Concurrencia y Paralelismo en CS1: la utilización de un Lenguaje Visual orientado.

Armando De Giusti[1,2], Fernando Emmanuel Frati[1,2], Fabiana Leibovich[1], Mariano Sanchez[1], Laura De Giusti[1], María C. Madoz[1]

[1] Instituto de Investigación en Informática LIDI (III-LIDI) – Facultad de Informática –UNLP, La Plata, Argentina
[2] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) , La Plata, Argentina

Resumen

Se presenta una discusión sobre la introducción de conceptos de concurrencia y paralelismo en un primer curso de Algorítmica (CS1) de las carreras de Informática.
En este marco se presenta el entorno visual LMRE (Lidi Multi Robot Environment) explicando la definición pedagógica del mismo, su implementación y las líneas de I/D asociadas.
En el trabajo se analiza la evolución del lenguaje y entorno monoprocesador Visual Da Vinci que se ha utilizado en la introducción a la programación en varias Universidades y las ventajas de este nuevo enfoque.
Se analizan y explican las características del lenguaje multiprocesador propuesto y la redefinición del entorno de tareas, así como se ejemplifica su uso en algoritmos concurrentes y paralelos.

Palabras clave: Algoritmos, Concurrencia, Paralelismo, Entorno, Multirobot, Algoritmos Concurrentes y Paralelos.

1. Introducción

Los cambios tecnológicos de los últimos años han derivado en que las máquinas paralelas que antes estaban restringidas a aplicaciones muy específicas (servidores web, HPC, simulaciones científicas, etc.) se han universalizado [6]. PCs, notebooks e incluso los celulares modernos incluyen procesadores de múltiple núcleos para conseguir mayor poder de cómputo [11] [12].

De hecho, en las últimas recomendaciones de la ACM/IEEE para carreras de Informática se aconseja incluir temas de concurrencia y paralelismo desde los primeros años en las carreras [1]. A modo de ejemplo:

- En [7] los autores reconocen que existe la necesidad de desarrollar herramientas que faciliten tratar con problemas de cómputo de altas prestaciones. Su propuesta radica en un software que facilita el desarrollo y ejecución de programas paralelos en clusters HPC, sobre hardware alquilado a Amazon y utilizan imágenes de máquinas virtuales para cada curso. Aunque el framework que desarrollaron es escalable y rentable, está enfocado a facilitar la gestión de los recursos necesarios para la enseñanza de la programación paralela, y no en facilitar al alumno la adquisición de sus principios.
- Otros autores presentan una estrategia de clases basada en herramientas de visualización destinadas a la conceptualización de los programas paralelos. En [8] la herramienta propuesta consiste en una aplicación que permite la introducción de código y la visualización del mismo a través de diagramas UML. Así se identifican visualmente posibles secciones de código concurrente. Esta propuesta constituye una ayuda en el análisis del código paralelo, pero aún el alumno debe conocer el lenguaje con las

primitivas específicas que soportan la concurrencia y su conceptualización.

- Recientemente en [9] los autores presentan un estudio de los últimos tres años de un curso CS1 en el que han introducido conceptos de concurrencia a través de casos prácticos y sencillos que deben ser implementados en JAVA. En el artículo se remarca la importancia de trasmitir a los alumnos los conceptos generales de concurrencia para despertar su interés en esos temas, como también la imposibilidad de abarcar temas avanzados como métricas de eficiencia, en una etapa temprana de la carrera.

Este tema y una discusión sobre las posibilidades de incorporar estos conceptos en un curso CS1 ha sido tratado por los autores en trabajos anteriores [2] [10].

1.1. El caso de la UNLP

El curso que se dicta actualmente en la Facultad de Informática de la UNLP abarca dos semestres y tiene el título genérico de "Algoritmos, Datos y Programas". Responde a un modelo clásico centrado en el aprendizaje de la expresión de algoritmos, la introducción a las estructuras de datos lineales y no lineales y la incorporación de conceptos relacionados con modularización, análisis de eficiencia, abstracción y reuso. En el 2do. semestre se introduce conceptualmente el paradigma de objetos.

Las tareas experimentales se realizan inicialmente con un entorno propio Visual Da Vinci [3] y posteriormente con Pascal.

A partir del año 2010 se ha propuesto un cambio gradual, que tiene 3 etapas:

- Incorporación de la descripción de las arquitecturas de los nuevos procesadores, como evolución de la máquina de Von Neumann e Introducción a los conceptos básicos de Concurrencia.
- Rediseño del programa de la asignatura, planteando la programación concurrente y paralela como el caso general y la programación secuencial como la excepción.
- Desarrollo de un entorno visual de expresión de Algoritmos concurrentes y paralelos que resulte en una evolución del Visual Da Vinci (VDV) y admita mecanismos de comunicación y sincronización, tanto por memoria compartida como por mensajes

La especificación del nuevo entorno visual llamado LMRE y una discusión sobre cómo incorporar las nociones de concurrencia y paralelismo fueron tratadas en [4] y [10] y en este trabajo se completa el análisis y se presenta un caso de estudio sobre el entorno.

El artículo se organiza de la siguiente manera: en la sección 2 se presenta un problema típico con su solución secuencial en código Da Vinci. La sección 3 presenta las modificaciones que se hacen sobre el lenguaje y el entorno para considerar múltiples procesadores (robots) trabajando concurrentemente. La sección 4 detalla una especificación y código ejecutable paralelo sobre LMRE. La sección 5 presenta algunos comentarios sobre la implementación del nuevo entorno. Finalmente, la sección 6 expone las conclusiones y líneas futuras de investigación.

2. Modelo Secuencial en Da Vinci

El objetivo en este entorno es resolver problemas donde se especifica el comportamiento de un *único robot*, el cual puede moverse en una ciudad compuesta por 100 avenidas (verticales) y 100 calles (horizontales) y es capaz de distinguir objetos (flores y papeles) y realizar operaciones con los mismos (juntarlos y/o depositarlos). Asimismo el robot puede "contar" e "informar" resultados. La Fig. 1 muestra el esquema tradicional de un programa Da Vinci.

El encabezamiento está compuesto por la palabra clave **programa** seguida de un identificador que determina el nombre del

programa.

```
programa unNombre
procesos
{Declaración de todos los procesos que se utilizarán}
variables
{Declaración de todas las variables a utilizar SOLO en el programa principal}
comenzar
{Sentencias}
Fin
```

Figura 1: Esquema de un programa Da Vinci

En la segunda parte, se declaran los subprogramas y las variables que utilizará el programa principal. El orden de estas declaraciones es importante. No se permiten declaraciones anteriores a subprogramas, de forma tal que no existe la posibilidad de hacer referencia a variables globales desde un subprograma [5].

La declaración de subprogramas comienza con la palabra clave **procesos**, y termina donde se encuentra la palabra clave **variables** o bien la palabra clave **comenzar.**

Por otro lado, la declaración de las variables comienza con la palabra clave **variables** y termina donde se encuentra la palabra clave **comenzar**.

El cuerpo del programa principal es una secuencia de sentencias, delimitada por las palabras clave **comenzar** y **fin**. El cuerpo del programa principal debe comenzar con una sentencia especial **Iniciar**, que sitúa el robot en la esquina (1, 1) de la ciudad. Luego, el comportamiento del robot se especifica con sentencias especiales. La Fig. 2 resume el conjunto de primitivas que puede ejecutar el robot.

Iniciar	PosAv
mover	PosCa
derecha	HayFlorEnLaEsquina
depositarFlor	HayPapelEnLaEsquina
depositarPapel	HayFlorEnLaBolsa
tomarPapel	HayPapelEnLaBolsa
tomarFlor	Informar(…)
Pos(av, ca)	

Figura 2: Primitivas de un programa Da Vinci

A modo de ejemplo, la Fig.3 muestra una solución a un problema clásico que se da en la materia:

"Realizar un programa que recoja todas las flores de la ciudad e informe el número total de flores recogidas."

```
programa cuentaFlores
procesos
  proceso levantarFlores(ES CantFlores: numero)
  comenzar
    mientras HayFlorEnLaEsquina
      tomarFlor
      CantFlores := CantFlores + 1
  fin
  proceso recorrerAvenida(ES CantFlores: numero)
  comenzar
    repetir 99
      levantarFlores(CantFlores)
      mover
    levantarFlores(CantFlores)
  fin
variables
  Flores: numero
comenzar
  Iniciar
  Flores := 0
  repetir 99
    recorrerAvenida(Flores)
    Pos(PosAv + 1, 1)
  recorrerAvenida(Flores)
  Informar(Flores)
Fin
```

Figura 3: solución Da Vinci al problema de juntar flores en la ciudad.

3. Conceptos del entorno LMRE

3.1. Modelo general

- ✓ Existen múltiples procesadores (robots) que realizan tareas y que pueden cooperar y/o competir.
- ✓ El modelo de ambiente ("ciudad") en la que desarrollan sus tareas admite áreas privadas, parcialmente compartidas y totalmente compartidas.
- ✓ Si instanciamos 1 sólo robot en un área que abarque toda la ciudad, repetimos el esquema del Da Vinci.
- ✓ Cuando dos o más robots están en un área compartida, compiten por el acceso a las esquinas del recorrido y a los recursos que allí existan. Para esto deben sincronizar.
- ✓ Cuando dos o más robots (en un área común o no) desean intercambiar información (datos o control) deben hacerlo por mensajes explícitos.
- ✓ La sincronización se da por un mecanismo equivalente a un semáforo binario.
- ✓ La exclusión mutua puede generarse con la declaración de las áreas alcanzadas por cada robot. Acceder a otras áreas de la ciudad, así como salir de ellas no está permitido.
- ✓ Todo el modelo de ejecución es sincrónico y permite la existencia de un reloj virtual de ciclos, que a su vez permite asignar tiempos específicos a las operaciones.
- ✓ El modelo debe ser escalable para asignaturas superiores en que se trate la concurrencia y el paralelismo.

3.2. Estructura de un programa en el entorno LMRE

Al igual que en la especificación Da Vinci, la estructura del programa tiene tres partes: encabezamiento, declaraciones y cuerpo.

La segunda parte se modifica para incluir dos nuevas secciones: la declaración de áreas y robots. El orden de las declaraciones es:

- Declaración de subprogramas
- Declaración de áreas
- Declaración de robots
- Declaración de variables del programa principal

Esta estructura se encuentra en la Fig. 4.

Los subprogramas cumplen la misma función y tienen la misma estructura que en el lenguaje Da Vinci. El cuerpo del programa principal ahora está restringido a la inicialización de robots y asignación de áreas a los mismos.

```
programa unNombre
procesos
  {Declaración de todos los procesos que se utilizarán}
areas
  {Declaración de áreas}
robots
  {Declaración de robots}
variables
  {Declaración de todas las variables a utilizar SOLO en el programa principal}
comenzar
  {SOLO inicialización de robots y asignación de áreas de la ciudad a estos}
fin
```

Figura 4: Esquema de un programa LMRE. Las nuevas secciones se resaltaron en rojo.

3.3. Declaración de áreas

La declaración de áreas comienza con la palabra clave **areas** y termina donde se encuentra la palabra clave **robots**. Un área de la ciudad es un subconjunto rectangular de esquinas de la ciudad por la que los robots pueden circular. Estas pueden ser de tres tipos:

- Área compartida (`areaC`): es el tipo de área por defecto y corresponde a una región de la ciudad de libre acceso, es decir,

cualquier robot puede circular por ella.

- Área privada (areaP): una región de este tipo sólo permite que haya un robot en ella. El intento de un robot de ingresar en un área privada de otro, genera un error en tiempo de ejecución. Notar que las áreas privadas permiten un mecanismo de exclusión mutua implícita entre robots.
- Área parcialmente compartida (areaPC): este tipo de regiones permiten el acceso de uno o varios robots, con la restricción de que deben haber sido autorizados previamente. Notar que las áreas parcialmente compartidas permiten un mecanismo de exclusión mutua selectiva entre robots.

Cada declaración de área comienza con un nombre de área, seguido de dos puntos y la palabra clave areaC, areaP o areaPC (para indicar su tipo) seguido de cuatro parámetros. Éstos representan las coordenadas inferior izquierda y superior derecha que ocupará el área dentro de la ciudad. Cada tipo de área tiene asociado un color, tal como muestra en la Fig. 5.

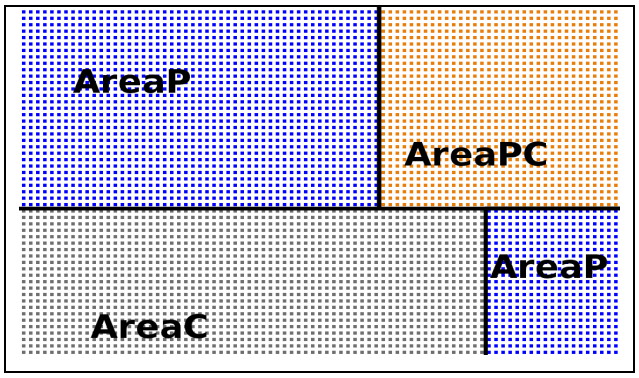

Figura 5: Ejemplo de configuración de áreas en la ciudad

El alcance y la visibilidad de las áreas corresponden a todo el programa, y deben ser asignadas a los robots sobre los que se quiera permitir su acceso.

3.4. Declaración de robots

La declaración de robots comienza con la palabra clave **robots** y termina donde se encuentra la palabra clave **comenzar**.

Los robots tienen una estructura casi idéntica a la del programa principal o subprocesos. Cada declaración de robot tiene las tres partes de un programa: encabezado, declaraciones y cuerpo.

El encabezado comienza con la palabra clave **robot** seguido de un nombre. Las declaraciones de procesos locales y variables locales siguen las mismas reglas que las declaraciones del programa principal, con la salvedad que no pueden declararse nuevas áreas o robots.

De esta manera, la creación de robots siempre será explicita. Una característica a destacar es que aun cuando se quiera utilizar el entorno para simular un ambiente como el del Da Vinci original, será necesario crear el único robot desde el programa principal.

El cuerpo de un robot es una secuencia de sentencias, delimitada por las palabras clave **comenzar** y **fin.** Estas sentencias se corresponden con los tipos de sentencias de Da Vinci, y sirven para definir el comportamiento del robot en la ciudad.

3.5. Cuerpo del programa principal

Desde el programa principal es necesario iniciar cada uno de los robots mediante la directiva Iniciar, que requiere del nombre del robot y su ubicación inicial.

Esta sentencia requiere de comprobaciones en tiempos de compilación para que dos o más robots no intenten ocupar la misma esquina, teniendo en cuenta también a qué tipo de área pertenece la esquina involucrada. Para dar un ejemplo, un robot no puede ocupar una esquina que pertenece a un área exclusiva de otro robot.

A estas sentencias se agrega un nuevo subconjunto, denominado sentencias de concurrencia.

3.6. Manejo de colisiones

Conceptualmente los “recursos compartidos” en este modelo de entorno se pueden reducir

al acceso a una esquina, donde puede haber objetos.

Evitar las "colisiones" en las esquinas es el problema básico de sincronización en LMRE.

Para esto el lenguaje cuenta con directivas que permiten bloquear y liberar el recurso:

- *bloquearEsquina* (BE): indica que el robot pide exclusión para la ocupación de una esquina (lo que a su vez le permite recoger o depositar objetos)
- *liberarEsquina* (LE): indica que el robot deja el recurso libre (la esquina ocupada).

3.7. Comunicación / Sincronización

Tal como se trató anteriormente, tenemos múltiples robots que trabajan en la ciudad. En muchos casos, deberán colaborar en la resolución de algún problema.

Esto requiere comunicación y sincronización. Se adopta un mecanismo explícito de pasaje de mensajes sincrónicos con dos directivas:

- *enviarMensaje* (EM): permite que un robot envíe un mensaje a otro (identificados por su nombre). Al enviar el mensaje, según el modelo sincrónico, el robot se queda en espera de recepción (sincronización) por el robot destino.

  El envío de mensajes a un destino especial, convierte los mismos en broadcast.

- *recibirMensaje* (RM): indica que un robot se quedará esperando hasta sincronizar con el envío de mensaje de otro.

  En la recepción se indica el nombre del robot del cual se espera el mensaje, o un nombre especial que indica que puede ser de cualquiera.

Para finalizar, la Fig. 6 resume las primitivas tratadas en las secciones anteriores.

4. Solución LMRE

La Fig. 7 muestra una solución al problema de recoger todas la flores de la ciudad, utilizando dos robots en el entorno LMRE.

```
AsignarArea(nombreRobot, nombreArea)
Iniciar(nombreRobot, av, ca)
bloquearEsquina(av, ca)
liberarEsquina(av, ca)
enviarMensaje(robotDestino, msj)
recibirMensaje(robotOrigen, msj)
```

Figura 6: LMRE: Primitivas de concurrencia.

La ciudad se divide conceptualmente en dos áreas de igual tamaño (area1 y area2). Luego en la sección de variables del programa principal, se declaran dos robots indicando el tipo al que corresponden.

A través de la sentencia asignarArea se indica qué robot puede circular por cuál área, para luego iniciarlos en la esquina inferior izquierda del área que les corresponde.

Nótese que los procesos levantarFlores y recorrerAvenida permanecen sin cambios significativos: solamente ha cambiado el número de veces que los robots deben repetir la acción de levantar flores y moverse por la avenida (en la solución Da Vinci este valor era de 99).

El código a ejecutar por cada robot es muy similar, con la salvedad de que el robot de tipo uno al finalizar debe enviar un mensaje al robot de tipo dos, para que éste calcule e informe el resultado final.

5. Implementación

El entorno LMRE está siendo desarrollado en el lenguaje de programación JAVA siguiendo las especificaciones expuestas en este trabajo.

Internamente, el entorno está compuesto por un intérprete de dos pasadas. En la primera, se realiza una verificación de la sintáxis y la semántica del programa de entrada, y una vez que esta fase se ejecuta exitosamente, el programa será ejecutado directamente, sin generar ningún código compilado ni intermedio.

```
programa cuentaFlores
procesos
  proceso levantarFlores(ES CantFlores: numero)
  comenzar
    mientras HayFlorEnLaEsquina
      tomarFlor
      CantFlores := CantFlores + 1
  fin
  proceso recorrerAvenida(ES CantFlores: numero)
  comenzar
    repetir 49
      levantarFlores(CantFlores)
      mover
    levantarFlores(CantFlores)
  fin
areas
  area1: areaP(1, 1, 100, 50)
  area2: areaP(1, 51, 100, 100)
robots
  robot tipo1
  variables
    Flores: numero
  comenzar
    Flores := 0
    repetir 99
      recorrerAvenida(Flores)
      Pos(PosAv + 1, 1)
    recorrerAvenida(Flores)
    enviarMensaje(verde, Flores)
  fin
  robot tipo2
  variables
    Flores: numero
    Flores1: numero
  comenzar
    Flores := 0
    Flores1 := 0
    repetir 99
      recorrerAvenida(Flores)
      Pos(PosAv + 1, 51)
    recorrerAvenida(Flores)
    recibirMensaje(rojo, Flores1)
    Informar(Flores + Flores1)
  fin
variables
robot1: tipo1
robot2: tipo2
comenzar
  AsignarArea(rojo, area1)
  AsignarArea(verde, area2)
  iniciar(rojo, 1, 1)
  iniciar(verde, 1, 51)
fin
```

Figura 7: Solución LMRE al problema de juntar flores.

La Fig. 8 muestra un prototipo del entorno, en el que puede verse el resultado de ejecutar el código de la Fig.7.

En este entorno puede verse la pantalla dividida en dos paneles principales:

1. A la izquierda se encuentra la zona de información, donde se aprecia una ventana de previsualización de la ciudad entera, los robots que forman parte de la ejecución como así también información sobre estos.
2. A la derecha, la región de ciudad seleccionada por el recuadro negro en la ventana de previsualización. En la figura puede observarse los recorridos realizados por ambos robots y parte de la barrera que divide sus áreas. También se observa la cuenta de flores que realizaron en la ciudad.

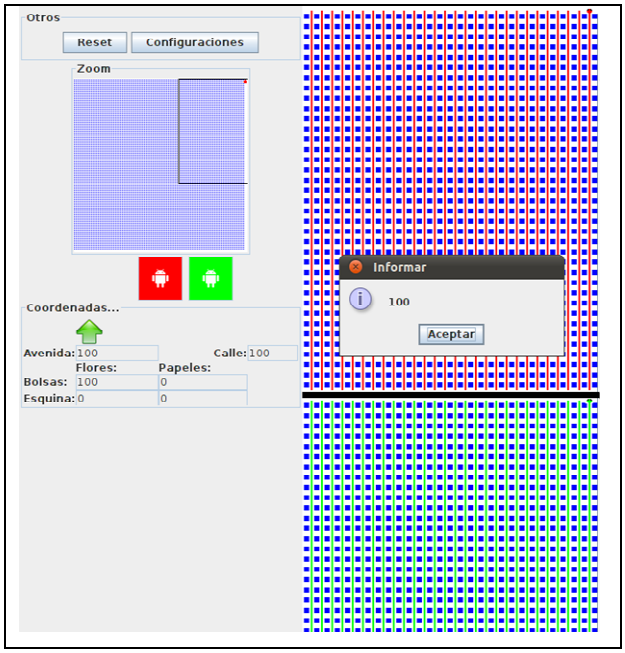

Figura 8: Prototipo del entorno LMRE.

Esta captura de pantalla corresponde al prototipo en desarrollo, y se tendrá una versión utilizable por los alumnos de CS1 en el 2do. Semestre de 2012.

6. Conclusiones y Líneas de Trabajo

Se ha presentado un análisis conceptual sobre la inclusión de temas de concurrencia y

paralelismo en un primer curso de Algorítmica, centrando los aspectos técnicos en la utilización de un entorno "multirobot" (LMRE) que simula la actividad de múltiples procesadores permitiendo analizar y resolver problemas básicos de concurrencia.

Dados los avances tecnológicos de las arquitecturas paralelas, la enseñanza de estos conceptos se vuelve una pieza muy importante en la formación de futuros profesionales en el área informática.

La herramienta desarrollada extiende las potencialidades del entorno Visual Da Vinci y permite exponer claramente los mecanismos de comunicación y sincronización entre múltiples procesos, así como la competencia por recursos compartidos.

El entorno permite la interacción en el análisis de los problemas, su ejecución paso a paso por el alumno y la visualización del movimiento sincrónico de los robots (procesos), favoreciendo la comprensión intuitiva de los conceptos básicos de concurrencia.

Derivado de este trabajo se desarrollan varias líneas de I/D, por un lado extendiendo primitivas y por otro pensando en la lógica de alumnos de los cursos de Programación Concurrente (3er. año) y Sistemas Paralelos (4to. Año). Entre ellas:

- Incorporar barreras y barreras selectivas como mecanismo de sincronización.
- Agregar una primitiva de posicionamiento a cada robot, que permite tomar decisiones en tiempo real.
- Utilización de variables compartidas en áreas tipo C y PC.
- Incorporar tiempo y duración específica a las actividades de los robots (por ej. recoger o depositar objetos) y también poder tener velocidades de movimiento diferentes para modelizar la heterogeneidad.
- Analizar el manejo de mensajes asincrónicos y su impacto en el entorno.

7. Referencias

[1] Force A.-C. J. C. T., "Computer science curriculum 2008: An interim revision of cs 2001", tech. rep., ACM Press, dec 2008.

[2] De Giusti A., Frati F. E. "¿Concurrencia y Paralelismo en el primer curso de Algorítmica?". V Congreso de Tecnología en Educación y Educación en Tecnología, 2010.

[3] Champredonde R., De Giusti A. "Design and implementation of the visual da vinci language". Tesina de grado Facultad de Informática UNLP, 1997.

[4] De Giusti A., Frati F. E., Leibovich F., Sanchez M., De Giusti L., Madoz M. C. "LMRE: Un entorno multiprocesador para la enseñanza de conceptos de concurrencia en un curso CS1". XVII Congreso Argentino de Ciencias de la Computación, 2011.

[5] De Giusti A. et al, "Algoritmos, datos y programas con aplicaciones en Pascal, Delphi y Visual Da Vinci", Pearson Education and Prentice Hall, 1 ed., 2001.

[6] Held J., Bautista J., Koehl S. "From a few cores to many: A tera-scale computing research overview". Intel Corporation, 2006.

[7] Ivica C., Riley J. T., Shubert C. "StarHPC - Teaching Parallel Programming within Elastic Compute Cloud", Proceedings of the ITI 2009 31st. Int. Conf. on Information Technology Interfaces, 353-356, 2009.

[8] Rague B. "Teaching parallel thinking to the next generation of programmers", Journal of Education, Informatics and Cybernetics, vol. 1, no. 1, pp. 43–48, 2009.

[9] Gross T. R. "Breadth in depth: a 1st year introduction to parallel programming", in Proceedings of the 42nd ACM technical symposium on Computer science education, pp. 435–440, 2011.

[10] De Giusti A., Frati F. E., Sanchez M., De Giusti L. "LIDI Multi Robot Environment: Support software for concurrency learning in CS1". CE Learning, (Denver, Colorado, USA) 2012.

[11] Grama A., Gupta A., Karypis G., Kumar V. "Introduction to Parallel Computing", Pearson Addison Wesley, 2nd Edition, 2003.

[12] Chapman B., The Multicore Programming Challenge, Advanced Parallel Processing Technologies; 7th International Symposium, (7th APPT'07), Lecture Notes in Computer Science (LNCS), Vol. 4847, p. 3, Springer-Verlag (New York), November 2007.