto de Desarrollo Económico e Innovación
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur
Hipólito Irigoyen 880 - Ushuaia - Tierra del Fuego
{bdepetris, daguil, hpendenti, ctejero, gfeierherd}@untdf.edu.ar

Resumen

La enseñanza y el aprendizaje de la programación han presentado y continúan presentando importantes desafíos para los docentes y los alumnos. En los últimos años estos han ido aumentando, como consecuencia de la necesidad de incorporar, además de los conceptos tradicionales de programación, los de programación concurrente.
Este trabajo tiene como objetivo describir el origen y la evolución de la herramienta que estamos utilizando para facilitar ambos procesos, así como las experiencias preliminares que durante este año estamos llevando a cabo en dos cursos de la carrera de Licenciatura en Sistemas.
Se detallan los principales logros y dificultades encontradas y se describen algunos de los trabajos futuros destinados a introducir mejoras en la herramienta.

Palabras clave: enseñanza, aprendizaje, programación, programación concurrente, visualización de algoritmos

Introducción

La experiencia demuestra que la tarea de enseñar y aprender a resolver problemas utilizando una computadora ha sido y es un proceso plagado de dificultades.
Además, en los últimos años parece incrementarse el número de estudiantes que fallan en este proceso, con las consiguientes manifestaciones de preocupación por parte de los docentes.
Todo parece indicar que los métodos tradicionales de enseñanza no se ajustan a las necesidades de los alumnos actuales. Algunas de las características de los alumnos que pueden citarse como causas de esta situación son las dificultades que muestran en la creación de algoritmos para resolver problemas y en el bajo desarrollo de las capacidades vinculadas a la abstracción.
La primera de ellas tiene raíces más profundas, ya que puede asegurarse que la imposibilidad que muestran los alumnos para crear algoritmos tienen su origen en su incapacidad para resolver aún los problemas más elementales.
En cuanto a la segunda, afirma Piaget que las capacidades vinculadas a la abstracción deberían desarrollarse en el ser humano en el cuarto (y último) estadio del desarrollo cognitivo, al que denomina el estadio de las operaciones formales. En el estadio anterior (7 a 12 años), denominado el de las operaciones concretas, se desarrollan modos de pensamiento que constituyen una forma primaria de pensamiento lógico, condicionado a su aplicación a situaciones de experimentación concretas. En el siguiente y último, que debería comenzar con la adolescencia, alrededor de los 12 años, aparece la lógica formal y la capacidad para trascender la realidad, manejando y verificando hipótesis de manera exhaustiva y sistemática.
La teoría de Piaget subyace en la afirmación de Rutherford y Ahlgren, citados por Dann y Cooper en su artículo sobre Alice [1]: “El aprendizaje de los estudiantes es usualmente de lo concreto a lo abstracto. La gente joven puede aprender más rápidamente acerca de

cosas que son tangibles y directamente accesibles a sus sentidos –visual, auditivo, táctil. Con la experiencia, van desarrollando la habilidad para entender conceptos abstractos, manipular símbolos, razonar lógicamente y generalizar."
No obstante, un último párrafo de la misma cita relativiza los límites etarios establecidos por Piaget: "Sin embargo estas habilidades se desarrollan lentamente, y la dependencia de mucha gente de ejemplos concretos de nuevas ideas persiste durante toda la vida."
Nuestra realidad resulta más acorde con la última afirmación de Rutherford y Ahlgren que con el límite de los 12 años establecido por Piaget, ya que muchos de los alumnos que ingresan a la Universidad (al menos entre los que han elegido carreras vinculadas a la disciplina informática), poseen un bajo nivel de desarrollo de estas habilidades.
Es por ello que, desde hace varios años, los cursos introductorios de programación (CS1 y CS2) recurren a distintas herramientas cuya característica principal (y común) es la de permitir visualizar distintos aspectos de los programas (en particular la ejecución de los mismos), lo que constituye, básicamente, una forma de trabajar sobre el pensamiento lógico en base a situaciones de experimentación concreta. Los microcosmos, inspirados en el trabajo de Papert, resultan un caso habitual de particular interés. [2]
Si bien estas herramientas han probado ser una ayuda importante para los alumnos de los primeros años (al menos en las etapas iniciales del aprendizaje), nuestra opinión es que deben ser utilizadas cuidadosamente, ya que es probable que retarden el proceso de adquisición de la capacidad de abstracción, indispensable en los profesionales de la disciplina. [3] [4] [5] [6]
Por último resulta necesario señalar que la capacidad de multiprocesamiento de los procesadores actuales plantea un nuevo desafío a la enseñanza de la disciplina. Ya no es suficiente que el alumno adquiera sólo conocimientos teóricos básicos de programación concurrente y paralela, como sucedía hasta hace apenas unos años. Ahora debe tener un conocimiento mucho más profundo de estos temas y debe ser capaz de diseñar soluciones a los problemas que puedan implementarse en procesos concurrentes y paralelos.
Este artículo está organizado como sigue. En primer lugar se mencionan las principales características del Visual Da Vinci (utilizado desde hace varios años en nuestras carreras) y del Da Vinci Concurrente, una evolución de aquel que se ha comenzado a utilizar este año. Luego se describen las características de las asignaturas en las que se lo utiliza. A continuación se presentan algunos comentarios preliminares sobre las breves y recientes experiencias realizadas con el Da Vinci Concurrente. Finalmente se enumeran algunos de los trabajos futuros.

Visual Da Vinci y Da Vinci Concurrente

El Visual Da Vinci (VDV) es un lenguaje de programación y un entorno integrado de desarrollo (IDE) creado en el III-LIDI de la Universidad Nacional de La Plata (UNLP), con el fin de facilitar los procesos de enseñanza y de aprendizaje inicial de la programación. [7]
Como lenguaje el VDV tiene una sintaxis similar a Pascal, lo que facilita, superadas las instancias iniciales del aprendizaje, comenzar a utilizar este último en la resolución de problemas de mayor complejidad.
Por otra parte, el conjunto de comandos es reducido. Los contenidos en él están pensados para desplazar un robot en una ciudad compuesta de calles y avenidas perpendiculares entre sí. Adicionalmente, el *robot* puede realizar algunas tareas elementales (recoger, depositar) sobre objetos (papeles y flores) ubicados previamente en las intersecciones de la ciudad o transportados por el robot.
La ejecución del algoritmo puede ser visualizada en la pantalla, lo que constituye la principal ventaja del producto en la iniciación al aprendizaje de la programación.

El VDV ha sido ampliamente empleado en la Facultad de Informática de la UNLP y en otras instituciones de educación superior. En nuestras carreras ha sido utilizado desde el año 1999, con resultados satisfactorios.
La experiencia acumulada en el uso, con resultados que son evidentemente positivos, también permitió identificar aspectos del producto que podían ser mejorados.
Como consecuencia de ello se plantearon dos trabajos de tesis de grado (una ya defendida y otra a punto de hacerlo), en los que se definió e implementó totalmente una nueva versión a la que denominamos Da Vinci Concurrente (DVC).

Características del DVC

El nuevo producto incorpora, además de las facilidades para la enseñanza de la programación concurrente que se describirán posteriormente, un conjunto de mejoras al Visual Da Vinci, que permiten utilizarlo con mejores resultados aún en la enseñanza de la programación de procesos secuenciales. Entre otras cabe señalar:

- Implementación multiplataforma.
- Enriquecimiento de los mensajes de error en tiempo de compilación y de ejecución.
- Incorporación del tipo de datos cadena de caracteres.
- Lectura de variables en tiempo de ejecución.
- Incorporación de un conjunto de primitivas (números aleatorios, conversión texto a número y número a texto, etc.)
- Eliminación del sangrado como mecanismo sintáctico para delimitar los bloques de código, reemplazándolo por las palabras clave comenzar y fin.
- Aceptación de una distribución de flores, papeles y obstáculos definida por el alumno y posibilidad de conservar la misma para futuras ejecuciones. Esto agiliza el proceso de depuración del programa mediante la estrategia de prueba y error.

Por otra parte, las extensiones del lenguaje orientadas a permitir su uso para introducir los primeros conceptos de programación concurrente, consisten en:

- Incorporación del tipo de dato abstracto semáforo, tanto general como binario. Los semáforos permiten resolver los problemas habituales de la concurrencia. Sobre este tipo de dato se puede operar exclusivamente a través de las primitivas *iniciarSemaforo*, *esperar* y *avisar*.
- Implementación de distintos planificadores de corto plazo, entre ellos los más comunes (FIFO, round robin, aleatorio). Si bien un programa concurrente debe funcionar de modo correcto, con independencia del planificador de procesos que utilice el sistema operativo, la posibilidad de elegir entre distintos planificadores permite apreciar la influencia de los mismos en la ejecución de procesos concurrentes.
- Incorporación de facilidades que permiten actuar sobre el no determinismo que caracteriza a los procesos concurrentes. El DVC cuenta con mecanismos que permiten:
  - ✓ la reproducción exacta de una ejecución concurrente que ha terminado (en forma correcta o incorrecta)
  - ✓ la posibilidad de manipular la intercalación de las trazas de los procesos secuenciales que intervienen en la ejecución concurrente, generando trazas definidas por el usuario (trazas forzadas), que a través de un análisis deductivo del código fuente se presume que llevan a situaciones de fallo y que, en función del no determinismo, pueden no producirse aún cuando se realicen un número elevado de ejecuciones del programa. En esta primera etapa, la responsabilidad de que

las trazas manuales introducidas respondan a los procesos concurrentes que se están ejecutando, recae exclusivamente en el usuario.

Características de las asignaturas

El VDV se ha utilizado en los últimos años en la asignatura Expresión de Problemas y Algoritmos. En el último año, ya disponible una primera versión operativa del DVC, ha sido reemplazado por este último. Como se describe más adelante, la experimentación con DVC se continúa en la asignatura Introducción a la Concurrencia.

Cabe señalar que las dos materias en las que se DVC se aplica actualmente han surgido durante la última modificación del Plan de Estudios de la carrera, producida durante el año 2009 e implementada gradualmente a partir del año 2010.

Las modificaciones producidas en ese momento tuvieron su origen, entre otras cuestiones, en el convencimiento de que la mejora de la enseñanza de la programación no se resuelve recurriendo exclusivamente a más y mejores herramientas. Entre muchos otros factores que inciden en los resultados, la ubicación de los espacios curriculares dedicados al tema en el Plan de Estudios, resulta de fundamental importancia.

La siguiente es una breve descripción de ambas asignaturas.

Expresión de Problemas y Algoritmos (EPA)

Se trata de una materia que corresponde al primer cuatrimestre del primer año de las carreras de Licenciatura en Sistemas y Analista Programador Universitario. Tiene asignada una carga horaria de 6 horas semanales.

Antes de la modificación del plan de estudios los temas estaban incorporados a la asignatura Algorítmica y Programación, que se dictaba recién en el segundo cuatrimestre de primer año. Experiencias con un curso preuniversitario de Resolución de Problemas, basado en el VDV, realizado durante los años 1999 y 2000, mostraron que los alumnos que tomaban el curso tenían un mejor rendimiento en la asignatura Algorítmica y Programación. [8]

Los resultados alentadores de dichas experiencias llevaron en un primer momento a continuar realizando los cursos preuniversitarios utilizando el VDV (que a partir de 2005 se ofrecieron simultáneamente en forma virtual) [9], y finalmente a la separación de las asignaturas y al dictado de Expresión de Problemas y Algoritmos en el primer cuatrimestre.

Actualmente, para la mayoría de los alumnos, recién ingresados a la carrera, es la primera actividad de resolución de problemas utilizando un lenguaje de Expresión de Problemas, y en particular con un lenguaje ejecutable en una computadora.

Se trata, por otra parte, de una habilidad que los alumnos deben adquirir necesariamente, ya que deberán ponerla en práctica (y profundizarla), tanto durante el desarrollo de sus estudios como en el futuro ejercicio profesional. Dada su importancia el objetivo de la asignatura es que los alumnos adquieran conocimientos que les permitan desarrollar y potenciar destrezas para realizar la actividad descripta.

El ejercicio de la actividad requiere contar con la capacidad de expresión suficiente como para indicarle a la máquina lo que debe llevar a cabo.

Los contenidos mínimos de la asignatura son:

- Análisis y resolución de problemas.
- Especificación simbólica.
- Expresión de soluciones en un lenguaje algorítmico.

Habitualmente, en las dos primeras semanas de clase se resuelven problemas utilizando un seudocódigo elemental y el lápiz y el papel. A partir de la tercera semana, y hasta la finalización de la asignatura, se utiliza el DVC.

En el segundo cuatrimestre, en la asignatura Algorítmica y Programación I, los alumnos

reemplazan este lenguaje por el lenguaje Pascal.

Introducción a la Concurrencia

La materia se dicta en el primer cuatrimestre de tercer año de ambas carreras. Tiene asignada una carga horaria de 4 horas semanales.
En sus orígenes los temas estaban incorporados a la asignatura Sistemas Operativos, pero por varios motivos en la modificación del plan de estudios producida en el año 2009, se separaron para integrar un nuevo espacio.
Entre las causas que justificaron la división cabe mencionar la importancia que tiene hoy la programación concurrente y paralela y la necesidad de introducir los conceptos básicos de programación concurrente en un momento temprano que permita su posterior maduración.
Si bien la materia se dicta desde el año 2011, este es el primer año en el que se aplicó el DVC en la etapa inicial.
Los contenidos mínimos de la asignatura son:

- Conceptos de concurrencia.
- Especificación de la ejecución concurrente.
- Comunicación y sincronización.
- Concurrencia con variables compartidas.
- Concurrencia con pasaje de mensajes.
- Sistemas multiprocesador para concurrencia real.
- Lenguajes de programación concurrente.
- Diseño y programación de algoritmos concurrentes.

Posteriormente se continúa utilizando Pascal Concurrente para la resolución de los problemas propuestos.

Comentarios preliminares sobre las experiencias realizadas

En esta sección se comentan los resultados obtenidos en las experiencias llevadas a cabo en ambas asignaturas.

Expresión de Problemas y Algoritmos

Este es el primer año en el que se utiliza DVC en reemplazo del VDV.
Si bien se ha tratado de una experiencia muy corta (el cuatrimestre aún no ha finalizado al momento de redactar este artículo), podemos señalar los siguientes aspectos positivos:

- La mejora introducida en los mensajes de error, fundamentalmente en los que se generan por errores en tiempo de compilación, se nota en la mayor autonomía que tienen los alumnos para resolverlos. En la versión anterior esto no ocurría y el alumno era muy dependiente del docente.
- Gracias a que rápidamente se corrigen los errores de sintaxis, lo que permite tener un programa ejecutable en un tiempo relativamente corto, los alumnos disponen de más tiempo para concentrarse en resolver los errores que se producen en tiempo de ejecución. Cabe señalar que esta actividad es la que contribuye a producir un *aprendizaje profundo* [10], pues el alumno aprende de sus errores y adquiere y potencia capacidades que lo ayudan a resolver problemas utilizando la computadora.
- La posibilidad incorporada al DVC de informar constantes alfanuméricas permite que los alumnos las incluyan para personalizar el seguimiento en la ejecución del código, así como acompañar con mensajes aclaratorios los datos que se van obteniendo.
- El agregado del tipo de dato *cadena*, y la posibilidad de leer variables en tiempo de ejecución permiten ampliar la gama de ejercicios que se plantean en las clases prácticas. Como consecuencia de ello se crea un ambiente de trabajo semejante al que podría obtenerse con lenguajes más completos, lo que seguramente

contribuirá a simplificar la transición a ellos cuando el alumno deba utilizarlos.

En cuanto a los aspectos que podrían perfeccionarse cabe mencionar la necesidad de mejorar la forma de distribuir objetos (papeles y flores) en el entorno, cuando estos son colocados manualmente antes de comenzar a ejecutar el programa. Si bien la posibilidad de guardar las distribuciones para utilizarlas en posteriores ejecuciones es una ventaja (especialmente cuando se está depurando el programa y se ha producido una falla en su ejecución), los objetos deben ser colocados uno a uno en cada esquina. Si en función del problema deben colocarse *n* objetos en una esquina, la operación debe repetirse *n* veces.

Introducción a la Concurrencia

El DVC se ha utilizado por primera vez en esta materia como lenguaje para introducir al alumno en los problemas que plantea la concurrencia, sus principios de solución y la resolución de problemas mediante algoritmos concurrentes.
Tradicionalmente la enseñanza de estos temas ha planteado un conjunto de dificultades debido a lo abstracto de los conceptos y el no determinismo del planificador, que impide utilizar algunas estrategias clásicas de la programación de procesos secuenciales únicos. A modo de ejemplo, ya no es posible aplicar la estrategia de prueba y error para probar y depurar los programas (consecuencia directa del no determinismo), y el análisis del código fuente mediante un enfoque deductivo, si bien aún permite identificar posibles defectos en el código, no resulta una tarea sencilla para nadie y menos aún para los principiantes.
Las expectativas sobre los resultados positivos de esta experiencia están basadas especialmente en la capacidad del producto de visualizar la ejecución de los algoritmos concurrentes especificados mediante el lenguaje.
Se utilizó DVC en la primer parte de la práctica de la materia, dedicada a introducir los primeros problemas de concurrencia mediante la definición de sección crítica y la implementación de soluciones a la exclusión mutua, tanto por software (algoritmo de Peterson), como mediante semáforos.
Una de las cuestiones que requirió mayor atención en la planificación previa de la materia fue como convertir los ejemplos típicos con los que se presenta el tema en los libros de texto, a la metáfora de la ciudad utilizada por el DVC.
Es evidente que cada proceso concurrente debe corresponderse con un robot, pero no resulta tan inmediato determinar cuáles serán los recursos compartidos por los cuáles los procesos compiten o a través de los cuáles colaboran para la solución del problema.
En esta primera etapa se han utilizado para ello tanto las intersecciones (denominadas esquinas en el DVC), por las cuáles sólo un robot de los dos podía pasar a la vez, como los objetos (flores o papeles) que puede manipular el robot. Estos objetos han sido empleados en una implementación del problema del productor y el consumidor con un buffer acotado, en la que un robot productor y otro consumidor ponen y sacan, respectivamente, papeles o flores en una esquina de la ciudad en la que pueden depositarse simultáneamente un máximo de *n* (tamaño del buffer) objetos.
La Figura 1 muestra el resultado final de haber ejecutado la solución al problema del paso de un único robot por vez por una intersección, trabajando con dos robots y utilizando semáforos. Se la incluye al único efecto de mostrar la interfase de la herramienta, con sus cuatro paneles: el de visualización y edición del código fuente, el de la ciudad y los objetos (en el que podrá visualizarse la ejecución de los algoritmos que impliquen al menos un robot), el de mensajes (tanto los de error como los generados explícitamente por el programador) y, finalmente, el que indica el contexto de ejecución (variables, orientación del robot, planificador, etc.)
Por su parte, las Figuras 2 y 3 muestran en detalle los paneles de visualización de la ciudad y del código, respectivamente. Por último, la Tabla 1 muestra el código completo correspondiente a la solución a este problema.

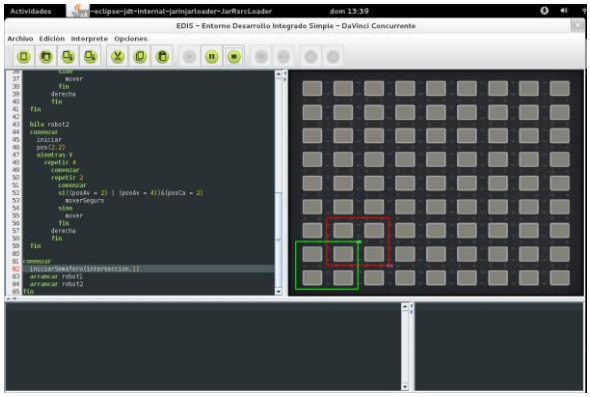

Figura 1

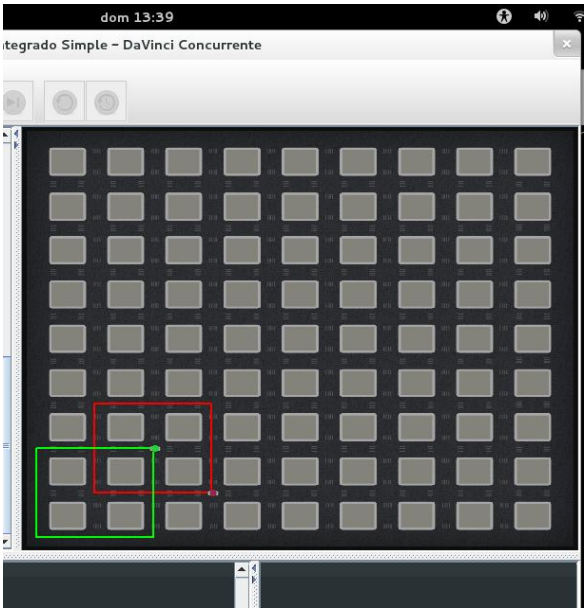

Figura 2

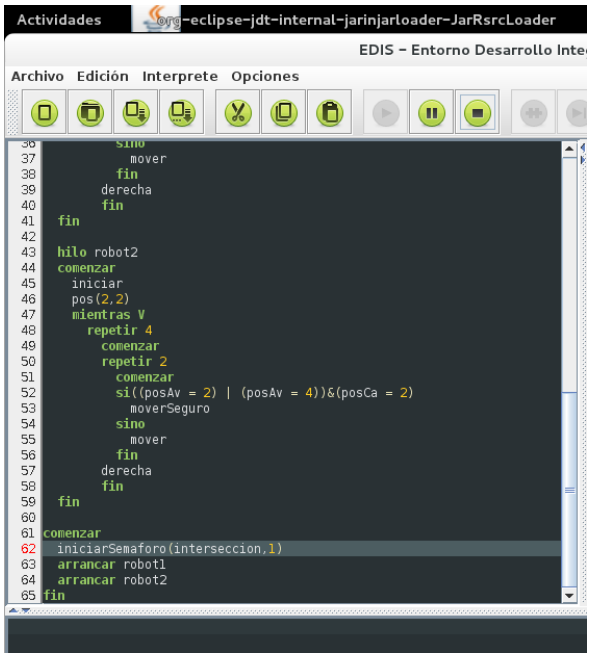

Figura 3

```
programa InterseccionSemaforo
variables
  interseccion:semaforoBinario

subprogramas
  procedimiento moverSeguro
  variables
    avance :logico
  comenzar
    avance := F
      mientras !avance
        comenzar
          esperar(interseccion)
            si(!hayRobot)
            comenzar
              mover
              avance := V
            fin
            avisar(interseccion)
          fin
 fin

hilos
 hilo robot1
  comenzar
    iniciar
      pos(1,1)
      mientras V
            repetir 4
              comenzar
                repetir 2
                comenzar
                  si((posAv = 1) | (posAv = 3))&(posCa = 3)
                    moverSeguro
                  sino
                    mover
                  fin
                  derecha
                fin
            fin
 hilo robot2
  comenzar
    iniciar
      pos(2,2)
            mientras V
              repetir 4
                comenzar
                  repetir 2
                  comenzar
                    si((posAv = 2) | (posAv = 4))&(posCa = 2)
                      moverSeguro
                    sino
                          mover
                    fin
                    derecha
                  fin
                fin

comenzar
  iniciarSemaforo(interseccion,1)
    arrancar robot1
    arrancar robot2
fin
```

Tabla 1

Si bien se ha tratado de una experiencia muy breve, se pueden obtener algunas conclusiones iniciales.
En lo que hace a las bondades del producto se pueden mencionar:

- La herramienta corre sobre cualquier plataforma, lo que es una ventaja importante, pues los alumnos han podido instalarlo sin inconvenientes en computadoras de su propiedad, en las que utilizan distintos sistemas operativos (Windows, Linux y MacOS).
- La mayoría de los alumnos ya había utilizado VDV al momento de cursar Expresión de Problemas y Algoritmos en su primer año. Si bien DVC presenta algunas diferencias, la sintaxis del lenguaje y el repertorio de instrucciones se han mantenido simples. También se ha mantenido la metáfora de la ciudad y el robot (en este caso más de uno), por lo que la adaptación no presentó problemas. Esta es una característica importante en términos de tiempo de aprendizaje y familiarización del alumno con la herramienta, más aún teniendo en cuenta la reducida carga horaria disponible para la asignatura.
- Permite encarar los problemas de la concurrencia desde una óptica asociada a situaciones concretas que se expresan naturalmente en el lenguaje. Esto disminuye el nivel de abstracción requerido para su interpretación.
- Implementa la posibilidad de ejecución paso a paso, lo que permite un seguimiento minucioso de la ejecución de un programa para depurarlo y visualizar la ocurrencia de eventos. Esto es esencial para la comprensión del problema en estas primeras instancias de desapego de la programación en base a un único proceso secuencial.
- Se puede utilizar desde una interfaz visual (que permite ver los robots desplazándose por la ciudad) o desde una consola (que muestra la traza de ejecución del proceso concurrente). Aunque no se ha utilizado este último modo, la posibilidad está prevista para aquellos casos en que se requiera.
- Si bien por razones de tiempo no se han utilizado aún las funcionalidades de repetir la última ejecución ni la de manipular las trazas, puede afirmarse que son estas las facilidades de la herramienta cuya aplicación resulta fundamental a la hora de probar y depurar los programas concurrentes.

Por otra parte, se han detectado los siguientes aspectos que justificarían continuar trabajando sobre el lenguaje para brindar más soluciones y posibilidades:

- El lenguaje no implementa la definición de constantes, que serían útiles al momento de desarrollar buenas prácticas de programación.
- El lenguaje no implementa estructuras de datos, como por ejemplo arreglos, que permitirían proponer nuevos problemas, tanto en la programación imperativa como concurrente.
- Sería de gran utilidad la incorporación de otros mecanismos para resolver los problemas de sincronización, como por ejemplo monitores y mensajes.
- Tan importante como lo anterior sería el agregado de primitivas atómicas, del tipo TS (Test and Set) o CSW (Compare and Swap), para utilizarlas en la implementación de soluciones spin-lock más elaboradas.
- Otros mecanismos adicionales que podrían incluirse son los procedimientos atómicos (coarse grained) a modo de regiones críticas condicionales.
- El producto no cuenta con mecanismos que permitan definir un tipo de robot (por ejemplo productor, o consumidor) y luego crear varias instancias del mismo. Cada robot es un hilo, y debe ser definido explícitamente. Para un número pequeño (dos o tres robots del

mismo tipo), la limitación se supera sin inconvenientes, pero constituye una limitación cuando se pretende incrementar el número.

- Sería interesante poder incorporar alguna representación de las herramientas de sincronización (por ejemplo los semáforos) a la representación visual de la ciudad. Por ejemplo, poner un semáforo en un esquina de la ciudad y visualizar como, en función de su estado, permite o no el paso de robots por esa esquina.

Finalmente falta trabajar en la creación de metáforas adaptadas al contexto (ciudad y robots), para los problemas clásicos de sincronización y exclusión mutua.
No obstante, aún con el escaso tiempo de experimentación con el producto, puede afirmarse que la experiencia ha sido satisfactoria, destacándose que en cuestión de minutos los alumnos pudieron instalar el producto, comenzar a utilizarlo y arribar a las primeras soluciones. Esto suele ser una tarea mucho más tediosa y extensa con otros lenguajes o herramientas afines.

Conclusiones y trabajos futuros

Puede afirmarse que las experiencias en ambas asignaturas han sido, en principio, muy satisfactorias.
La herramienta ha facilitado tanto el proceso de aprendizaje por parte de los alumnos, como el de enseñanza por parte de los docentes.
También, y no de menor importancia, son las sugerencias de mejoras que se han producido como consecuencia de su primera aplicación. Estas están mencionadas en la experiencia de cada asignatura.
El hecho de que la mayor parte de ellas se presenten en Introducción a la Concurrencia es consecuencia de que es esta la primera vez que se utiliza la herramienta para introducir los conceptos básicos de programación concurrente. La experiencia en Expresión de Problemas y Algoritmos tiene como antecedentes las realizadas en los últimos años con VDV, como consecuencia de las cuales se obtuvieron las sugerencias de mejora que se incorporaron al DVC.
Los trabajos futuros contemplan la introducción de las mejoras indicadas. Estas, más las necesarias para utilizar el lenguaje utilizando robots físicos, forman parte de uno de los proyectos de investigación en los que trabajan los autores.
Por otra parte se trabaja también en el diseño de instrumentos que permitan evaluar en forma más sistemática las experiencias previstas para el próximo año.

Referencias

1. Dann, W., Cooper, S.: Education: Alice 3: concrete to abstract. Communications of ACM 52, 8, 27–29, (August 2009)
2. Papert, S: Desafío a la mente: computadoras y educación. Ediciones Galápago. (2da. edición, Buenos Aires, 1982)
3. Brooks. F. P.: No Silver Bullet - Essence and Accident in Software Engineering. Proc. IFIP Tenth World Computing Conference, 1069 - 1076, (1986)
4. Devlin, K.: Why universities require computer science students to take math, Communications of ACM 46, 9, 37–39, (September 2003)
5. Kramer, J.: Is abstraction the key to computing?, Communications of ACM 50, 4, 36–42, (April 2007)
6. Utting, I., Cooper, S., Kölling, M., Maloney, J., Resnick, M.: Alice, Greenfoot, and Scratch -- A Discussion, Transactions on Computer Education 10, 4, Article 17, 11 pages, (November 2010)
7. Champredonde, R., De Giusti, A.: Design and Implementation of Visual Da Vinci. In: III Congreso Argentino en Ciencias de la Computación (CACiC 1997), La Plata (1997)
8. Feierherd G., Depetris B., Jerez M.: Una evaluación sobre la incorporación temprana de algorítmica y programación en el ingreso a informática. Proc. CACiC, VII Congreso Argentino de Ciencias de la Computación, Calafate (2001)
9. Feierherd G., Depetris B.: Ushuaia: la necesidad de las TICs en la Educación Universitaria. Proc. GCETE'05, Global Congress on Engineering and Technology

Education (GCETE'05). Santos (Brasil), (2005)
10. Valenzuela J.: Habilidades de pensamiento y aprendizaje profundo, Revista Iberoamericana de Educación, 46, 7, (Julio 2008)