Un enfoque basado en la resolución de problemas para la enseñanza de la POO

Sonia V. Rueda **Iris P. Señas**

Laboratorio de Investigación y Desarrollo en Informática y Educación (LIDInE)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur - Bahía Blanca
svr@cs.uns.edu.ar - ips@cs.uns.edu.ar

La enseñanza de la programación en las materias iniciales de las carreras de Ciencias de la Computación ha sido siempre un desafío importante. En los nuevos planes de estudio de Licenciatura en Ciencias de la Computación e Ingeniería en Sistemas de Computación hemos adoptado desde el primer año el paradigma de programación orientada a objetos con un enfoque basado en la resolución de problemas. El objetivo de la asignatura Introducción a la Programación Orientada a Objetos es que los alumnos adquieran más tempranamente en la carrera la capacidad de interpretar un diagrama de clases escrito en un lenguaje de diseño y de implementarlo adecuadamente en un lenguaje de programación orientado a objetos y usando librerías, a la vez que adquieren contenidos conceptuales básicos de programación y del paradigma en particular.

Ambas carreras son dictadas por el Departamento de Ciencias e Ingeniería de la Computación de la Universidad Nacional del Sur.

1. Introducción

La enseñanza de la programación en las materias iniciales de las carreras de ciencias de la Computación ha sido siempre un desafío importante. Numerosas líneas de investigación coinciden en que los alumnos deben adquirir destreza en el uso de uno o varios lenguajes de programación, pero también deben aprender algunos conceptos fundamentales en la disciplina y sobre todo desarrollar capacidades vinculadas al pensamiento formal.

Los enfoques basados en la resolución de problemas resultan adecuados para integrar los aspectos mencionados. Las fases que abarca el proceso de resolución de un problema en general pueden asimilarse a las etapas de desarrollo de una aplicación de software. El aprender **desde** la resolución de problemas permite desarrollar la capacidad de abstracción partiendo de la interpretación de la especificación hasta la construcción de un modelo. En estos enfoques el lenguaje de programación tiene un rol importante pero no protagónico. Del mismo modo que en el proceso de desarrollo de software el lenguaje de programación es una de las herramientas utilizadas; en los contenidos de una materia de programación el aprendizaje de esta y otros recursos debería quedar de alguna manera subordinado a un objetivo más ambicioso: desarrollar aplicaciones que permitan resolver problemas de pequeña y mediana complejidad. Es más, la estructura y una buena parte de los contenidos de una materia pueden mantenerse aún cuando el lenguaje de programación se reemplace por otro.

Una decisión más significativa es la elección del paradigma de programación con el cual los alumnos se iniciarán en la disciplina. En este sentido hay experiencias realizadas con el paradigma declarativo y aplicativo, aunque durante años el modelo imperativo ha sido el más habitual para los cursos iniciales.

En los últimos años el paradigma de programación orientada a objetos (POO) se ha consolidado y ha ido desplazando al modelo imperativo en el desarrollo de aplicaciones reales y este hecho ha provocado la necesidad de revisar los currículos. La POO no debería

presentarse ya en materias avanzadas, sino que debería anticiparse de alguna manera. En este sentido es importante considerar que las necesidades del mercado exigen incluir en los contenidos básicos de las carreras de computación cada vez más temas y los núcleos curriculares no pueden crecer indefinidamente. Al incorporarse nuevos contenidos, se debe reducir la presentación de otros.

Nuestra propuesta es entonces disminuir la participación del paradigma imperativo en las materias iniciales de programación y adoptar ya en el primer año de la carrera el paradigma de programación orientada a objetos. Esta modificación permite presentar en las materias iniciales contenidos de ingeniería de software que hasta el momento se introducían en materias más avanzadas.

El cambio de paradigma se realizó en el contexto de una modificación en los planes de de la Licenciatura en Ciencias de la Computación y de la Ingeniería en Sistemas de Computación, que implicó varios cambios significativos. La intención fue mantener los perfiles en ambas carreras y respetar el núcleo curricular propuesto por la Red UNCI, lo cual implica necesariamente actualizar la estructura de los planes y los contenidos de las materias.

2. Alternativas de Diseño Curricular

En **[Meyer 93]** y **[Meyer 03]** Bertrand Meyer recomienda rediseñar la enseñanza de la programación en las universidades a partir de los fundamentos de la POO. La idea es no retrasar la introducción de conceptos y técnicas orientadas a objetos hasta llegar a las materias avanzadas, sino comenzar en las iniciales.

La propuesta se contrapone a otras más conservadores basadas en un enfoque evolutivo, esto es, presentar los paradigmas de programación más significativos siguiendo el orden en el que surgieron históricamente.

Meyer considera que la aplicación temprana de los métodos OO, favorece el desarrollo del pensamiento formal de los alumnos y permite adquirir destrezas fundamentales para la construcción de software a gran escala. En lugar de aplicar un estilo botton-up o top-down, la propuesta es *outside-in*, esto es, aprender a **hacer** software de buena calidad, **usando** software de buena calidad.

El método de enseñanza está basado en la aplicación sistemática del **diseño por contrato**, esto es, el uso de librerías de componentes reusables clasificadas en niveles (estructuras de datos, gráficos, etc.). El código fuente de las librerías está disponible y constituye un modelo a imitar de alta calidad. Sin embargo, en los niveles iniciales, la expectativa es que los alumnos se familiaricen únicamente con las interfaces y se abstraigan del código específico, adoptando el estilo de producción de software propio de la POO. Este diseño curricular, que ha sido aplicado intensamente durante los últimos 15 años por el autor y su grupo de trabajo, plantea una forma de **currículo invertido** en el cual los alumnos comienzan como **consumidores** de componentes reusables para convertirse luego en **productores**.

En nuestro Departamento el trabajo se orientó durante varios meses a analizar el diseño curricular propuesto por Meyer y evaluar la posibilidad de aplicarlo en nuestras carreras. El diseño por contrato es un método de desarrollo de software formal, con un fuerte fundamento matemático. En nuestras carreras los alumnos logran esa formación matemática recién en el segundo año. Una alternativa era entonces comenzar con las materias de programación una vez que los alumnos hubiesen adquirido las competencias previas requeridas para comprender métodos formales. Sin embargo, atendiendo el aspecto motivacional, nuestra consigna ha sido siempre incluir materias de programación desde el primer año de la carrera. Esto implica que dichas materias no pueden requerir una formación matemática previa mayor a la que brinda la escuela media, claramente

insuficiente para adoptar el esquema de Meyer.

Nuestra propuesta incluye en el primer año dos materias de programación. La primera adopta el paradigma imperativo y un enfoque bottom up tradicional para presentar los conceptos de variable, expresión, tipo, instrucción, estructura de control, etc. En esta materia desarrollan programas simples aplicando diseño top-down y programación estructurada. En la segunda materia se presentan las limitaciones del paradigma imperativo para soportar reusabilidad y extensibilidad de software y se plantea el paradigma OO como la alternativa adecuada para el desarrollo de aplicaciones de mediana y gran escala. Este proceso involucra varias etapas en cada una de las cuales se requieren capacidades diferentes.

Las etapas de especificación, análisis y diseño demandan altos niveles de abstracción y en aplicaciones de mediana y gran escala resulta adecuado utilizar métodos formales. De modo que las capacidades inherentes a estas etapas escapan de las expectativas del primer año. Lo que consideramos que un alumno de primer año sí puede hacer es **interpretar** un diseño ya elaborado a partir de un lenguaje de modelado e implementarlo en un lenguaje de programación.

En los trabajos prácticos y proyectos los alumnos parten de una especificación y un diagrama de clases en UML que retiene los aspectos relevantes de esta especificación. El diseño sugiere en sí mismo la utilización de paquetes de componentes reusables. En forma transversal a la construcción del programa se presentan los conceptos inherentes al paradigma y el soporte provisto por el lenguaje para cada uno de ellos.

En las dos materias de programación correspondientes al segundo año, además de los temas propios de estructuración de datos y algoritmia, se incluyen contenidos específicos de ingeniería de software. Los alumnos comienzan a estar entonces en condiciones de enfrentar las etapas iniciales del proceso de desarrollo. En todas las materias el enfoque adoptado es aprender a programar *desde* la resolución de problemas.

3. Aprender *desde* la resolución de problemas

Durante muchos años, el énfasis en la enseñanza estuvo centrado en transmitir conceptos y procedimientos cuyo significado con frecuencia no llegaba a comprenderse. Una visión alternativa consiste en considerar el aprendizaje como una construcción social que incluye conjeturas, pruebas y refutaciones y cuyos resultados serán analizados en un contexto social y cultural. La idea subyacente es que **saber es hacer** y el hacer es un proceso creativo y generativo. Los alumnos se comprometen con actividades que plantean situaciones problemáticas cuya resolución requiere analizar, descubrir, elaborar hipótesis, confrontar, reflexionar, argumentar y comunicar ideas.

En este contexto el término **resolución de problemas** ha tomado un gran auge en los últimos tiempos, creciendo su inclusión en disciplinas como Matemática, Física, Química e Informática. Sin embargo, este término ha sido usado con diversos significados y acompañó a diferentes concepciones. Tampoco hay una interpretación única para la palabra **problema** y lo que para un autor puede constituir un problema, puede no serlo de acuerdo con la definición de otro. A continuación, presentamos algunas definiciones de autores reconocidos en el área y describimos el enfoque conceptual adoptado en este trabajo.

3.1 ¿Qué es un problema?

Una de las principales contribuciones en el desarrollo de estrategias didácticas basadas en la resolución de problemas ha provenido, hasta el momento, de la Matemática. Numerosas publicaciones se ocupan de analizar qué supone la resolución de problemas como actividad cognitiva y ofrecen algunas propuestas de sistematización para la

misma. El aporte de Polya es reconocido como el punto de partida a partir del cual surgieron numerosos trabajos de investigación, destinados a analizar y comprender los métodos que conducen a la resolución de problemas y, en particular, las operaciones involucradas en ese proceso.

En 1965, en su libro *Mathematical Discovery*, Polya indica que tener un problema significa "*buscar de forma conciente una acción apropiada para lograr un objetivo claramente concebido, pero no alcanzable en forma inmediata*" **[Polya 65].**

Casi tres décadas después, Parra define un problema como *"una realidad incompleta, una pregunta que demanda una respuesta, una pulsión, una incitación a salir de un estado de desequilibrio a otro de equilibrio"*. Pero agrega también: *"un problema lo es en la medida en que el sujeto al que se le plantea (o que se plantea él mismo) dispone de los elementos para comprender la situación que el problema describe y no dispone de un sistema de respuestas totalmente constituido que le permita responder de manera inmediata"* **[Parra 90].**

En el área de Ciencias de la Computación, Newell y Simon elaboraron en 1972 un marco teórico y un programa que simulaba el proceso de resolución de problemas de un humano. Este aporte tuvo un impacto significativo en la evolución de la psicología cognitiva **[Newell 72]**. En este contexto, "*un problema puede pensarse como una discrepancia entre un estado inicial y un estado final que constituye la meta a alcanzar*". Esta concepción es central en Ciencias de la Computación, porque su evolución no ha sido producto de un avance puramente tecnológico, sino que también está ligado a la descripción de los hechos y de los fenómenos de la mente y de la naturaleza, y la manera en que estos hechos y fenómenos se producen.

3.2 ¿Qué significa un enfoque basado en resolución de problemas?

Desde la década del '80, la frase resolución de problemas se ha utilizado en educación con diferentes concepciones. Las siguientes son tres alternativas frecuentes:

- **Aprender *desde* la resolución de problemas**
- **Aprender *sobre* la resolución de problemas**
- **Aprender *a* resolver problemas**

Aprender *desde* la resolución de problemas implica abordar los contenidos conceptuales de una asignatura específica a partir de problemas, probablemente cercanos a la vida cotidiana, que constituyan de alguna manera un desafío. Esta alternativa es atractiva, motivadora y puede aplicarse en diferentes asignaturas, pero puede demandar una cantidad considerable de tiempo. Con frecuencia, los profesores plantean ejercicios para practicar y reforzar los temas presentados. Algunos de ellos pueden estar especificados a través de un enunciado que describe una aplicación para el tema presentado. Sin embargo, si la resolución va a demandar únicamente interpretar el enunciado y aplicar el procedimiento o las herramientas que se acaban de presentar, desaparece parte de la dificultad que caracteriza a un problema.

Aprender *sobre* la resolución de problemas se refiere a aprender acerca de los procesos y estrategias cognitivas que aplicamos cuando estudiamos, resolvemos problemas o procesamos información. La intención es que los alumnos adquieran conciencia sobre sus procesos (percepción, atención, comprensión, memorización, comunicación) y sus estrategias (ensayo, elaboración, organización, estudio), y desarrollen habilidades para controlarlos en forma consciente y deliberada. Esta concepción puede ser muy enriquecedora pero en nuestro caso se aleja de los objetivos fundamentales de las materias básicas de programación.

Aprender *a* resolver problemas es la alternativa que se aplica en asignaturas y talleres diseñados específicamente para que

los alumnos adquieran destreza en la resolución de problemas de diferentes áreas. También es habitual que los profesores de materias básicas de programación destinen algunas clases a proponer problemas diversos a sus alumnos. En general, se incluyen problemas de ingenio, enigmas, acertijos y problemas de aplicación real. Con frecuencia los alumnos logran alcanzar intuitivamente soluciones para las que quizá existen métodos formales que ellos todavía desconocen.

En esta alternativa, los problemas deben constituir un desafío, pero al mismo tiempo deben ser una meta alcanzable. En términos del concepto de la zona de desarrollo próximo de Vigotsky, la dificultad debe ser algo superior a la zona de dominio o capacidad del alumno **[Vigostky 79].** Esta situación es extremadamente enriquecedora, pero encontrar la zona de desarrollo próximo de un grupo heterogéneo puede no ser sencillo. Una alternativa es que al menos algunas de las actividades se planteen con diferentes grados de dificultad, de modo que puedan adaptarse a las competencias previas de los distintos miembros del grupo.

Muchos alumnos se sienten atraídos por este tipo de propuesta, pero con frecuencia estas clases tienen una fuerte componente de entretenimiento que las hace atractivas, aunque poco estructuradas y sin un objetivo preciso. Cuando los grupos son heterogéneos y las estrategias cognitivas muy diversas, resulta difícil mantener un nivel de participación uniforme. El riesgo es que la resolución quede muy ligada a las capacidades naturales de cada alumno, sin que efectivamente haya una contribución para aumentarlas.

Otro de los inconvenientes es que, como en el enfoque anterior, aunque la resolución de enigmas y problemas de ingenio pueda desarrollar capacidades significativas para la programación, el proceso es lento y en general en una materia de programación hay objetivos específicos que deben alcanzarse a través de actividades más concretas.

Las tres concepciones tienen sus ventajas y sus limitaciones y corresponden a un **enfoque basado en la resolución de problemas**. En este trabajo el objetivo es en aprender a programar **desde** la resolución problemas, de modo que adoptamos fundamentalmente la primera alternativa. Sin embargo, intentamos también favorecer el análisis, la exploración y la reflexión acerca de estrategias que orientan el proceso de búsqueda de la solución, acercándonos de alguna manera al aprendizaje **sobre** la resolución de problemas.

3.3 ¿Qué factores intervienen en el proceso de resolución de problemas?

Hasta el momento, no existe un marco explicativo completo acerca de cómo se aprende a resolver problemas. Desde las Ciencias de la Computación, Dijkstra **[Dijkstra 65]**, **[Dijkstra 97]** define la resolución de problemas como un proceso cognitivo complejo en el que se aplica el conocimiento almacenado en la memoria a corto y a largo plazo y que involucra factores de naturaleza cognitiva, afectiva y motivacional.

Dentro de las investigaciones vinculadas a la resolución de problemas matemáticos, el aporte de Polya **[Polya 45], [Polya 65]** fue extendido por Schoenfeld **[Schoenfeld 83]**, **[Schoenfeld 85]**, quien considera, no sólo las operaciones cognitivas involucradas en esta actividad, sino también las metacognitivas.

4. La organización de la clase y la selección de los problemas

Gran parte de las investigaciones de los últimos años consideran el aprendizaje como una actividad social y constructiva. El constructivismo se mueve entonces desde una esfera puramente epistemológica hacia la práctica pedagógica **[Carretero 93]**. Desde esta perspectiva, el alumno no es un receptor dentro de un proceso de instrucción, sino que tiene una actitud participativa en un proceso interactivo.

El rol fundamental del docente es diseñar cuidadosamente un entorno que favorezca la adquisición de los conceptos y las formas de razonamiento propuestos. Notemos que, cuando hablamos de entorno, no estamos refiriéndonos exclusivamente a las actividades, sino también a las interacciones que deberían surgir para provocar cierto tipo de situaciones. Crear un entorno interactivo para el aprendizaje puede resultar difícil por varios motivos.

En primer lugar, el docente debe decidir cuándo intervenir y qué sugerencias pueden ayudar a sus alumnos ante una situación de bloqueo, sin que su intervención los conduzca directamente a la solución. Para lograrlo, debe intentar percibir el proceso de pensamiento de los miembros del grupo mientras buscan la solución y no sólo de los individuos aislados.

En segundo lugar, es poco común que los alumnos vean a sus profesores **pensando** acerca de resolución de un problema. Con frecuencia, los docentes organizan sus clases como una exposición, en la cual presentan la solución final y de alguna manera describen cómo llegaron a ella. Aunque resulte paradójico, si los alumnos entienden rápidamente la solución propuesta, la clase puede resultar poco efectiva. Aquellos que han intentado sin éxito resolver el problema previamente, sienten cierta frustración al no haber logrado hallar la solución, que ahora ven surgir tan fácil y naturalmente. Los que enfrentaban el problema por primera vez en ese momento, pueden quedar con la falsa impresión de que la resolución es rápida y sencilla, cuando en realidad la explicación puede haberlo sido, pero el alumno no participó en el proceso, aunque pueda entenderlo. Una situación similar se produce cuando los estudiantes trabajan en grupo: aun cuando todos terminen comprendiendo el procedimiento que los condujo a la solución, probablemente sólo algunos sean capaces de desarrollarlo de manera autónoma.

Una manera de mejorar la capacidad de resolución de problemas de los alumnos, es embarcarse con ellos en la resolución. Esta modalidad requiere de un alto nivel de improvisación en la clase y aumenta el riesgo de desorganización. Los alumnos realizan diferentes aproximaciones hacia la solución y el docente debe analizar dinámicamente cuán fructíferas pueden ser. Trabajar bajo esta modalidad requiere experiencia, confianza y demanda una cantidad considerable de tiempo.

Una alternativa es entrelazar actividades promoviendo diferentes niveles de interacción. En algunas, el docente propone un problema, seleccionado expresamente para provocar la necesidad de alguna forma de análisis o razonamiento particular, organiza la resolución y realiza sugerencias apropiadas. Luego, pueden plantearse otros problemas, de alguna manera vinculados con el anterior, pero algo más sencillos. En algunos casos la vinculación estará centrada en el conocimiento de base y en otros en las estrategias heurísticas. Los alumnos trabajan en forma individual o en grupos y luego confrontan las diferentes soluciones propuestas.

La selección de los problemas debe hacerse considerando especialmente los contenidos conceptuales de la materia. En este caso referidos a la POO y el lenguaje de programación adoptado. A partir de ellos se elegirán problemas que favorezcan el trabajo con estrategias heurísticas y el desarrollo de la metacognición.

El diseño de las actividades está orientado entonces a:

- Interpretar modelos OO referidos a situaciones reales, especificados en un lenguaje de diseño
- Implementar una solución en un lenguaje de programación que satisfaga los requerimientos establecidos en el enunciado y sea consistente con el modelo
- Desarrollar habilidades para analizar y comprender enunciados de problemas, enfatizando la importancia de que cada palabra, frase y oración haya sido

comprendida adecuadamente antes de elegir o aplicar una estrategia de resolución.

- Estimular la búsqueda de soluciones alternativas y la aplicación de estrategias heurísticas, desarrollando distintas formas de razonamiento.
- Favorecer la reflexión y la discusión acerca de las distintas estrategias y formas de razonamiento.
- Lograr mayor exactitud y precisión en el lenguaje utilizado al trabajar con un problema.
- Desarrollar una actitud positiva ante el error, usándolo como recurso de aprendizaje.
- Aumentar la perseverancia y el esfuerzo por superar situaciones de bloqueo.
- Provocar un desafío cognitivo

Con frecuencia, un enunciado que para un grupo de alumnos representa un problema, será sólo un ejercicio para otro, pero en conjunto, todos deberían enfrentar alguna forma de desafío en cada clase. En cualquier caso, estos enunciados deberían demandar cierto esfuerzo de interpretación.

En la selección de los problemas y en la planificación de las clases, es muy importante destinar un espacio para que los alumnos reflexionen acerca de las dificultades que planteó cada problema y de las estrategias de resolución empleadas. Las dificultades pueden no haber sido las mismas para todos y también es muy probable que surjan diferentes caminos a través de los cuales se halló la solución. Este tipo de actividad apunta a que los alumnos tomen conciencia acerca de sus capacidades cognitivas y aprendan a controlarlas, en un contexto concreto. La actitud del alumno es fundamental porque de su compromiso con la propuesta depende en parte que pueda avanzar en la resolución.

La puesta en común permite que cada alumno describa el camino que siguió para llegar o intentar llegar a la solución y las dificultades que encontró. Cada propuesta de resolución exitosa probablemente haya sido el camino elegido por varios alumnos, que con alguna variación, habrán aplicado las mismas estrategias. También los errores suelen repetirse, la mayoría de las veces, el docente anticipó estas situaciones, pero en algunos casos puede aparecer alguna cuestión no prevista. En cualquier caso, la reflexión acerca del error mejora la interpretación del problema, no sólo de quienes lo cometieron, sino probablemente de otros miembros del grupo.

5. El proceso de resolución de un problema

La resolución de problemas es una tarea que requiere tiempo y esfuerzo, pero al mismo tiempo puede resultar una experiencia placentera y motivadora. Resolver un problema tiene algo de descubrimiento, aumenta nuestro conocimiento, aporta nuevos puntos de vista, y mejora nuestra capacidad para enfrentar nuevas situaciones en el futuro.

El proceso de resolución de un problema involucra varias etapas:

Analizar y comprender el enunciado. Esta etapa parece obvia y normalmente es la que se atraviesa con mayor celeridad. Sin embargo, es fundamental y muchas veces la situación de bloqueo ante un problema se debe justamente a que no se ha entendido completamente.

Construir la solución: En esta etapa se elige una o un conjunto de estrategias combinadas y a partir de su aplicación se alcanza una solución.

Verificar la solución: La etapa final es confrontar los resultados obtenidos con el problema planteado, verificando que la solución sea correcta.

Es evidente la vinculación entre estas etapas y las que conforman el ciclo de vida del software en el modelo en cascada:

1. Especificación de requerimientos
2. Análisis y Diseño

3. Implementación
4. Verificación
5. Mantenimiento
6. Documentación

Este modelo es presentado en las clases iniciales de la segunda materia de programación para enfatizar la importancia de concebir a la programación como un proceso. Sin embargo, en los problemas presentados no abordamos cada una de estas etapas por varios motivos. En primer lugar en un curso básico de programación no trabajamos con usuarios de modo que la especificación de requerimientos está dada por el enunciado del problema y nos ocupamos más bien de su interpretación adecuada. En segundo término, aunque esto ya no es una práctica convencional, los enunciados de los problemas planteados incluyen también un diagrama de clases, de modo que las etapas de análisis y diseño también han sido atravesadas previamente. Los alumnos interpretan modelos ya elaborados en un lenguaje de diseño.

De modo que la etapa **Analizar y comprender el enunciado** consiste en interpretar la especificación de requerimientos y el diseño de una aplicación. La etapa de **Construir la solución** se corresponde con la Implementación de la solución usando un lenguaje de programación, que va a ser seguida por la verificación respecto al enunciado.

La etapa de mantenimiento es también de reproducir en una materia inicial, cuando las aplicaciones son simples. Sin embargo, algunas de las actividades propuestas tienen como objetivo partir de un problema resuelto y modificarlo a partir de un cambio en la especificación. La documentación es una actividad fundamental en el proceso de desarrollo de software, transversal a las demás etapas. En nuestro caso enfatizamos la importancia de incluir comentarios adecuados en el código, consistentes con lo especificado en el diagrama UML.

Conclusiones

El enfoque adoptado en este trabajo considera a la POO como un paradigma muy adecuado para desarrollar las compentencias intelectuales que requiere el desarrollo de software, desde las materias iniciales. A partir de esta consideración se elaboraron los planes de estudio de las carreras dictadas en el Departamento de Ciencias e Ingeniería de la Computación de la Universidad Nacional del Sur.

La modificación de los planes se consensuó con todos los integrantes del cuerpo docente y se realizó considerando dos aspectos adicionales, relacionados entre sí. Por un lado, adoptar en las materias iniciales paradigmas y lenguajes de programación que se utilizan en el desarrollo de aplicaciones comerciales, resulta muy motivador para los alumnos.

Por otra parte, una cantidad muy importante de alumnos comienza a trabajar en actividades relacionadas a la disciplina cuando cursa segundo o tercer año de la carrera. El aprender conceptos y herramientas que van a aplicar en su trabajo, les permite no solo encarar con mayor solvencia sus primeras experiencias laborales, sino también tener una mejor percepción de la escala de problemas que pueden ir resolviendo a medida que avancen en la carrera.

Bibliografía

[Alonso 01] Alonso, I. *La resolución de problemas matemáticos. Una alternativa didáctica centrada en la representación*. Tesis Ph. D. Universidad de Oriente. Cuba. (2001).

[Ausubel 88] Ausubel, D., Novak, J. & Hanesian, H. *Psicología cognitiva: Un punto de vista cognoscitivo.* Méjico. Trillas. (1988)

[Carretero 93] Carretero M., *Construct-ivismo y Educación*. Aique, (1993).

[Cohen 91] Cohen B. *The Inverted Curriculum*, Report, National Economic Development Council, London, 1991.

[Dijkstra 65] Dijkstra, E. *Programming Considered as a Human Activity*. Proceedings of the IFIP Congress (1965)

[Dijkstra 97] Dijkstra, E. *A Discipline of Programming.* Prentice Hall (1997)

[Huertas 03] Huertas, J. *Motivación y Aprendizaje* FLACSO (2003)

[Meyer 93] Meyer B., *Towards an Object-Oriented Curriculum*, in *Journal of Object-Oriented Programming*, vol. 6, no. 2, May 1993, pages 76-81. Revised version in *TOOLS 11 (Technology of Object-Oriented Languages and Systems*), eds. R. Ege, M. Singh and B. Meyer, Prentice Hall, Englewood Cliffs (N.J.), 1993, pages 585-594.

[Meyer 97] Meyer B., *Object-Oriented Software Construction, 2nd edition*, Prentice Hall, 1997, especially chapter 29, *Teaching the Method*.

[Meyer 01] Meyer B., *Software Engineering in the Academy*, in Computer (IEEE), vol. 34, no. 5, May 2001, pages 28-35.

[Meyer 03] Meyer B., *The Outside-In method of teaching introductory programming*, 2003.

[Mingins 99] Mingins C., Miller J., Dick M., Postema M., *How We Teach Software Engineering, in Journal of Object-Oriented Programming* (JOOP), vol. 11, no. 9, 1999, pages 64-66, 74.

[Murphy 97] Murphy E. *Constructivism: From Philosophy to Practice* (1997)

[**Pedroni 05] Pedroni M. and Meyer B.** *The Inverted Curriculum in Practice*, SIGCSE 2006, Houston, 1-5 March 2006.

[Newell 72] Newell, A., & Simon, H. A. *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall. (1972).

[Parra 90] Parra, B. *Dos concepciones de resolución de problemas*, Revista Educación Matemática, vol. 2, núm. 3, pp. 22-31 (diciembre 1990)

[Polya 45] Polya, G. *How To Solve It: A New Aspect of Mathematical Method*, Princeton University Press, 1945,1957,1973.

[Polya 65] Polya, G. *Mathematical Discovery: On Understandig, Learning and Teaching Problem Solving.* John Wiley & Sons. (1965)

[Pozo 99] Pozo, J.J. y Gómez Crespo *Aprender y Enseñar Ciencia* Morata Madrid (España), 1999

[Rueda 89] Rueda, S., Castro, S. & Zanconi, M. *Resolución de Problemas y Algoritmos*. Notas del curso. Universidad Nacional del Sur. Argentina (1989)

[Rueda 02] Rueda, S. & García, A. *Análisis y Comprensión de Problemas: Fundamentos, Problemas Resueltos y Problemas Propuestos*. Notas del curso de nivelación. Universidad Nacional del Sur. Argentina (2002)

[Schoenfeld 83] Schoenfeld, A. *Ideas y tendencias en la Resolución de Problemas* en el libro "La enseñanza de la matemática a debate". (pp. 7-12). Ministerio de Educación y Ciencia. Madrid. España. (1983).

[Schoenfeld 85] Schoenfeld, A. *Sugerencias para la enseñanza de la Resolución de Problemas Matemáticos* en el libro "La enseñanza de la matemática a debate". (pp.13-47). Ministerio de Educación y Ciencia. Madrid. (1985).

[Vigostky 79] Vigostky, L. *El desarrollo de los procesos psicológicos superiores*. Barcelona. Crítica. (1979)

[Wirth 02] Wirth N. *Computer Science Education: The Road Not Taken*, opening address at ITiCSE conference, Aarhus, Denmark, June 2002, available (September 2003)

[Wirth 02] Wirth N. *Computer Science Education: The Road Not Taken*, opening address at ITiCSE conference, Aarhus, Denmark, June 2002, available (September 2003)