Rol y Organización del Área Programación en una Carrera de Ingeniería en Sistemas de Información

Castellaro, Marta[1]
Agüería, Stella[2]
Leone, Horacio[3]

Área Programación[4]
Departamento Sistemas
Facultad Regional Santa Fe
Universidad Tecnológica Nacional

Resumen

El este trabajo se presenta la inserción curricular de la disciplina PROGRAMACIÓN en el nuevo plan de estudios de la carrera Ingeniería en Sistemas de Información (Universidad Tecnológica Nacional) y su implementación en la Facultad Regional Santa Fe.

Sintéticamente se expone en que forma el *Área Programación*, como se la concibe en esta propuesta, participa en las distintas dimensiones de la formación del Ingeniero en Sistemas de Información a lo largo de toda la curricula.

Finalmente se analizan las experiencias que se están obteniendo a partir de la implementación de plan mencionado, que si bien no permiten obtener conclusiones finales, son útiles para realizar el seguimiento del proyecto.

[1] Profesor Titular - Dpto. Sistemas (Fac. Reg. Santa Fe, UTN) - E-Mail: mcastell@arcride.edu.ar
[2] Profesor Asociado - Dpto. Sistemas (Fac. Reg. Santa Fe, UTN) - E-Mail: sagueria@arcride.edu.ar
[3] Profesor Asociado - Dpto. Sistemas (Fac. Reg. Santa Fe, UTN)
INGAR (Fund. ARCIEN, CONICET) - E-Mail: hleone@arcride.edu.ar
[4] UTN -FRSF. Lavaise 610, (3000) Santa Fe, Argentina
Tel-Fax : (042) 690348 / 608979

Rol y Organización del Área Programación en una Carrera de Ingeniería en Sistemas de Información

INTRODUCCIÓN

Las primeros curriculums de carreras de Sistemas de Información (años 80), en general, se conformaron considerando a la Programación como una disciplina central, en la que se podían observar tres etapas:

- Una correspondiente al nivel inicial de la formación, con carácter predominantemente práctico (Resolución de problemas, Lenguajes de Programación, Manejo de archivos, Administradores de Bases de Datos), donde aparecían lenguajes de tipo administrativos en sus máximos detalles.
- Una segunda etapa, tomada de Ciencias de la Computación, donde se desarrollaban los fundamentos teóricos de la disciplina (Autómatas, Gramáticas Formales, Computabilidad, etc.).
- Una etapa final, donde se incluían materias abocadas a la enseñanza de lenguajes de última aparición, incluyendo los ejes del paradigma en que se sustentaban y las metodologías de diseño de software correspondientes.

De esta manera, un estudiante evolucionaba en esta disciplina, con situaciones como las siguientes: Comenzaba resolviendo problemas en forma estructurada con lenguaje Pascal, luego aprendía en detalle cuestiones tales como generación de reportes en Cobol y a continuación se abocaba a comandos de SQL. En el ciclo superior estudiaba los fundamentos de un lenguaje, constructores, generadores y por último (generalmente en alguna materia optativa o de especialización) hacía un curso de C++, incluyendo el conocimiento de la programación orientada a objetos.

Esta formación, además de aparecer desordenada, reforzaba el método *"primero aprender a utilizar y luego, como especialización, comprender los fundamentos"*. Además, los nuevos paradigmas, que deben ser las herramientas más importantes del futuro profesional, se presentaban como una *información* final, luego de varios años de razonamiento y ejercitación en otros métodos y modelos conceptuales, no presentados explícitamente. En este contexto se

realizaba la parte formativa de programación en la etapa final de la carrera, en lugar de hacerlo durante el comienzo de la misma.

PARADIGMA DE ENSEÑANZA

Entendiendo que este modelo curricular debía reformarse, se trabajó en busca de una organización diferente. Los bases conceptuales y criterios tenidos en cuenta fueron los siguientes:

a) Principios generales:
Debido a que la informática es simultáneamente una disciplina matemática, científica e ingenieril, el ejercicio profesional en las distintas áreas emplea diferentes metodologías de trabajo, para realizar desarrollo, investigación y trabajos de aplicación.

Para la definición de un diseño curricular, se deben considerar **las áreas temáticas y los procesos** que caracterizan las distintas metodologías de trabajo.

Así, se consideraron las siguiente áreas constitutivas [1]:

Algoritmos y Estructuras de Datos
Arquitectura
Inteligencia Artificial y Robótica
Bases de Datos y manejo de información
Comunicación hombre-máquina
Computación numérica y simbólica
Sistemas Operativos
Lenguajes de Programación
Metodologías e Ingeniería de Software

Cada área del curriculum, debía contemplar cada uno de tres procesos principales :

Un significante teórico {st} (definiciones, axiomas, principios matemáticos, interpretación de resultados), para desarrollar y comprender los principales fundamentos matemáticos que se aplican a la disciplina informática.

Un significante de abstracciones {sa} el que se origina en el desarrollo de las ciencias experimentales (formulación de hipótesis, modelización, diseño de experimentos, análisis de resultados).

Un significante de diseño e implementación {sdi} (requerimientos, especificaciones, desarrollo de sistemas o dispositivos para resolver problemas).

La **teoría** en la disciplina puede encontrarse en un curso introductorio donde se desarrollen los principios matemáticos que se aplican a la misma, pero también debe emerger en los estudios de aspectos específicos del área. La **abstracción** puede introducirse por diversos caminos, en clases y laboratorios, en trabajos grupales extra-aula. El proceso de **diseño e implementación** puede lograrse por experiencias directas o por el estudio de diseños de otros (casos) y en proyectos de laboratorio (con las restricciones de contexto del mundo real).

En cada área es además necesario considerar el **contexto social y profesional** inherente (bases culturales, sociales, legales, éticas). Deben tenerse en cuenta los roles individuales en los procesos, apreciar las cuestiones filosóficas, problemas técnicos, y valores estéticos que juegan un papel importante en el desarrollo de esta disciplina. Es importante generar capacidades para poder anticipar el impacto de introducir un determinado producto o entorno de trabajo, analizar consecuencias individuales, grupales e institucionales, así como las implicancias legales.

b) Contexto específico:

En este marco curricular, se trató de definir el **rol** de la *Programación.*

Se entiende, en una primera aproximación, que el término *Programación* se refiere al conjunto de actividades que involucran la descripción, desarrollo y efectiva implementación

de soluciones algorítmicas para problemas bien especificados, incluyendo además herramientas de estilo en lenguajes de programación. A partir de esta definición, la *Programación* se presenta en las nueve áreas antes citadas y, por lo tanto, su rol es multidimensional.

Los estudiantes desarrollan programas durante el diseño de software, ejercitan y modifican programas durante los trabajos de laboratorio, leen programas en el material de estudio de los cursos, textos y publicaciones, y utilizan programas para presentar sus resultados. En un sentido más amplio, la *Programación* es una **herramienta básica de comunicación** que estudiantes y profesionales emplean en forma cotidiana, no sólo para interactuar con computadoras, sino también con sus colegas. Esta dimensión referente a la comunicación es la que da significado a la adquisición de un estilo de programación, y es una de las que motiva la reformulación del *Área Programación* en el contexto de la carrera de Ingeniería en Sistemas de Información. En general, el ingeniero no será un programador, sino que se comunicará pensando en términos de *programación.*

PROPUESTA CURRICULAR

El resultado de la primer etapa de este trabajo condujo a la formalización de una propuesta curricular, cuyos principales ejes son:

- **Articular** un ***Área Programación*** *(AP),* como *Ciencia Tecnológica Básica* [2], obligatoria que representa el 13 % del total del plan:
  - ⇒ *Primer nivel.* Introducir los conceptos matemáticos básicos para la disciplina mediante un curso de ***Matemática Discreta*** (*MAD*) (lógica proposicional, relaciones y funciones, estructuras matemáticas) {**st**}.
  - ⇒ *Primer nivel.* Iniciar el estudio de ***Algoritmos y Estructuras de Datos*** (*AED*) en forma conjunta, empleando una estrategia funcional y imperativa. {**st - sa**}
  - ⇒ *Segundo nivel.* Presentar y analizar los conceptos y estructuras básicas de ***Sintaxis y Semántica de Lenguajes de Programación*** (*SSL*) {**st - sa**}.

⇒ *Segundo nivel.* Se tratan tres paradigmas básicos de programación (*PPR*), con fundamentos y prácticas en laboratorios {**sa** - **sd**}.

- **Complementar** los contenidos del *AP* en el ciclo básico, incluyendo como electivas, ***Talleres de Programación*** donde se enfatice la resolución de problemas y la implementación de programas en entornos de trabajo específicos.

- **Integrar** al *AP* con asignaturas paralelas pertenecientes a un ***Área Troncal***, denominada *Área Integradora*, que se desarrollan en forma anual en los cinco años de la carrera. En estas materias se incluyen trabajos grupales sobre casos de estudio y problemas reales en Sistemas de Información:

  ⇒ En el *Segundo nivel*, se desarrolla ***Análisis de Sistemas***, junto a *SSL* y *PPR* , de manera que se vinculen datos, procesos y métodos de descripción con los temas de datos abstractos, estructuras de programas, y que las herramientas de análisis ya se encuadren en el contexto de los distintos paradigmas desarrollados en el ***Área Programación***.

  ⇒ En el *Tercer nivel*, se desarrolla ***Gestión de Datos*** (*GDA*), junto a la integradora ***Diseño de Sistemas***, de manera que se complementen los modelos y metodologías con herramientas de implementación, en el contexto de un paradigma. En el 2do. cuatrimestre, se desarrolla ***Redes de Información*** junto al curso electivo ***Taller de Programación***, lo que permite enriquecer los aprendizajes de construcción de programas, con los entornos de desarrollo.

- **Aplicar** la programación en asignaturas del 4to. y 5to. nivel correspondientes a *Ciencias Tecnológicas Aplicadas* [2] (Investigación Operativa, Simulación, Inteligencia Artificial, Teoría de Control), con metodologías y lenguajes del dominio de discurso de las mismas.

- **Especializar** la formación en el *AP* mediante la oferta de materias electivas en el 4to. y 5to. nivel. Por ejemplo: Programación Concurrente y Distribuida.

- **Sintetizar** los conocimientos adquiridos en el *AP*, en el último año, en la realización de un ***Proyecto Final***, que consiste en un desarrollo o aplicación de envergadura, donde se pongan en juego las distintas disciplinas tratadas durante la carrera. En esta etapa de la formación tienen un rol importante los conocimientos adquiridos en el ***Área de Programación***, para ser utilizados en la fundamentación de la planificación y ejecución del proyecto. Además se deben considerar aquellos aspectos de la *programación* vinculados con los recursos humanos intervinientes en el proyecto y contextos institucionales.

IMPLEMENTACIÓN DE LA PROPUESTA

Se esta desarrollando el segundo nivel de la propuesta. Los objetivos y contenidos de las asignaturas básicas del *AP*, correspondientes a los niveles que se están dictando son:

AED

Objetivos:

Introducir la disciplina Programación en lo que respecta a[3], [4], [5]:

- El planteamiento de problemas a resolver.
- El estudio de algoritmos computacionales: Técnicas y esquemas de diseño; corrección y eficiencia.
- La organización de los programas: paradigmas, lenguajes, desarrollos y estilos.
- Las estructuras de datos, como una manera conceptual de organizar los datos y herramientas para su implementación.

Contenidos

- Introducción a la programación y a la especificación de algoritmos.
- Especificación funcional y Especificación imperativa de algoritmos.
- Corrección y complejidad de algoritmos.
- Taller de lenguajes: uno funcional (Mulisp) y otro imperativo (Pascal).

SSL

Objetivos:

Introducir en el análisis y evaluación de los conceptos más importantes de los lenguajes de programación (LP) como herramienta para la producción de programas [8], [9], [10], [11].

Contenidos:

- Introducción al Diseño y Desarrollo de Software. Objetivo en el diseño de los lenguajes (Fiabilidad, Mantenibilidad, Eficiencia).
- Evolución de conceptos en los lenguajes de programación: Abstracción de datos y Abstracción de control.
- Sintaxis y Semántica de los Lenguajes de programación.
- Tipos de datos.
- Estructuras de control.
- Introducción a la Semántica Formal.
- Taller de análisis comparativo de lenguajes y construcción de programas elementales: ADA, Pascal, Modula2, etc.

PPR

Objetivos:

Introducir al alumno en el concepto de paradigmas de programación y la presentación de los principales paradigmas (además del imperativo): funcional, lógico y orientado a objetos. Se estudiará cada paradigma a partir de la revisión del modelo que define al mismo, es decir cuales son los conceptos primitivos con que se cuenta en el mismo para construir un programa y en que forma interaccionan en la ejecución del programa [12], [13], [14], [15], [16], [17], [18].

Contenidos:

- Introducción a los paradigmas (ventajas y limitaciones).
- Paradigma de programación lógica. Conceptos básicos del cálculo con Cláusulas de Horn. Resolución. Prolog.

- Paradigma Funcional. Cálculo Lambda. Scheme.
- Paradigma orientado a objetos. Conceptos básicos. Smalltalk.

CONCLUSIONES

La modificación del *AP*, en lo referente a esta redefinición y reorganización de contenidos, implicó necesariamente la adaptación del cuerpo docente del área, el cual se había formado con un enfoque estructurado e imperativo. Uno de los primeros resultados positivos del cambio propuesto, fue la asimilación de los docentes de esta percepción del *AP* , admitiendo las múltiples dimensiones en que puede ser utilizada la *programación* por un Ingeniero en Sistemas de Información y la necesidad de la presentación de los diferentes paradigmas con que se asocia este enfoque.

Actualmente se está completando el 2do. año de la carrera. En la etapa en que se encuentra la implementación del proyecto se observa que el alumnado se encuentra motivado con el mismo, y que los rendimientos académicos son satisfactorios, medidos en términos de porcentajes de regularización, promoción, proyectos grupales ejecutados, etc. Si bien estos no son parámetros de calidad, son indicadores que permiten realizar el seguimiento del proyecto. No se podrán evaluar los resultados reales hasta que la implementación del nuevo enfoque alcance las etapas de **Aplicación** y **Síntesis**.

BIBLIOGRAFIA

[1] ACM/IEEE-CS, Joint Curriculum Task Force. "Computing Curricula 1991". ACM Press,1991

[2] ICI - CONFEDI, "V Taller sobre Unificación Curricular en la Enseñanaza de la Ingeniería. Conclusiones". Tucumán, 1996.

[3] Abelson H., G. Sussman, J. Sussman, "Structure and Interpretation of Computer Programs". MIT Press (1984).

[4] Universidad Politécnica de Madrid, "Plan de Estudios de Ingeniería Informática". España. (1996).

[5] Señas P., García A., "Estructuras enlazadas: una aproximación independiente de la implementación". Tercer Ateneo de Profesores Universitarios de Computación, Bahía Blanca, (1995)

[6] Galve -Gonzalez y otros, "Algorítmica (Diseño y análisis de algoritmos funcionales e imperativos)". Ed. Ra-ma, Madrid (1993).

[7] Aho A., Hopcroft J., Ullman J., "Estructuras de Datos y Algoritmos". Addison-Wesley (1988).

[8] Ghezzi C., Jazayeri M., "Programming Language Concepts". Ed. Wiley & Sons (1987).

[9] PRATT T., "Lenguajes de Programación. Diseño e Implementación". Ed. Prentice Hall (1987).

[10] WATT D.A., "Programming Languages, Sintaxis and Semantics". Prentice Hall (1991).

[11] Bal H. D., Grune D., "Programming Language Essentials". Addison-Wesley (1994).

[12] Kowalski R. A.."Logic for Problem Solving". North Holland, Amsterdam, (1979).

[13] Sterling L. , E. Shapiro. "The art of Prolog: Advanced Programming Techniques". The MIT Press (1986).

[14] Tasistro A., J. Vidart, "Programación Lógica y funcional". EBAI (1988).

[15] MacLennan B. J.. "Functional Programming - Practice and Theory". Addison-Wesley (1990).

[16] Lalonde y J. Pugh."Inside Smalltalk. Vol. I y II". Prentice Hall International (1991).

[17] Strostrup. "What is Object-Oriented Programming?". IEEE Software, 10-19 (1988).

[18] BOOCH G., "Object-Oriented Design with Applications". The Benjamin/Cummings Publish, (1991).