SISTEMAS DE NUMERACIÓN: UNA METODOLOGÍA DE ENSEÑANZA BASADA EN EL ENFOQUE ALGORÍTMICO

Marcia Mac Gaul – Marcela F. López

Universidad Nacional de Salta. Facultad de Ciencias Exactas
mmacgaul@cidia.unsa.edu.ar – mfflopez@unsa.edu.ar

Resumen

Este trabajo está escrito por docentes-investigadores que trabajan principalmente en las asignaturas básicas de Programación, de la carrera Licenciatura en Análisis de Sistemas de la Universidad Nacional de Salta. Además dirigen Proyectos de Investigación y Desarrollo acreditados por el Consejo de Investigaciones y un PICTO acreditado en la ANPCyT.

Se describe el diagnóstico de situación del alumnado inicial, la cátedra, la variable tecnológica de contexto y la relevancia del problema de enseñar contenidos clásicos de ciencias de la computación, desde un enfoque de resolución de problemas computacionales a través del diseño algorítmico. Se refiere brevemente una aplicación desarrollada para la construcción de algoritmos y la funcionalidad que brinda para la depuración y documentación. Este software provee una galería de componentes reutilizables que constituyen la base de una metodología desarrollada para la creación de algoritmos computacionales. Se ilustra la metodología aplicándola a un caso práctico: la conversión de números enteros expresados en distintos sistemas de numeración. El enfoque algorítmico transversalmente asociado a éste y otros contenidos de la asignatura procuran potenciar las competencias de abstracción en diversas situaciones problemáticas propias de la Computación y la reutilización y adecuación de componentes para un diseño algorítmico de calidad.

Palabras claves: Algoritmos – Diagramas – Sistemas de Numeración – Metodología de enseñanza – Herramientas.

Introducción

Este trabajo está escrito por informáticos, cuyas carreras docentes se han desarrollado, principalmente, en las asignaturas básicas de Programación, de la carrera Licenciatura en Análisis de Sistemas de la Universidad Nacional de Salta. Además dirigen, integran e integraron Proyectos de Investigación y Desarrollo[1] acreditados por el Consejo de Investigaciones de la misma Universidad y un Proyecto de Investigación Científica y Tecnológica Orientado (PICTO)[2] acreditado en la Agencia Nacional de Promoción Científica, Tecnológica y de Innovación (ANPCyT).

Desde esa experiencia se sostiene que la docencia universitaria y la investigación son actividades complementarias. Los autores, en su rol de profesores aplican estrategias didácticas elaboradas a partir de los resultados de la investigación y en ésta, usan el aula como un verdadero laboratorio a partir del cual se registran datos, se efectúan conjeturas, se contrastan resultados y se alcanzan algunas conclusiones que se trasladan a la práctica docente, enriqueciéndola.

A lo largo de estos años de trabajo, la transferencia del grupo se concretó, principalmente, en el desarrollo de una aplicación que interpreta algoritmos, permitiendo su ejecución automática con fines

[1] Proyectos CIUNSa. Nº 1638 (2007-2009) *"Una estrategia metodológica: el uso de las NTICs en el ingreso masivo universitario"*y CIUNSa. Nº 1865/3 (2010-2013) *"Entornos Virtuales para la Articulación entre el Nivel Medio y carreras universitarias de Ciencias Exactas"*.

[2] PICTO Nº 36724 (2008-2010).

de prueba y documentación. Sin embargo, existe la certeza que esa producción estaría incompleta si no se complementara con el enfoque didáctico que se propone a partir de la investigación de los procesos cognitivos de los estudiantes iniciales de Programación. Por tanto, se reconoce la importancia de un detallado análisis del problema y su correspondiente diseño algorítmico, antes de traducir esa solución a un lenguaje de programación.

Diagnóstico de situación

La realidad de los ingresos masivos está afectada de una serie de problemas que impactan directamente en la calidad educativa. Entendido como un fenómeno multidimensional, parte de ellos corresponden a razones exógenas a la Universidad. Sin embargo, otros son propios y merecen especial atención, por lo que se enuncian a continuación como una aproximación diagnóstica que sustenta las acciones desarrolladas y las futuras líneas de trabajo.

El alumno inicial: el ingreso a la carrera es masivo. La matrícula de la carrera es elevada, con una inscripción media de seiscientos estudiantes. El alumno que concurre es, en su gran mayoría, un egresado reciente del nivel anterior. No siempre dispone de procesos cognitivos que le permita un grado de abstracción adecuado para permanecer en el primer año. Tampoco cuenta con hábitos ni estrategias de estudio que le permita consolidar su inclusión en el sistema superior.

La cátedra: el ingreso masivo se atiende con un plantel docente integrado por profesores y auxiliares. Los primeros están a cargo del dictado de contenidos en su mayoría teóricos que se imparten como clases magistrales a un numeroso grupo de alumnos, durante las instancias presenciales. No obstante, el uso de recursos tecnológicos imprimen una dinámica más participativa de los estudiantes frente a la ejecución de algoritmos clásicos cuya teoría sustentan. Las clases prácticas se desarrollan en diversas comisiones de cursado paralelo, cada una de ellas a cargo de un Jefe de Práctica y Auxiliares Docentes Alumnos. Parte de estas prácticas corresponden a actividades de programación en laboratorios de computadoras.
La asignatura Elementos de Programación integra el Plan de Estudio de L.A.S 2010. Está ubicada en el primer cuatrimestre de primer año, constituyendo así la primera materia del área de Computación que cursan los alumnos. Los contenidos pueden distinguirse en tres grandes ejes:
Conceptos iniciales de la programación, con énfasis en el diseño de algoritmos.
Elementos de computación básicos que se asientan sobre formalizaciones de la Matemática Aplicada.
Contenidos complementarios e introductorios orientados a la alfabetización informática.

La variable tecnológica: desde los proyectos de investigación se orientó hacia la construcción de un espacio en el que las competencias tecnológicas de los jóvenes se apliquen al ámbito educativo; en contraposición a otras estrategias que las ignoran, propiciando exclusivamente la cultura del papel. Desde el año 2008, la cátedra cuenta con un entorno virtual montado sobre la plataforma MOODLE, en el servidor e-cidia, perteneciente al Centro de Investigación y Desarrollo en Informática Aplicada de la UNSa. El diseño de este curso virtual se implementó a la par del debate y selección de nuevas estrategias didácticas acordadas por el cuerpo docente. Luego de cuatro años de experiencia, podemos afirmar que los docentes fueron tomando, paulatinamente, nuevos roles de tutoría más propios de los escenarios educativos actuales. La modalidad *extended-learning* adoptada prevé la asistencia de las estudiantes a instancias presenciales y la participación en actividades montadas sobre la plataforma, dirigidas principalmente al seguimiento de los aprendizajes en un marco de construcción grupal y colaborativo.
Por otra parte, el equipo de investigación, desarrolló e implementó desde el año 2009,

una aplicación de tipo escritorio denominada *Diagramar*. Se trata de un intérprete gráfico que permite crear, editar y ejecutar un Diagrama N-S[3] y visualizar la prueba de escritorio que se obtiene como resultado de ejecutar el diagrama.

Relevancia del problema: el Plan de Estudio de LAS vigente desde el año 2010, posee como principal diferencia con su antecesor del año 1997, que la asignatura inicial de Computación, modifica el orden de los contenidos impartidos según los ejes mencionados anteriormente. El contenido central de ambas materias cuatrimestrales, Elementos de Computación (Plan 1997) y su correspondiente Elementos de Programación del plan actual, es la resolución de problemas computacionales a través del diseño algorítmico. En el plan anterior, este contenido se desarrollaba en la segunda mitad del cuatrimestre, luego de presentar, entre otros, Sistemas de Numeración y Álgebra de Boole. La principal fortaleza de este diseño curricular, era la de trabajar con menor cantidad de alumnos un contenido históricamente reconocido como complejo. Esto era así debido a que un primer parcial eliminatorio, que evaluaba Sistemas de Numeración y Álgebra de Boole, reducía notablemente el número de estudiantes habilitados para continuar el cursado. Sin embargo, la principal debilidad radicaba en el escaso tiempo disponible para que un estudiante inicial llegara a construir estructuras lógicas de razonamiento, adecuadas para resolver exitosamente una variedad de problemas computacionales clásicos, tales como ordenamiento, búsqueda, inserción, eliminación y otros que son fundamentales para introducirse en la Programación. Esta debilidad se acentuaba al momento de aplicar estos algoritmos fundamentales a la resolución de problemas más complejos, cuyas narrativas están en un contexto de realidad y por lo tanto, se necesita la adaptación de estos componentes básicos y su combinación con otros elementos lógicos de la diagramación.

Para revertir esta situación en la nueva asignatura, los contenidos Resolución de Problemas Computacionales y Algoritmos fundamentales, se ubican en las cuatro primeras unidades del programa, mientras que Sistemas de Numeración y Álgebra de Boole se postergan para los últimos momentos del cursado. Este cronograma permite asignar al tema Algoritmos, cuatro meses de dedicación, propiciando así un abordaje más gradual de contenidos y técnicas. Respecto a la dificultad de trabajar este tema con un alumnado masivo e inicial, habilitamos un espacio de apoyo tutorial desde el entorno virtual, promoviendo el uso de la aplicación *Diagramar*, para concretar la reutilización de los algoritmos clásicos en la solución de problemas asociados con los temas subsiguientes de Sistemas de Numeración y Álgebra de Boole. De esta forma, **la cátedra adopta un enfoque algorítmico en el abordaje de todos los contenidos de la materia**, orientado a un proceso de aprendizaje gradual y contextualizado a problemas propios de la disciplina. Para la especificación algorítmica se utilizan principalmente los diagramas N-S; los cuales permiten realizar una representación gráfica del diseño de programas bajo el paradigma estructurado (también se especifican algoritmos usando pseudocódigo, pero mantener un vocabulario controlado y una adecuada identación para las estructuras de control resulta algo complejo para el estudiante inicial). La diagramación está basada en el diseño *top-down* que propone como estrategia para la resolución de un problema, dividirlo en subproblemas cada vez más pequeños y simples hasta obtener instrucciones elementales destinadas a la construcción de un programa. A esta metodología de diseño, se suma un importante concepto asociado a la reutilización de código. Su abordaje permite no solo trabajar sobre aspectos técnicos fundamentales para la construcción de software de calidad, sino que propicia un espacio para la reafirmación de conocimientos a través de la reutilización de

[3] Nassi-Schneiderman

las propias producciones de los estudiantes y, fundamentalmente, de algoritmos elementales que dimos en llamar ***componentes***, los cuales se encuentran definidos y documentados en forma previa e integran una galería de la aplicación *Diagramar*.

Surge de la investigación, que tradicionalmente, el diseño de los algoritmos se realiza en el aula presencial con papel y lápiz, herramientas que no permiten trabajar en profundidad las diferentes actividades involucradas, tales como la prueba y la documentación. A esto se suma la dificultad manifestada por los estudiantes para realizar satisfactoriamente las pruebas de sus algoritmos, lo que genera un alto grado de dependencia con el docente, cuyo rol se desvirtúa, constituyéndose en sólo un *probador de código*. Por otra parte, los docentes advierten que los estudiantes no comprenden cómo los diagramas que diseñan pueden transformarse en los programas reales que se ejecutan en una computadora. En general, no pueden captar la dinámica de funcionamiento de aquello que escriben en el papel, percibiendo al diagrama planteado como una descripción estática y no como un proceso dinámico en el que "suceden cosas" durante la ejecución de las instrucciones allí planteadas. En este aspecto, el rol del docente también se ve distorsionado, ya que era una práctica común utilizar la pizarra en el aula para mostrar el comportamiento dinámico de los algoritmos, sin obtener mayores éxitos.

A medida que el uso de la tecnología se fue incorporando como un recurso que colabora a mejorar la calidad de la propuesta educativa, los estudiantes manifestaron la necesidad de contar con un software que permitiera no sólo el diseño de algoritmos, sino también su ejecución, depuración y documentación y, sobre todo, les posibilitara un aprendizaje más autónomo. A partir de estos requerimientos, el análisis en profundidad de los procesos cognitivos de los estudiantes al resolver problemas computacionales y los resultados de la encuesta aplicada sobre atributos de calidad, se modeló *Diagramar*, adecuado para el aprendizaje de conceptos y técnicas asociados al diseño de algoritmos que considerara las necesidades manifestadas por los alumnos-usuarios.

El docente-investigador usa este software con el objetivo de indagar más profundamente el proceso de aprendizaje de sus alumnos. La aplicación permite pausar la ejecución y el profesor aplica este recurso cuando identifica momentos de ruptura entre lo que el estudiante cree que el algoritmo realiza y lo que la aplicación procesa.

Breve descripción de la aplicación: Diagramar posee las siguientes herramientas.

- Prueba de Escritorio, con la que se ejecutan los Casos de Prueba,
- Casos de Prueba que ofrece una serie de recomendaciones para ejecutar la prueba, procurando ejercitar diversos caminos de ejecución y
- Documentación, que permite al usuario editar las anotaciones marginales de su trabajo de diagramación generando un documento que se vincula al gráfico del diagrama desarrollado.

Estas funcionalidades se incorporaron al software para dar respuesta a un requerimiento común a todos los estudiantes entrevistados; en sus propias palabras, *"cuando veo un problema parecido bien resuelto, lo uso para diagramar la solución del nuevo problema"*. Además, permite que el alumno se habitúe a la consulta y producción de documentación asociada al desarrollo de programas.

Desde el punto de vista de la aplicación, el usuario ve la construcción de un nuevo algoritmo como una tarea muy sencilla ya que se realiza a través de la selección del elemento a insertar y de la zona del diagrama en la que debe ser ubicado. Cada objeto insertado admite la configuración de sus propiedades a través del panel ubicado en la zona derecha de la pantalla. Los conceptos de modularidad y reutilización de código se trabajan a través de

una galería de componentes estándar para ser utilizados por el usuario. Los componentes se dividen conceptualmente en grupos según el tipo de variables que utilizan: simples, indizadas unidimensionales o vectores y bidimensionales o matrices.

El aspecto central de la aplicación reside en la posibilidad que brinda para la depuración del algoritmo, efectuando la ejecución con diferentes velocidades de inspección de las instrucciones, una ejecución paso a paso, la visualización de la prueba de escritorio, la documentación de los casos de prueba, la documentación de la justificación del diseño y el almacenamiento del algoritmo diseñado. También permite su exportación como archivo de imagen bajo diferentes formatos, que facilita la socialización de las producciones a través del Aula Virtual de la cátedra. La experiencia indica que los debates relativos a temas clásicos de la algoritmia se enriquecieron a partir de la posibilidad de compartir en el entorno virtual los diagramas desarrollados por estudiantes y docentes. La comunidad somete a consideración sus producciones y aporta comentarios respecto a las realizaciones de los otros, en una dinámica en la que los archivos ejecutables de la aplicación y los archivos gráficos con los diagramas, son recursos imprescindibles para el planteo de los debates.

Metodología para la creación de algoritmos computacionales

La metodología se aplica a partir de la conceptualización y definición de aquello que se entiende como problema computacional. Entendemos que, la resolución de problemas computacionales implica el descubrimiento de un algoritmo. Juan José Cueto en su libro Método para la solución de problemas utilizando POO indica que “un algoritmo es un método para la solución de un problema” … “por lo tanto debemos entender la importancia de estudiar el método de solución “algoritmo” para comprender cómo se está solucionando el problema.” En una primera aproximación al concepto de algoritmos se puede decir que un algoritmo es “un conjunto de instrucciones para ejecutar una tarea” (J. Glenn Brookshear). El mismo autor afirma que “una vez descubierto un algoritmo para efectuar una tarea, la realización de ésta ya no requiere entender los principios en que se basa dicho algoritmo, pues el proceso se reduce a seguir las instrucciones”. Entonces, la resolución de un problema implica el desarrollo de un esfuerzo intelectual que produce un algoritmo, el cual, al ser aplicado, genera un resultado, es decir, una salida en función de una determinada entrada. Sobre la base de este sustento teórico, la metodología que proponemos para resolver un problema y obtener así una especificación algorítmica probada, consta de las siguientes fases:

FASE I: Comprender el problema
Enfatizamos entre los estudiantes la necesidad de trabajar a partir de una formulación del problema clara, concisa, precisa, que evite todo tipo de ambigüedades. Cuanto mejor esté formulado un problema mayor será la facilidad para encarar su solución. Se propone trabajar esta fase en tres subfases.

Fase I.1) Reconocer términos: es posible que en la formulación del problema existan palabras cuyo significado no sea conocido o interpretado en el contexto de la consigna. Este desconocimiento opera en los estudiantes como obstáculo para la cabal comprensión del problema y por tanto, lo desorienta hacia los posibles caminos de resolución. Es imprescindible despejar toda duda respecto de los términos desconocidos, ya sea a través de una investigación bibliográfica o recurriendo al redactor del problema para refinar los aspectos no interpretados. Se ilustra con un ejemplo de cada caso.
Problema A: “Dada una cierta cantidad de datos correspondientes a notas de estudiantes, de desea determinar el promedio y la moda de los mismos”. En este caso, habría que asegurarse de comprender el significado y proceso de cómputo del *promedio* y la *moda*. No tener certeza acerca de estos términos

obliga a recoger información a través de alguna fuente, por ejemplo, un libro de estadísticas.
Problema B: "Dados los datos de facturación correspondientes a un año, en una cadena de heladerías, se desea determinar los montos mensuales y estacionales de facturación, distinguidos por cada local de la cadena". La formulación del problema es clara para quien lo redactó, posiblemente el gerente de la cadena de heladerías. Sin embargo, el analista puede no descifrar la expresión *estacional*. Una posible interpretación es considerar las estaciones primavera, verano, otoño e invierno; pero también es posible pensar en otra separación lógica propia del rubro del negocio, digamos por ejemplo, diciembre-marzo, abril-agosto y septiembre-noviembre. Lo conveniente, en estos casos de incertidumbre, es consultar una fuente fidedigna directamente asociada con el destinatario de la solución computacional que se construya.

La metodología de la cátedra al respecto de la fase I.1 consiste en habilitar nuevas entradas al diccionario que provee MOODLE, para circunstancias como la que ilustra el problema A, en el que una mínima investigación bibliográfica resuelve la situación; o postear en el foro de Asistencia Temática lo que el docente, en calidad de "cliente" precisa respecto de los términos no reconocidos del problema B. Todo foro de asistencia académica está configurado "por grupos separados", lo que permite a diferentes docentes adoptar libremente su criterio al momento de evacuar las dudas sobre los términos en discusión. Un profesor cuyo grupo está avanzado al momento de resolver problemas como los del nivel B, puede complejizar la situación problemática solicitando la entrada de datos de inicio y fin de cada período estacional.

Fase I.2) Identificar Entradas, Salidas y Condiciones: en esta fase, se trabaja sobre el reconocimiento de la E/S y las condiciones sobre esos datos como restricciones que sobre ellos operan. También es importante aclarar que la formulación del problema no siempre tiene definidos los identificadores (denominaciones elegidas para los datos) de la E/S. Por ello, una de las primeras tareas de esta fase, además de identificar la E/S, es asignar un identificador válido para cada dato. Si por ejemplo, se requiere de un dato edad, un posible identificador válido es justamente "edad", "E", o cualquier otra denominación que permita comprender la naturaleza del valor almacenado bajo dicho identificador. Desde el punto de vista pedagógico, se orienta hacia la elección criteriosa de identificadores y a una inmediata documentación de respaldo.

Fase I.3) Diseñar casos de prueba: un caso de prueba está constituido por una colección de datos de entrada y las condiciones o restricciones que sobre ellos operan, necesarios para obtener un resultado válido como salida. Una salida es correcta si el resultado es el esperado para los datos provistos como entrada. Esta es una importante fase dentro del análisis porque contribuye a que el alumno, mediante situaciones concretas de manipulación de datos, los opere reflexivamente a través de un proceso que constituye la primera aproximación algorítmica de la solución.
Por ejemplo, retomando el Problema A relativo al cálculo del promedio y moda de un conjunto de notas de estudiantes, un caso de prueba sería:
Entrada:
Se poseen 10 notas (condición para cada dato nota: número natural entre 1 y 10). Las notas son 5, 6, 2, 8, 3, 9, 6, 1, 10, 6.
Salida: Promedio = 5,6 ; Moda = 6

La función principal del docente durante esta fase es la orientación para que el estudiante reconozca otro u otros casos de prueba que ejerciten posibilidades diferentes. Por ejemplo:
Entrada:
Se poseen 10 notas (condición para cada dato nota: número natural entre 1 y 10). Las notas son 5, 6, 5, 8, 3, 9, 6, 5, 10, 6.

Salida: Promedio = 6,3 ; Moda = 5 y Moda = 6 (existe más de una moda).

Al inicio de la asignatura el estudiante desarrolla un primer trabajo práctico orientado a la interpretación de consignas de diferente complejidad, pero sólo a los efectos de cumplir con la Fase I. Los casos de prueba se dejan en reserva para retomarlos luego de concretar las correspondientes especificaciones algorítmicas que se ejecutan sobre la aplicación *Diagramar*.

FASE II: Seleccionar componentes
Esta fase se corresponde con la de *concebir un plan*. Para el diseño de esta estrategia es necesario conocer todos los aspectos vinculados al problema, en especial, aquellos recursos que pueden combinarse y contribuir a la solución del mismo. Estos recursos son los que se denominaron Componentes. Poseer una galería de componentes facilita la selección y adaptación de los mismos con el fin de ensamblarlos coordinadamente para construir la solución. En este contexto metodológico, definimos **Componente** como un proceso elemental realizado por un autómata. Su característica principal es la de poseer una única funcionalidad, claramente definida.

FASE III: Diseñar el algoritmo
Los sistemas informáticos deben contar con la capacidad de cumplir tres tareas básicas: entrada (captación de datos), procesamiento y salida (transmisión de los resultados). Una primera idea sobre lo que es un algoritmo, es justamente ese conjunto de tareas ensambladas correctamente para que, a partir de una cierta entrada se pueda brindar la salida que resuelva el problema formulado.

FASE IV: Ejecutar pruebas de escritorio
Esta fase es crucial para lograr un desarrollo autónomo del estudiante que diseña sus primeros algoritmos. Por ello, se define en primera instancia cuáles son los objetivos de la prueba:
La prueba es un proceso de control con la intención de descubrir un error.
Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
Una prueba tiene éxito si descubre un error no detectado hasta entonces.
Cualquier producto de ingeniería (y de muchos otros campos) puede ser probado de una de dos formas: 1) conociendo la función específica para la que fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa; 2) conociendo el funcionamiento del producto, se pueden desarrollar pruebas que aseguren que "todas las piezas encajan"; o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada. Como es sabido, el primer enfoque se denomina *prueba de la caja negra* y el segundo *prueba de la caja blanca*. La orientación docente y las prestaciones de *Diagramar* están dirigidas a que la fase de prueba se desarrolle usando el enfoque de caja blanca.

Aplicación de la metodología a un caso práctico: la conversión de Sistemas de Numeración

En la unidad 2 del programa se presentan, entre otros, dos componentes que se aplican a la solución de diversos problemas algorítmicos que usan tipos de datos simples: Separación_Dígitos_Entero y Composición_Número_Entero. En la unidad 6 se presentan los sistemas de numeración binario, octal y hexadecimal y los métodos de conversión entre ellos. Para este trabajo interesa desarrollar sucintamente el Método de Suma Ponderada de potencias de la base y el Método de la División Reiterada.

Conversión de sistemas de numeración: desde que se inicia la unidad 6, el tema de operaciones en una misma base se presenta haciendo hincapié en los algoritmos asociados a dichos contenidos y se propone a los alumnos que luego de algunas prácticas de operaciones usando números concretos,

formulen los correspondientes algoritmos que permitan sumar, obtener complementos y multiplicar en una base menor o igual a diez.

En lo que se refiere a conversión de sistemas también se mantiene y profundiza el mismo enfoque. Se presenta el tema estableciendo el concepto de base *b* y el de conjunto de caracteres, *C*, que contiene los dígitos que, individualmente o en combinaciones entre ellos, representan todo número del sistema. Se establece que la conversión se puede realizar de un sistema de cualquier base a otro de cualquier base, siendo la primera la base de partida y la segunda la base de llegada.

Se considera entonces a *C* como el conjunto de caracteres del sistema de numeración de partida, cuya base es *b* (*b* >1), y a $a_i \in C$ los elementos del conjunto *C* o dígitos de dicho sistema. Se establece además que *C*' es el conjunto de caracteres del sistema de numeración de llegada, cuya base es *b*' y que a'_j los elementos del conjunto *C*'.

Método de la Suma Ponderada: se presenta el número a convertir como una suma ponderada de potencias de la base *b*

$N_b = (a_n a_{n-1} \dots a_1 a_0)_b =$
$= (a_n b^n + a_{n-1} b^{n-1} + \dots + a_1 b^1 + a_0 b^0)_b$
Y se establece la equivalencia de cada dígito de base *b* en un número de la base *b*`

$(a_0)_b \equiv (g_0)_{b'}$; $(a_1)_b \equiv (g_1)_{b'}$;
. . . . . ; $(a_n)_b \equiv (g_n)_{b'}$
Como así también la equivalencia de la propia base del sistema de partida b , en el sistema de numeración de base b '. Teniendo en cuenta que b se expresa en base 10, resulta:

$(b)_{10} \equiv (b^*)_{b'}$
Entonces, se hace notar a los alumnos que el número convertido se obtiene a partir de una suma ponderada:

$N_b = (a_n b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \dots + a_1 b^1 + a_0 b^0)_B \equiv$

$\equiv (\gamma_n (b^*)^n + \gamma_{n-1} (b^*)^{n-1} + \gamma_{n-2} (b^*)^{n-2} + \dots + \gamma_1 (b^*)^1 + \gamma_0 (b^*)^0)_{b'}$

Este proceso se puede ver algorítmicamente estableciendo la conversión de cada uno de los dígitos, pudiendo contar para ello con una tabla de conversión y contando también con la conversión de la propia base, de esta forma, el proceso culmina con una sumatoria ponderada.

Los alumnos que ejercitan este proceso notan que, si bien el método es de aplicación general, resulta sumamente valioso para conversiones donde la base de llegada es la base 10, a fin de no tener que diagramar procesos que realicen un álgebra desconocida por la computadora.

Método de la División Reiterada: la presentación de este método se basa en mostrar cómo en sistema decimal, la división entera de un número por 10 genera como resultado, en el cociente los dígitos mas significativos y en el resto el dígito correspondiente a las unidades del número dado. Se justifica el método analizando cada resto de división donde el cociente es la base de llegada expresada en sistema de partida y comprobando que cada resto resulta menor que dicho cociente, por lo tanto, es el equivalente a un dígito en la base de llegada.

Nuevamente, se presenta a los alumnos este proceso en forma algorítmica y se discute respecto a la conveniencia de aplicar este método en los casos en que la aritmética sea la decimal, es decir, en los casos en que la base de partida sea *b*=10.

Reutilización de Componentes en el caso práctico: una vez que son presentados ambos métodos de conversión, se trabaja la conversión de sistemas desde el enfoque algorítmico guiado por la siguiente secuencia:

1. Identificación de los componentes asociados al proceso de conversión de números enteros: *"Separación Dígitos Entero"* y *"Composición Número Entero"*.

2. Adecuación de los mismos para su reutilización en el contexto de las situaciones problemáticas relacionadas con la conversión de sistemas.
3. Construcción de su propio componente de conversión de números enteros.
4. Validación del componente a través de la herramienta de prueba y depuración que ofrece *Diagramar*.
5. Socialización del componente entre pares y docentes a través del entorno virtual.
6. Reutilización de su componente en la transferencia propia de los problemas clásicos de la conversión de sistemas.

Se ilustra la metodología descripta con un par de figuras: para el paso 1, la figura 1 que muestra parcialmente el componente *"Separación Dígitos Entero"* y la figura 2 que muestra los pasos 3 y 4 correspondientes al siguiente problema de conversión: *"elaborar un diagrama que permita convertir un número entero X expresado en sistema decimal, a su equivalente en base B (1<B<10) o viceversa, es decir, convertir a sistema decimal el número entero X)$_B$*".

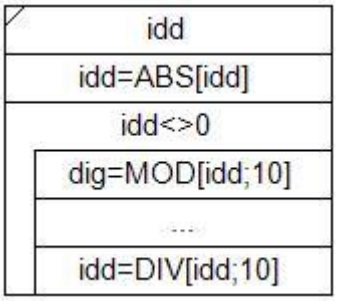

Figura 1: Componente Separación Dígitos Entero

Donde:

idd es el identificador de la variable que contiene el número entero cuyos dígitos se desea separar.

dig es el identificador de la variable que contiene cada dígito del número entero **idd**, obtenido por aplicación de la función MOD.

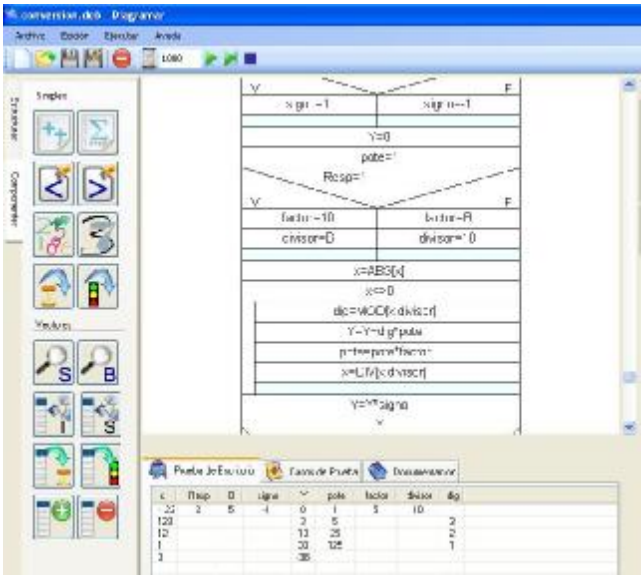

Figura 2: Reutilización de componentes

A modo de conclusión

Como puede apreciarse, el enfoque algorítmico se encuentra transversalmente asociado a otros temas que aborda la asignatura, procurando potenciar las competencias de abstracción en diversas situaciones problemáticas propias de la Computación y la reutilización y adecuación de componentes para un diseño algorítmico que paulatinamente debe incorporar calidad en las producciones de los alumnos iniciales de Programación.

Por otra parte, este enfoque propicia el uso de las herramientas diseñadas a medida de nuestros estudiantes, tanto la aplicación *Diagramar* como la publicación de las producciones personales a través del entorno virtual de la cátedra. De esta forma, al finalizar el cursado en el que el contenido central es el diseño algorítmico, los estudiantes poseen, además de la galería de componentes suministrada por la aplicación, una enriquecida colección de algoritmos probados y documentados que se retoman en la materia correlativa para su programación.

Bibliografía

De Giusti, A. (2001) *Algoritmos, datos y programas*. Ed. Pearson Education S.A.

del Olmo, P., Vargas, C., Mac Gaul, M. (2010) *Software que permite crear software: una experiencia de intervención didáctica con alumnos de primer año*. En actas de las Segundas Jornadas de Ingreso y Permanencia en Carreras Científico-Tecnológicas - IPECyT.

Mac Gaul, M. Aballay, P., Zamora, R. Soria, M. (2009) *Interpreter Chart Diagram N-S.* En actas del XV Congreso Argentino de Ciencias de la Computación – CACIC.

Mac Gaul, M., del Olmo, P., López, M., Fernández, E., Massé Palermo, M.L., Vargas, C. (2010) *Repensando estrategias didácticas en nuevos escenarios educativos*. En actas del Congreso Iberoamericano de Educación – METAS 2021.

Presuman, R. (2005) *Ingeniería del Software* 6ta. edición. Ed. Mcgraw-hill.

Wirth, N. (1986) *Introducción a la Programación Sistemática*. 2da. edición. Ed. El Ateneo.

Ginzburg, M. C. (1985) *Técnicas Digitales con Circuitos Integrados*. 2da. edición. Ed. Edigraf.