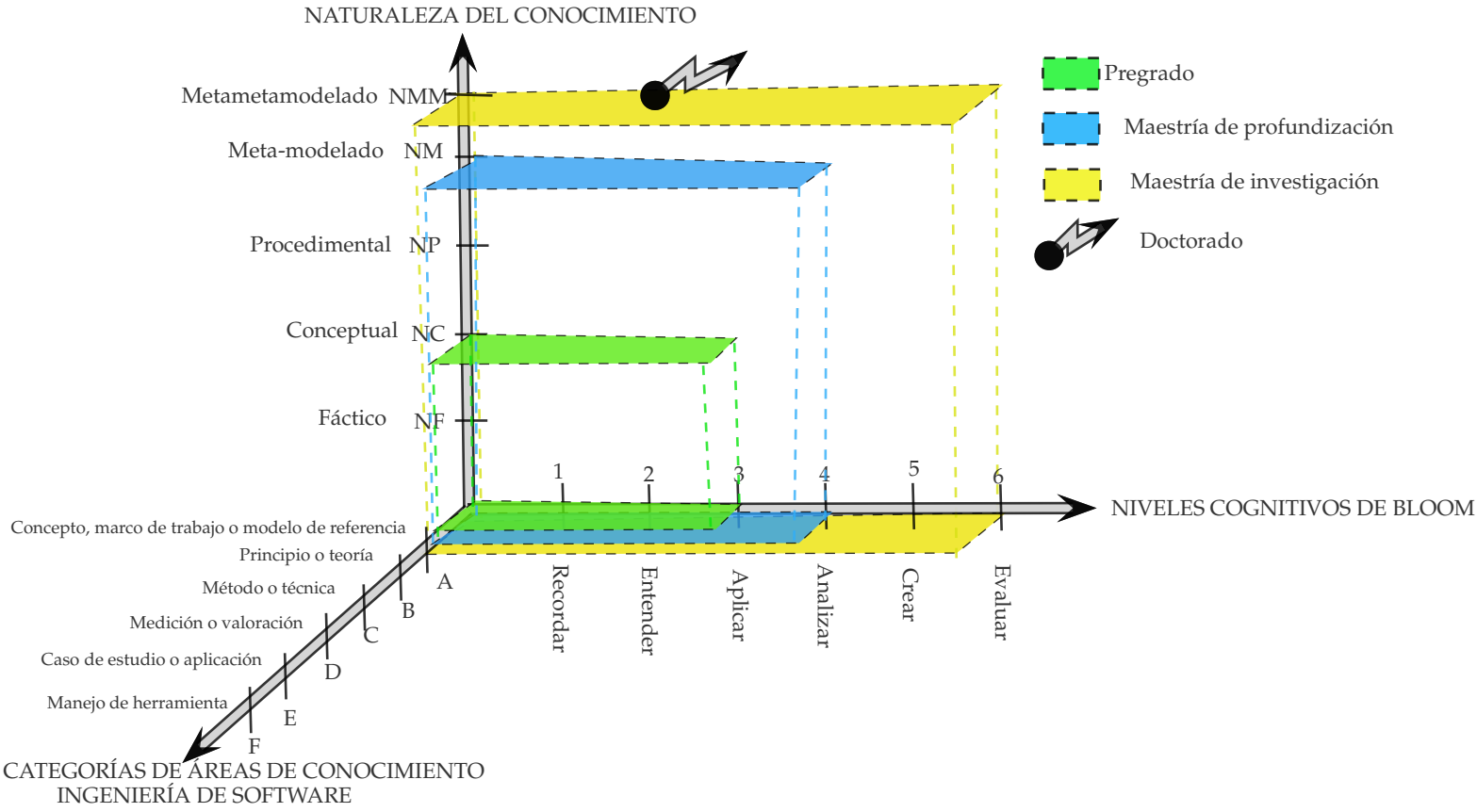
Universidad Distrital Francisco José de Caldas

Lineamientos para el diseño de programas de formación en Ingeniería de Software



Henry Alberto Diosa

Universidad Distrital Francisco José de Caldas
Grupo de Investigación
Arquitecturas de Software

Publicado por Universidad Distrital Francisco José de Caldas. Segunda impresión. Versión con correcciones editoriales de la primera impresión.

http://arquisoft.udistrital.edu.co

Autor: Henry Alberto Diosa.
Título: *Lineamientos para el diseño de programas de formación en Ingeniería de Software*
ISBN: 978-958-787-006-0 Primera edición, abril de 2018.
Corrección de estilo: Editorial UD
Coordinación editorial primera edición: Nathalie De La Cuadra N.
Montaje de cubierta de la primera edición: Astrid Prieto
Editor: Editorial UD - Henry Alberto Diosa
Diagramación y montaje de cubierta segunda impresión: Henry Alberto Diosa
© 202104
Primera impresiónr: Editorial UD
Segunda impresión: Henry Alberto Diosa
Segunda impresión, Abril de 2021

Contenido

Introducción

El otorgamiento de los registros calificados para niveles posgraduales en Colombia, doctorados y maestrías en sus diversas modalidades (Profundización e investigación), está bajo la responsabilidad del Consejo Nacional de Acreditación (CNA)[1]. En este proceso, previo al diligenciamiento de una serie de instrumentos documentales, se aplica una visita de pares académicos a la institución interesada en obtener la aprobación del registro para que sea autorizada la apertura de inscripciones y matrícula y se pueda iniciar su normal funcionamiento. Lamentablemente, estas visitas de pares académicos[2] no logran garantizar la real calidad y pertinencia de los programas avalados. Algunas de las razones que generan este hecho son:

1. Los pares académicos pueden avalar programas posgraduales genéricos con diferentes énfasis, de estos solo pueden ser conocedores parcialmente desde el punto de vista disciplinar; es decir, es imposible o realmente excepcional que un par académico posea la formación requerida para valorar diversos énfasis en maestrías y doctorados genéricos.

2. Lo anterior conlleva el riesgo de una superficial apreciación del contenido temático disciplinar propuesto por cada institución y a un riesgo inherente de aprobación de planes de estudio deficientes, frente al estado del arte mundial, en los énfasis que el par académico es débil.

3. Las universidades e instituciones universitarias ya son conocedoras de la instrumentalización necesaria para lograr registros calificados y desafortunadamente pueden orquestar una documentación (el papel puede con todo...) y procesos transitorios para reflejar una madurez y un ejercicio aparente de conceptualización de las disciplinas que en la cotidianeidad no es real.

Lo anterior impacta directamente en lo que Sergio de Zubiría ilustra claramente como la tensión entre calidad y pertinencia de la formación universitaria [1]. Esta coexistencia complementaria de la pertinencia y la calidad merece un corto análisis. La pertinencia, entendida como programas académicos en Ingeniería de Software que respondan a la formación de profesionales y posgraduados

[1] http://www.cna.gov.co

[2] En algunos casos solo realiza la visita un par académico

que solucionan los problemas del contexto colombiano y que tienen en cuenta el estado del arte nacional e internacional de la disciplina, se puede evidenciar con solo revisar los indicadores de crecimiento de la industria del software a nivel nacional [2],[3],[4] e internacional [5],[6].

La calidad es directamente proporcional a la idoneidad y compromiso de profesores y estudiantes soportados por una infraestructura suficiente a la labor de formación; lo anterior es parte integral del currículo. Los análisis y aportes sobre cómo garantizar la idoneidad y el compromiso de profesores no son parte del alcance de este documento; es responsabilidad de las universidades, en sus contextos particulares, la incorporación de profesores internos y externos que brinden un soporte idóneo al currículo que determina la formación de sus estudiantes posgraduales. En este sentido, el documento se concentra en aspectos del currículo sobre perfiles de referencia que deben cumplir los estudiantes admitidos a programas de formación en Ingeniería de Software y sobre el alcance del plan de estudios. Es relevante tener en cuenta que la infraestructura soporte a los procesos de formación se debe hacer consistente con las exigencias que imponen los elementos claves que se desarrollan en esta propuesta.

Teniendo en cuenta las premisas anteriores, se acepta que por necesidades de la industria de software y del desarrollo de la disciplina la pertinencia es obvia. Por lo anterior, se puede evidenciar una justificación facilista de los programas y énfasis posgraduales en Ingeniería de Software, sin contar con la conceptualización profunda de la disciplina ni contextualización coherente con el estado del arte internacional. En síntesis, se acepta la creación de programas de formación en esta disciplina más por pertinencia que por calidad en su formulación; lo precedente rompe con el sentido de una academia responsable con la disciplina y genera un desequilibrio formativo desde la misma formulación de los programas.

Por otra parte, la debilidad gubernamental en los procesos de aprobación de los registros calificados no debe ser excusa para que las universidades públicas y privadas no asuman la responsabilidad de apuntar positivamente hacia la pertinencia y alta calidad. Por esta razón, la propuesta busca establecer por lo menos un documento conceptual que formule los elementos esenciales de los currículos orientados a la formación pregradual y posgradual relacionados con la Ingeniería de Software y que se ubique de manera aceptable en el estado del arte actual de la disciplina con una clara contextualización a las necesidades de la industria del software en Colombia.

Es evidente, en los documentos de acreditación del énfasis en Ingeniería de Software de algunas universidades, la carencia de un sustento conceptual asociado a las propuestas de formación posgradual en la disciplina a nivel internacional, se someten solo a la instrumentalización documental que el CNA exige y una leve referenciación del contexto nacional. Esto ha sido exitoso en cuanto a cumplir y lograr la aprobación del énfasis. Este antecedente genera la responsabilidad interna en cada universidad o IES (Institución de Educación Superior) de establecer un marco conceptual del currículo en el énfasis en Ingeniería de Software que esté conforme al marco referencial de programas posgraduales en el orden internacional y que facilite la homologación de idoneidad de nuestros

graduados, junto con un análisis de pertinencia de la formación que se les imparte.

Estructura del documento

La Figura 1 presenta de manera resumida el esquema de desarrollo del presente documento.

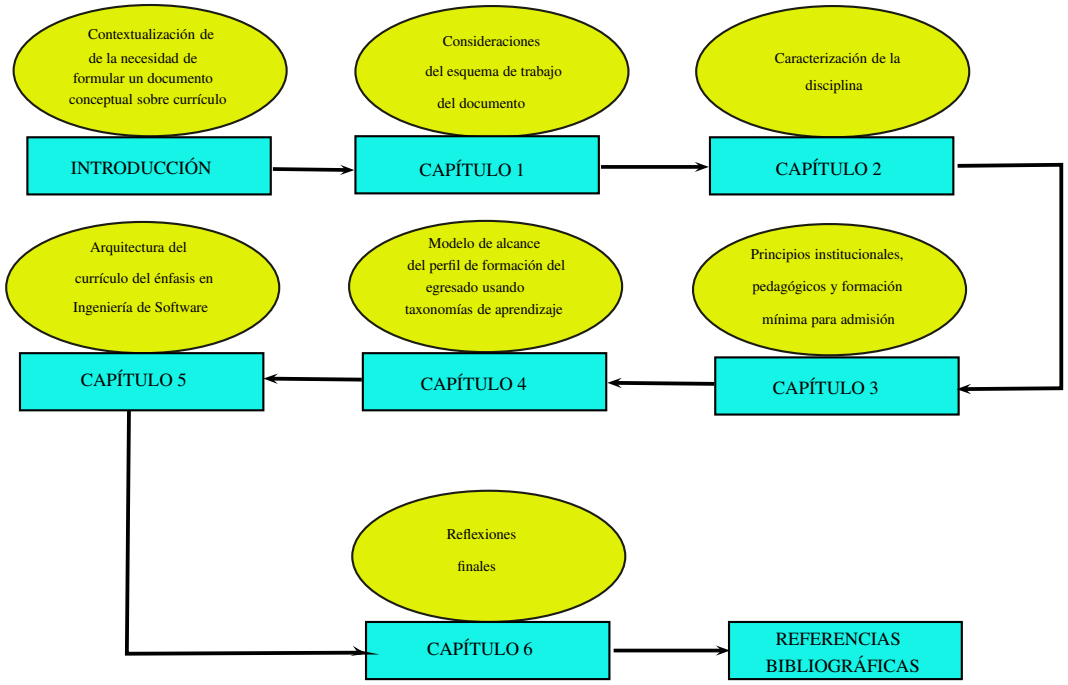


Figura 1. Resumen gráfico de la estructura del documento.

Capítulo 1

Consideraciones sobre el enfoque de este documento

Vivimos en la actualidad en un mundo que los hombres volvimos vertiginoso y con frecuencia la ausencia de profundidad analítica acompaña las decisiones; en consecuencia, la propensión al error es alta y muchas veces las consecuencias negativas son irreversibles.

1. El esquema de trabajo usado en el documento se inspira en su mayor parte en la propuesta de referencia de alcance internacional elaborada por el *Integrated Software and Systems Engineering Curriculum* (**iSSEc**) para currículos de programas de maestría [7] en Ingeniería de Software. Este marco de referencia se ha contextualizado a la naturaleza, las modalidades y los niveles de formación en Ingeniería de Software. Al final del documento se hace un análisis de un caso de estudio particular enfocado en la Maestría en Ciencias de la Información y las Comunicaciones (MCIC, en adelante) de la Universidad Distrital Francisco José de Caldas.

2. La propuesta del **iSSEc** no atiende las necesidades de formación en investigación y se intenta cubrir este vacío.

3. Dada la complejidad y extensión temática de la disciplina (ver áreas de conocimiento en la Tabla 2.1) se necesita que el programa académico cubra un mínimo común de competencias profesionales avanzadas, académicas y de investigación según la modalidad de formación. Esta propuesta trata de mantener una correlación 50/50 entre el currículo flexible y la base común de formación para las dos modalidades del énfasis.

4. Esta propuesta parte de la premisa de que la formación del nivel pregradual se debe armonizar con los objetivos del programa posgradual en lo que se refiere al énfasis en Ingeniería de Software.

5. Los elementos del currículo propuestos en este trabajo reconocen a la Ingeniería de Software como una disciplina que cuenta con un cuerpo de conocimiento bien establecido y con un sólido desarrollo en sus aspectos teóricos y prácticos.

6. Hay que tener en cuenta que la disciplina en cuestión adolece de una rápida evolución y dinámica de cambio; por consiguiente, debe existir

una revisión recurrente y periódica de las diversas áreas de formación del énfasis en Ingeniería de Software en sus dos modalidades. Es obvio que la modalidad de investigación, al atender los fundamentos esenciales, debe ser menos volátil que la modalidad profesionalizante. En estas revisiones es importante tener en cuenta tanto los avances y cambios de la disciplina en el ámbito local, como en el ámbito internacional.

7. Los cursos o espacios académicos deben actualizarse por lo menos cada dos años.

8. Es tal la complejidad de atender una adecuada formación, tanto por la extensión como por la profundidad en las diversas áreas de conocimiento (enunciadas en la Tabla 2.1), que se debe evitar la mala práctica de acomodar cursos que no tienen relación directa con las temáticas propias de la disciplina; es decir, cada espacio académico debe contar con una justificación clara en su relación directa con alguna de las áreas de conocimiento. La proliferación de espacios académicos no relacionados o lejanamente relacionados se puede evitar con la revisión exhaustiva por parte de un comité académico científico conformado por profesores de amplia experiencia y formación en Ingeniería de Software o las disciplinas relacionadas. La equivocación de acomodar cursos o espacios académicos no relacionados generará costos frente a la credibilidad del programa de formación.

9. Esta propuesta identifica las principales habilidades, destrezas y conocimiento que un profesional posgraduado debe poseer. Estas van desde los aspectos de formación técnica idónea hasta asuntos de formación ética y de buenas prácticas de comunicación.

10. Uno de los objetivos de este documento es diferenciar claramente el alcance de un programa de formación pregradual frente a uno de formación posgradual en Ingeniería de Software; lo anterior, teniendo en cuenta las dos modalidades: de profundización (llamada también profesionalizante) y la de investigación.

11. Los requisitos para ingresar a un programa de formación posgradual en Ingeniería de Software se identificarán explícitamente en este documento; estos requisitos se establecerán con base en dos premisas: el conocimiento esperado del aspirante y su experiencia en el ejercicio de la disciplina.

Capítulo 2

Sobre el alcance de la disciplina

Es la universidad, por naturaleza, un espacio de reflexión disciplinar profunda. Su alejamiento de este principio la puede hacer irrelevante para el círculo social que la sostiene y tolera.

La Ingeniería de Software es considerada una de las cinco disciplinas fuertemente relacionada con la computación y por esta razón es importante diferenciarla, por lo menos en su definición, de las otras cuatro [9][1]:

1. **Ingeniería del Computador.** Relacionada con el diseño y la construcción de computadores. Esta disciplina se encarga del estudio del hardware, software, comunicaciones y la interacción entre estos. Su origen se cimenta en la evolución de las teorías, principios y prácticas de las ingenierías eléctrica y electrónica junto con las matemáticas, aplicadas al diseño y la construcción de dispositivos basados en el concepto de computador. El énfasis de cualquier currículo en esta disciplina será más sobre el hardware que el software. Algunas de las áreas de trabajo en esta disciplina son: sistemas empotrados, desarrollo de dispositivos como teléfonos celulares, dispositivos de grabación y reproducción digitales, sistemas de sensores y alarmas, máquinas de rayos X, herramientas con control programado; en general, dispositivos con alguna capacidad computacional integrada al hardware. Para apoyar la formulación de un programa en Ingeniería del Computador se puede consultar a [10], allí se propone un marco de referencia curricular de orden internacional para la disciplina.

[1] Por otra parte, no se debe obviar que existe también una relación fuerte con otras disciplinas no relacionadas directamente con las ciencias de la computación y que dan soporte al ejercicio de la Ingeniería de Software en algunas de sus áreas de conocimiento; estas disciplinas son: 1) Administración general; 2) Matemáticas; 3) Gestión de proyectos; 4) Gestión de calidad; 5) Ingeniería de sistemas.

2. **Ciencias del Computador.** Esta disciplina tiene tres perspectivas de trabajo que la definen:

   a) *Diseño e implementación de software.* Frecuentemente, los científicos de la computación se hacen cargo de labores de programación y de la supervisión de estas labores desarrolladas por otros programadores. La meta principal es mantener una óptica amplia y profunda sobre el uso y aplicación de los modelos de computación y las diversas técnicas de programación.

   b) *Concepción de nuevos usos de los computadores.* Son ejemplos los avances en diversas áreas de conocimiento relacionadas con los computadores como: redes de comunicaciones de datos (avances en la WWW[2]), bases de datos (la ciencia de los datos [11]), interfaces humano-computador (programas para reconocimiento de voz o de otras acciones humanas) [12] e inteligencia computacional (algoritmos de minería de datos [13]), entre otros.

   c) *El desarrollo de soluciones efectivas para problemas de computación.* Científicos de la computación desarrollan mejores maneras de almacenar datos [14], enviar datos sobre redes de comunicaciones [15], desplegar imágenes complejas [16], procesamiento algorítmico y complejidad computacional [17][18], búsqueda de información [19], entre otras áreas de problemas computacionales.

   Mientras otras disciplinas preparan a sus graduados para trabajos específicos, los graduados en ciencias del computador (algunas veces denominada en nuestro medio como ciencias de la computación) pueden adaptarse con facilidad a nuevas tecnologías e ideas donde la computación está subyacente. Para apoyar la formulación de un programa en Ciencias del Computador se puede consultar en [20], allí se propone un marco de referencia curricular de orden internacional para la disciplina.

3. **Sistemas de información.** Los profesionales de esta disciplina trabajan primordialmente sobre la información que los sistemas de computador pueden manejar para el logro de objetivos empresariales y en el soporte a los procesos que una empresa puede implementar y mejorar usando tecnologías de la información. En este sentido, un programa académico en Sistemas de Información debe proveer a los profesionales graduados de capacidades para comprender factores técnicos y organizacionales conjugados con la habilidad de apoyar a las organizaciones en determinar cómo la información y los procesos de negocio soportados en tecnología pueden generar ventajas competitivas.

   Los profesionales en sistemas de información deben poseer una comprensión sólida de prácticas y principios organizacionales para servir de puente de comunicación efectiva entre las comunidades de TI[3] y el nivel de gestión

[2]World Wide Web.
[3]Tecnologías de Información.

o directivo; en consecuencia, los formados en esta disciplina se ven involucrados en el diseño de sistemas colaborativos y de comunicación organizacional soportada en tecnología.

Para apoyar la formulación de un programa en Sistemas de Información se puede consultar a [21]; allí se propone un marco de referencia curricular de orden internacional para la disciplina.

4. **Tecnologías de la Información.** Se puede visualizar esta disciplina desde una perspectiva complementaria a la de Sistemas de Información; si esta última enfatiza su enfoque de trabajo sobre la "información", la disciplina de Tecnologías de Información se concentra en la infraestructura de tecnologías de información de las organizaciones y en cómo las personas la usan. En el mundo actual las organizaciones requieren profesionales que brinden el soporte necesario para elegir, instalar, gestionar y mantener los sistemas de computador, redes de comunicaciones y software que soportan el trabajo cotidiano; por esta razón, se debe contar con un equipo de profesionales que resuelva cualquier problema con la infraestructura tecnológica. Adicionalmente, los formados en la disciplina deben apoyar los procesos de adquisición de productos de hardware y software. Para apoyar la formulación de un programa en Tecnologías de Información se puede consultar a [22], allí se propone un marco de referencia curricular de orden internacional para la disciplina.

5. **Ingeniería de Software.** La ingeniería de software se ha definido como:

   - "La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; esto es, la aplicación de ingeniería de software" [23].
   - "Ingeniería de software es la disciplina de desarrollar y mantener sistemas de software que se comportan de manera fiable y eficiente, son razonables para desarrollar y mantener..." [9].
   - "La aplicación sistemática de conocimiento tecnológico y científico, métodos, y experiencia para el diseño, implementación, pruebas, y documentación de software" [23].
   - "La fundamentación y uso de principios sólidos de ingeniería (métodos) para obtener en forma económica software que sea fiable y que funcione sobre máquinas reales de manera eficiente" [24].
   - "Ingeniería de Software es esta forma de ingeniería que aplica los principios de las ciencias del computador y matemáticas para lograr soluciones efectivas en costo a problemas de software" [25].
   - "La ingeniería de software incluye procesos, métodos y herramientas que permiten elaborar a tiempo y con calidad sistemas complejos basados en computadoras..."[26].
   - "Ingeniería de Software es la rama de las Ciencias del Computador que crea soluciones prácticas y efectivas en costo a problemas de

> procesamiento de información y computación, aplicando de manera preferente conocimiento científico, desarrollando sistemas basados en software al servicio de la humanidad..."[27].

Dado que este documento se enfoca en la última disciplina enunciada y uno de los objetivos es soportar conceptualmente una propuesta de lineamientos curriculares para dos modalidades: investigación y profundización(profesionalizante), vale la pena cimentar claramente las razones para llamarla ingeniería y a su vez las posibilidades de formación disciplinar investigativa y profesionalizante.

2.1 Caracterización como ingeniería

Los fundamentos o pilares mínimos de cualquier ingeniería serían las características o cualidades genéricas deseables en los ingenieros de software graduados. Estos se pueden resumir en los siguientes [28]:

1. Los ingenieros proceden a la toma de decisiones evaluando cuidadosamente las opciones y escogiendo un enfoque apropiado para la(s) tarea(s) por ejecutar en cada punto decisivo de un proceso, teniendo en cuenta el contexto correspondiente. Lo apropiado del enfoque puede ser valorado por medio de análisis costo-beneficio.

2. Los ingenieros miden atributos de las entidades de interés y cuando es conveniente trabajan con un enfoque cuantitativo, calibran y validan sus mediciones. También usan aproximaciones basadas en juicio experto y datos empíricos.

3. Los ingenieros ponen énfasis en el uso de procesos disciplinados cuando crean un diseño y pueden trabajar en equipo para lograrlo.

4. Los ingenieros pueden ejercer múltiples roles: investigar, desarrollar, diseñar, producir, probar, construir, operar, gestionar y otros roles relacionados como: vender, asesorar y enseñar.

5. Los ingenieros usan herramientas que apoyan el desarrollo sistemático de actividades y tareas dentro de las fases de un proceso; por consiguiente, la escogencia y el uso apropiado de herramientas es clave para la ingeniería.

6. Los ingenieros, a través de sociedades profesionales y comunidades académicas e industriales, avanzan en el desarrollo y la validación de principios, estándares y buenas prácticas de la profesión.

7. Los ingenieros reusan artefactos de análisis, diseño y construcción de los sistemas relacionados con la disciplina que ejercen.

8. Los ingenieros determinan mínimamente tres perspectivas de sistemas en estudio: funcional, estática y dinámica. Con la perspectiva funcional responden al qué se desea con el sistema; con la perspectiva estática determinan las estructuras y sus relaciones para construir el sistema, y con la perspectiva dinámica modelan el comportamiento en tiempo de producción o ejecución del sistema en cuestión.

Con la anterior caracterización se pueden fundamentar los requerimientos mínimos de perfil para el ingreso al programa posgradual que es tema de este trabajo. Sin embargo, se necesita una más precisa diferenciación de la naturaleza propia de la Ingeniería de Software. Esta última posee una naturaleza especial porque pone un gran énfasis en la abstracción, modelado, organización y representación de la información, así como a la gestión del proceso y del cambio de artefactos durante su etapa de desarrollo y mantenimiento. La Ingeniería de Software exige una serie de actividades protectoras de la calidad en el proceso de desarrollo y parte del principio de que el software evoluciona y hay que mantenerlo para evitar su degradación lógica. Otro aspecto relevante es la naturaleza del resultado en esta ingeniería, el producto final en un proceso de desarrollo de software es una máquina lógica ejecutable sobre un computador, no es un resultado material corroíble por medios naturales convencionales. Lo anterior está directamente relacionado con el proceso connatural en esta disciplina que debe aplicar múltiples niveles de abstracción.

2.2 Caracterización como disciplina

Área de conocimiento
AC1. Requerimientos de software
AC2. Diseño de software
AC3. Construcción de software
AC4. Verificación y validación de software
AC5. Mantenimiento de software
AC6. Gestión de configuración de software
AC7. Gestión de procesos de ingeniería de software
AC8. Gestión de ingeniería de software
AC9. Métodos y modelos de ingeniería de software
AC10. Calidad de software
AC11. Práctica profesional de ingeniería de software
AC12. Aspectos económicos de la ingeniería del software
AC13. Fundamentos de computación
AC14. Fundamentos matemáticos
AC15. Fundamentos de ingeniería

Tabla 2.1. Áreas de conocimiento de la Ingeniería de Software. Fuente [29].

La Ingeniería de Software ha consolidado un cuerpo de conocimiento conformado por una relación coherente de áreas de conocimiento (ver Tabla 2.1). Cada área de conocimiento puede generar retos de investigación interesantes, lo cual promueve una tarea para abordar en un programa académico de esta naturaleza: definir los nichos temáticos de trabajo científico en los que la universidad o IES se hará más fuerte (modalidad de investigación) y los nichos problémicos de mercado en los que aspirará a ser líder (modalidad profesionalizante).

El carácter disciplinar de esta ingeniería permite admitir la necesidad de ampliar sus horizontes a través de la investigación. Los cimientos claros en las ciencias de la computación y por ende en las matemáticas discretas abren unas posibilidades inconmensurables de desarrollo investigativo usando métodos formales propios a los pilares teóricos de la disciplina.

Otro aspecto relevante, que abre retos de investigación de otra naturaleza, es la fuerte concentración en entidades abstractas y lógicas en vez de entidades físicas y concretas, comunes en otras ingenierías. El mundo de entidades que se especifican en diversos niveles de abstracción (modelo, meta-modelo y meta-meta-modelo) hacen de la Ingeniería de Software un terreno fértil para el desarrollo de lenguajes maquinales que a futuro incorporarán estructuras abstractas inteligentes como ciudadanos de primera clase. Estos avances generarán nuevas perspectivas en las ciencias de la computación y en la misma disciplina. El reuso de abstracciones como unidades computacionales independientes son avances que ya se han hecho concretos y que en esa vía abren otras posibilidades de investigación en arquitecturas emergentes de software y en optimización autónoma de sistemas basados en software.

La diferencia de los procesos de desarrollo de software frente a los procesos convencionales de manufactura genera retos de gestión que abren perspectivas de investigación en la combinación de modelos de proceso y estudios de caso que aporten sobre la hibridización de estos.

La naturaleza evolutiva del software y de sus arquitecturas subyacentes hacen del mantenimiento de software un terreno productivo para la investigación en nuevos modelos de evolución o control del cambio en los sistemas en producción. Derivado de este tema es el análisis de optimización de sistemas legados, lo que puede generar ahorros sustanciales de recursos computacionales que se pueden tasar en recursos económicos.

2.3 Caracterización como profesión y disciplinas relacionadas

La profesionalización de la Ingeniería de Software busca aportar especialistas en una o varias de las áreas de conocimiento, que brinden garantías de aplicación del conocimiento a problemas reales que otros miembros de la sociedad no tienen capacidad de hacerlo. La idoneidad propia que cualquier profesional debe satisfacer tiene que ir acompañada del cumplimiento de una serie de cualidades que los empleadores demandan en el ejercicio profesional. En Colombia,

este perfilamiento no es común; por esta razón y la cercanía de nuestro modelo de educación superior al de los Estados Unidos, por lo menos en su concepción básica, se tomará como base el informe que hace la *National Association of Colleges and Employers (NACE)* respecto a las destrezas y cualidades que esperan los empleadores de los profesionales que reclutan [30]. Cuando los empleadores fueron indagados por las cualidades que más valoran en los candidatos a un empleo, se ubicaron en orden de preferencia las siguientes:

1. Liderazgo y habilidad para trabajar en equipo.
2. Sólidas habilidades para la comunicación escrita.
3. Destrezas para la solución de problemas.
4. Alto valor ético en el trabajo.
5. Destrezas analíticas.

Lo anterior da indicio de que se deben garantizar estas cualidades como resultado de la formación profesional. En el caso del énfasis en un nivel de formación posgradual avanzado, estas cualidades deben ser satisfechas junto con la formación disciplinar básica requerida, que más adelante se especificará.

La modalidad profesionalizante del énfasis en Ingeniería de Software debe garantizar el manejo idóneo del área o las áreas de conocimiento que el programa académico lidere; lo anterior, en el sentido de la capacidad de aplicar los conocimientos téorico-prácticos a la solución de problemas reales con mayor solidez teórica que un practicante profesional solamente con estudios pregraduales.

Disciplina
DR1. Ingeniería del computador
DR2. Ciencias del computador
DR3. Administración
DR4. Matemáticas
DR5. Administración de proyectos
DR6. Gestión de calidad
DR7. Ergonomía del software
DR8. Ingeniería de sistemas

Tabla 2.2. Disciplinas relacionadas con la Ingeniería de Software. Fuente [29].

Otro elemento analítico para tener en cuenta en el perfil profesional avanzado es la claridad que debe tener el estudiante frente a la relación de la Ingenierá de Software con disciplinas complementarias (ver Tabla 2.2). Entre estas disciplinas vale la pena dar relevancia a la Ingeniería de Sistemas y a la Ingeniería del Computador por su directa relación con estudiantes egresados de la formación pregradual. Antes se hizo claridad sobre el alcance de la Ingeniería del Computador y su fuerte cercanía a la formación en Ingeniería Electrónica. Respecto

a la Ingeniería de Sistemas como disciplina a través de la cual se desarrollan sistemas complejos, confiables y a gran escala [7], se debe tener en cuenta que parte de estos sistemas complejos pueden estar basados en software. Por esta razón, es común encontrar una línea de formación electiva en Ingeniería de Software en muchos programas académicos de Ingeniería de Sistemas en Colombia; este es el caso en nuestra facultad.

Lo expresado en el párrafo anterior genera una razón de pertinencia interna del énfasis en Ingeniería de Software y permite correlacionar de manera natural proyectos de grado pregraduales con los proyectos finales de maestría.

Capítulo 3

Sobre los principios institucionales y el proceso de admisión

La credibilidad, el mayor activo de la universidad, sólo se construye con una institucionalidad madura y fuerte cifrada en las normas y en el respeto al conocimiento.

La universidad pública, como institución fundamental en la sociedad, debe establecer unos principios mínimos generales que demarcan su enfoque y alcance. A continuación se establecen los principios mínimos institucionales y pedagógicos que los programas académicos con énfasis en Ingeniería de Software deberían respetar y aplicar:

I. La formación en el programa académico será perdurable y de alto valor agregado para el ejercicio profesional o de investigador en la disciplina.

II. El pensamiento creativo y crítico será estimulado como elemento fundamental para el uso de criterios transparentes y equilibrados en la aplicación de herramientas y destrezas tecnológicas.

III. El programa académico mantendrá un ejercicio de revisión permanente para garantizar su actualización coherente con el estado del arte en la disciplina. A más tardar cada tres años se efectuará una revisión de contenidos programáticos por parte de un comité académico-científico donde participen los profesores de las diversas áreas de conocimiento. Cada revisión analítica deberá generar un reporte técnico de revisión donde se justifiquen los cambios y actualizaciones que se incorporarán.

IV. La capacidad de autoprendizaje será estimulada por parte de los docentes del programa académico. Las dos modalidades garantizarán a los estudiantes tener conciencia de las rutas temáticas que deben abordar para ser idóneos en el(las) área(s) de conocimiento de su interés.

V. El tiempo de dedicación de los estudiantes es indispensable para una formación competente en los egresados. Para la modalidad profesionalizante

se exigirá una dedicación de mínimo medio tiempo presencial y medio tiempo semipresencial o virtual. Para el caso de la modalidad en investigación la dedicación será de tiempo completo presencial. Este escenario obliga a contar con becas, logística e infraestructura para apoyar a los estudiantes en la modalidad de investigación.

VI. El proceso de admisión debe garantizar que ingresen los estudiantes más idóneos para asumir los retos de formación del programa académico. Un proceso de admisión riguroso impactará directamente en la mejora de la calidad del programa académico y de sus egresados.

VII. El ambiente institucional favorecerá los procesos de enseñanza-aprendizaje, de práctica empresarial y de investigación en las dos modalidades.

VIII. La internacionalización del programa académico será un mínimo por cumplir. Este pilar de trabajo debe garantizarse con el apoyo institucional para la participación frecuente de los profesores y estudiantes del programa académico en los congresos internacionales de mayor calidad en el mundo junto con la realización de pasantías de investigación y de trabajo.

Así mismo, para la visibilización de la producción académico-científica se debe contar con una unidad de apoyo de idiomas extranjeros (con particular énfasis en el idioma inglés). Esta unidad prestará soporte a la traducción y elaboración de reportes técnicos, libros de resultados y artículos de investigación por parte de estudiantes y profesores. En este mismo sentido, facilitará el desarrollo de habilidades y destrezas en cuanto al hablar y escuchar del idioma extranjero de interés.

IX. Dado que el conocimiento procedimental es uno de los más difíciles de enseñar [31] el modelo pedagógico prevaleciente en los cursos del programa académico se centrará en proyectos semestrales de aula que permitirán "aprender haciendo".

3.1 Los requisitos mínimos para la admisión al programa académico

En concordancia con la propuesta presentada por el iSSEc [7] para programas posgraduales en Ingeniería de Software, se enuncian algunas expectativas mínimas de formación para la admisión a un programa avanzad en Ingeniería de Software, manteniendo un nivel comparable al ámbito internacional.

El aspirante debe contar con formación pregradual en computación o en su defecto en una disciplina ingenieril o campo científico relacionado con computación. La Tabla 3.1 enuncia los conocimientos mínimos que debe poseer un aspirante al programa académico en tres bloques asociados con áreas de conocimiento propiamente dichas[1].

[1] Los elementos temáticos escritos en texto de color azul son adicionales a los propuestos por el iSSEc [7]

Áreas de conocimiento	Nivel de Bloom[2]
Fundamentos matemáticos	
1. Matemáticas discretas. Teoría de conjuntos, funciones, relaciones. Lógica básica. Técnicas de prueba. Conceptos básicos de conteo. Teoría de grafos básica. Probabilidad discreta.	**3**
2. Lógica de predicados y proposicional. Proposiciones, operadores, tablas de verdad, leyes de la lógica, cuantificadores, argumentación e inferencia.	**3**
3. Probabilidad y estadística. Conceptos básicos de probabilidad, variables aleatorias y distribuciones de probabilidad, teoría de estimación, prueba de hipótesis, análisis de regresión, análisis de correlación.	**4**
4. Procesos estocásticos. Conceptos básicos de cadenas de markov, conceptos básicos de teoría de decisión.	**4**
Fundamentos en computación	
1. Fundamentos de programación. Conceptos básicos de programación, lenguajes interpretados y no interpretados, máquinas virtuales, POO, programación declarativa, TADs, construcción de intérpretes y compiladores, sistemas de tipos, conceptos básicos de semánticas de lenguajes de programación, diseño de lenguajes de programación.	**3**
2. Estructuras de datos y algoritmos. Análisis básico de algoritmos, técnicas algorítmicas, fundamentos de algoritmos de computación, algoritmos distribuidos.	**2**
3. Arquitectura de computador. Sistemas digitales y lógica digital, lenguaje de máquina, organización de la máquina a nivel de ensamble, arquitectura y organización del sistema de memoria, interfaces de hardware y comunicación, organización funcional, arquitecturas alternativas y multiprocesamiento, arquitectura para redes y sistemas distribuidos, aspectos de desempeño.	**2**
4. Sistemas operativos. Principios y conceptos generales, manejo de concurrencia, administración de tareas, gestión de memoria, gestión de dispositivos, seguridad y protección, sistemas de archivos, sistemas empotrados y de tiempo real, tolerancia a fallos, evaluación de desempeño, guiones de comandos y programación.	**2**
5. Redes y comunicaciones de datos. Conceptos básicos de computación distribuida, comunicación de datos, seguridad de redes de datos, Internet, conceptos básicos de Ingeniería Web, gestión de red, tecnologías de datos multimedia, computación móvil, conceptos básicos de redes inalámbricas.	**2**
6. Construcción y diseño modular. Abstracción, ocultamiento de información, diseño de interfaces, diseño procedimental, excepciones, abstracción de estructuras de programación.	**3**

Áreas de conocimiento	Nivel de Bloom[2]
Fundamentos en Ingeniería de Software	
1. Requerimientos de software. Fundamentos de requerimientos de software, levantamiento de requerimientos, análisis de requerimientos, especificación de requerimientos, validación de requerimientos.	**2**
2. Diseño de software. Fundamentos de diseño de software, diseño arquitectónico y estructural, notaciones de diseño de software, métodos y estrategias de diseño.	**2**
3. Construcción de software. Fundamentos de construcción de software, técnicas de programación en el marco de un modelo de computación específico .	**3**
4. Pruebas de software. Fundamentos de pruebas de software, técnicas de prueba, niveles de aplicación de pruebas, semántica axiomática para la realización de pruebas de software.	**1**
5. Mantenimiento de software. Fundamentos de mantenimiento de software, técnicas para efectuar mantenimiento, tipos de mantenimiento de software.	**1**
6. Gestión de Ingeniería de Software. Planeación de proyectos de software, fundamentos sobre gestión de configuración de software, modelos de estimación de software.	**1**
7. Procesos en la Ingeniería de Software. Definición e implementación de procesos, modelos de medición de proceso y producto.	**1**
8. Calidad de software. Fundamentos de calidad de software, prácticas de gestión de calidad de software, diseño de software conducido por la calidad .	**1**

Tabla 3.1. Prerrequisitos de conocimiento para aspirantes a maestría con énfasis en Ingeniería de Software.

Otro camino que puede garantizar los requisitos mínimos de formación es la experiencia previa en desarrollo de software de los aspirantes. Esta experiencia debe ser de al menos dos años en alguna(s) de las áreas de conocimiento de la Ingeniería de Software y certificada por empresas dedicadas al sector y no generará omisión en el proceso de admisión meritocrático que se establezca para el programa académico.

Otro requisito institucional para la admisión al programa académico, en la modalidad de investigación, es la adscripción previa del aspirante a un grupo de investigación relacionado con el énfasis. Esto debe ir acompañado con el compromiso requerido para la dedicación de tiempo completo del estudiante a través de beca, apoyo empresarial o recursos propios por parte del estudiante.

[2]Para la semántica de los niveles de Bloom, ver sección 4.2.1

Capítulo 4

El perfil esperado del egresado

El paso por la Universidad forma positivamente o deforma a un ser humano. Las deformidades académicas debilitan el activo más valioso de la Universidad: Su credibilidad ante la sociedad.

En esta sección se postula lo que un egresado de un programa de maestría con énfasis en Ingeniería de Software debe satisfacer mínimamente respecto a las capacidades que lo hacen un investigador o profesional avanzado en la disciplina. Inicialmente se establecen resultados de formación comunes a las dos modalidades, influenciados esencialmente por la propuesta hecha en [27] y luego se establecen factores diferenciadores entre la modalidad de investigación y la modalidad profesionalizante.

4.1 Capacidades comunes

Un egresado de una maestría con énfasis en Ingeniería de Software debe ser idóneo para:

1. Determinar, de la manera menos ambigua posible, las necesidades de un cliente y traducirlas a requerimientos de sistema y de software.

2. Reconciliar objetivos conflictivos en proyectos de desarrollo de software; encontrar compromisos aceptables dentro de limitaciones de costo, tiempo y conocimiento requerido.

3. Comprender la naturaleza de problemas no-estructurados o abiertos y de alta complejidad.

4. Diseñar soluciones apropiadas usando enfoques de ingeniería responsables.

5. Evaluar diseños y productos.

6. Comprender y ser hábil para aplicar teorías y modelos que sean fundamento para el diseño de software.

7. Trabajar efectivamente en contextos interdisciplinarios. Tener la capacidad para crear una sinergia positiva entre la tecnología de la computación con la tecnología de los clientes.

8. Interpretar y respetar restricciones técnicas adicionales y propias de los contextos de los clientes.

9. Trabajar efectivamente dentro de sistemas existentes, tanto en contextos de artefactos de software como organizacionales.

10. Comprender y ser capaz de usar elementos de solución técnica actuales que incluyan herramientas específicas, componentes y marcos de trabajo.

11. Abstraer elementos de soluciones computacionales como algoritmos y arquitecturas.

12. Programar de manera efectiva; esto incluye la creación de código, uso y ensamble de componentes e integración de múltiples subsistemas.

13. Aplicar modelos de proceso, métodos de análisis, diseño y desarrollo con las respectivas técnicas para lograr la realización de soluciones basadas en software.

14. Organizar y liderar equipos de desarrollo; incluyendo la fase de conformación del equipo y la etapa de negociación.

15. Comunicar efectivamente, verbalmente y por escrito, los resultados de las diversas fases, actividades y tareas de un proyecto de desarrollo.

16. Aprender nuevos modelos, técnicas y tecnologías emergentes.

17. Integrar conocimiento desde múltiples fuentes para desarrollar soluciones a problemas de desarrollo de software.

18. Servir como un agente de cambio para introducir nuevas tecnologías a un contexto particular del cliente.

4.2 Capacidades esperadas por modalidad

Para la diferenciación explícita de las capacidades esperadas de los estudiantes al final de su proceso de formación, se toman como guía los diez elementos del perfil en cuanto a los aḿbitos técnico, ético y de aprendizaje propuestos por el iSSEc en el 2009 [7]; sin embargo, en este trabajo se contextualizan en lo profesionalizante a nuestro ámbito y se amplía a la modalidad de investigación que no es atendida en la formulación de este consorcio académico-industrial. La Tabla 4.1 describe suscintamente las capacidades esperadas y determinadas en su alcance para cada modalidad.

EJE DE FORMACIÓN	PROFUNDIZACIÓN	INVESTIGACIÓN
Manejo del cuerpo de conocimiento	Habilidad para desarrollar, desde cero, un sistema basado en software de unas cuantas miles de líneas de código fuente. Capacidad de integrar componentes desarrollados por terceros. En el proceso de desarrollo y evolución del sistema se manejan de manera idónea el análisis, diseño y validación del producto de software. Los artefactos de software producidos serán de alta calidad.	Adicional al alcance profesionalizante se debe contar con un conocimiento profundo de los niveles de meta-metamodelado de por lo menos dos áreas de conocimiento de la Ingeniería de Software y un acercamiento introductorio a los niveles de metamodelado o modelado en las demás áreas de conocimiento.
Cobertura de dominio [1]	Comprender las diferencias entre dominios de problema y tipos de aplicación en tanto el software requerido y los modelos de proceso, métodos y técnicas de ingeniería de software a aplicar pueden diferenciarse (Para un buen ejemplo revisar [32]).	Diferenciar conceptualmente y en cuanto a alcance los perfiles de los metamodelos para que sean aplicados en diferentes dominios de aplicación. A nivel de meta-metamodelado, por naturaleza independiente del dominio de aplicación, manejar profundamente por lo menos una propuesta que tenga la capacidad representativa de algunas de las propiedades relevantes y transversales del software que le aportan complejidad: Concurrencia, secuencialidad, movilidad y distribución, entre otras.
Profundidad del conocimiento	Manejo profundo de por lo menos un área o subárea de conocimiento de la Ingeniería de Software que se debe evidenciar claramente en el trabajo de grado.	Manejo profundo del nivel de meta-metamodelado en por lo menos un área o subárea de conocimiento; se debe evidenciar claramente en los niveles de formalismo utilizados en la tesis de maestría o doctorado.
Aspectos éticos	Habilidad para sopesar las decisiones éticas que involucra un proyecto de desarrollo y la continuidad de un comportamiento ético profesional que favorezca a la sociedad y colectivos receptores de utilidad del software en desarrollo o terminado. Esto debe propiciarse con momentos de práctica empresarial facilitados por la universidad o IES.	Habilidad para sopesar las decisiones éticas que involucra un proyecto de desarrollo y la continuidad de un comportamiento ético profesional que favorezca a la sociedad y colectivos receptores de utilidad del software en desarrollo o terminado. Esto debe propiciarse con momentos de trabajo en equipo con el grupo de investigación al que se encuentra adscrito el estudiante.
Interdisciplinariedad afín	Comprender la relación entre la Ingeniería de Software y otras disciplinas con la aplicación de sus principios y mejores prácticas al desarrollo de proyectos de desarrollo de software.	Comprender de manera profunda la relación entre los fundamentos teóricos de otras disciplinas y la Ingeniería de Software. Un egresado del énfasis en la modalidad de investigación debe contar con la capacidad de relacionar y aplicar los principios y mejores prácticas de otras disciplinas en la Ingeniería de Software.
Trabajo en equipo	Capacidad de ser miembro de equipos de desarrollo, pueden ser equipos geográficamente dispersos y de connotación multinacional. Para esto debe contarse con el manejo de por lo menos un segundo idioma, preferiblemente el inglés. Se debe contar con habilidades y destrezas efectivas de comunicación oral y escrita a nivel profesional. Lo anterior se debe conjugar con la capacidad del egresado de liderar trabajo en equipo de por lo menos un área del desarrollo de proyectos: Gestión de proyectos, análisis de requerimientos, arquitectura y diseño, construcción, gestión de la calidad o verificación y validación.	Capacidad de ser miembro de equipos de investigación e innovación, pueden ser equipos geográficamente dispersos y de connotación multinacional. Para esto debe contarse con el manejo de por lo menos un segundo idioma en nivel avanzado, preferiblemente el inglés. Se debe contar con habilidades y destrezas efectivas de comunicación académica oral y escrita. Lo anterior se debe conjugar con la capacidad del egresado de liderar trabajo en equipo de por lo menos un área de I+D en alguna de las áreas de conocimiento de la Ingeniería de Software.
Manejo del contexto	Capacidad de conciliar objetivos conflictivos de proyectos y establecer compromisos aceptables y evaluables con limitaciones de costos, tiempo, conocimiento, riesgos, sistemas existentes y asociadas a las características organizacionales.	Manejo profundo de los fundamentos teóricos de técnicas de estimación, medición y gestión de proyectos de software con capacidad de aplicarlos a proyectos reales.

EJE DE FORMACIÓN	PROFUNDIZACIÓN	INVESTIGACIÓN
Capacidad de auto-aprendizaje	Habilidad para aprender nuevos modelos, técnicas y tecnologías emergentes siendo consciente de la necesidad del desarrollo profesional.	Habilidad para efectuar revisiones sistemáticas de literatura [33] y para comprender de manera profunda los artículos publicados en su especialidad en el ámbito académico y de investigación con el objetivo de mantenerse al tanto de los avances en las áreas de conocimiento de su interés [2].
Capacidades tecnológicas	Habilidad para analizar tecnologías de software actuales respecto a sus debilidades y fortalezas con el objetivo de aplicarlas adecuadamente.	Conocimiento profundo de los desarrollos científicos subyacentes a las tecnologías actuales para el desarrollo y puesta en producción de software. Habilidad para determinar los fundamentos teóricos de tecnologías de software actuales en por lo menos un área de conocimiento.

Tabla 4.1. Resultados generales de formación por modalidad para el perfil del egresado.

[1]Este eje de formación propone el reto de determinar sobre qué dominios de trabajo práctico se desea que los estudiantes aborden problemas del contexto; dominios de aplicación pueden ser: Transporte, telecomunicaciones, salud, educación, financiero, agroindustria, gobierno, entre otros. Los tipos de aplicación en cada dominio pueden ser: Sistemas distribuidos, sistemas habilitados para la Web, aplicaciones móviles, empotradas en hardware, con capacidades de inteligencia computacional, de tiempo real, de fiabilidad-crítica, de "e-learning", entre otras.

[2]Un currículo de maestría(modalidad de investigación) debe ofrecer cursos de entrenamiento en cienciometría y bibliometría.

4.2.1 Modelando el alcance en los perfiles de formación

Las taxonomías de aprendizaje son útiles para definir los objetivos de formación de los cursos de un programa académico. Esta definición de objetivos no solo se soporta en los tópicos temáticos por ser cubiertos, sino también en el nivel de comprensión de los mismos en la disciplina en cuestión [34].

Los niveles de Bloom para el cumplimiento de objetivos de aprendizaje desde la perspectiva cognitiva [35] son un bondadoso instrumento para la determinación de los requerimientos mínimos por cumplir por parte de un egresado del énfasis en Ingeniería de Software. La utilidad de la taxonomía de Bloom está en facilitar la determinación del alcance en amplitud temática y profundidad de cualquier curso o currículo que facilite su contraste. Los objetivos educacionales, trabajados desde esta perspectiva, se organizan jerárquicamente de lo menos a lo más complejo; es decir, el cumplimiento de un nivel indica que el estudiante ya maneja los anteriores. La semántica de logros, en cada nivel cognitivo, de la propuesta original de Bloom [35],[36] se resume en la Tabla 4.2 con una presentación en filas de fondo coloreado, útil en adelante para relacionar cada color con un nivel cognitivo:

NIVEL COGNITIVO	DESCRIPCIÓN	VERBOS RELACIONADOS	CONTEXTUALIZACIÓN A LA ING. DE SOFTWARE[31]
1. Conocimiento	Está relacionado con recordar ideas, información y principios en una forma aproximada al modo en que fueron aprendidos.	Definir, identificar, nombrar, rotular, enunciar, listar y emparejar o igualar a nivel de sinonimia, entre otros.	Los estudiantes deben conocer la existencia de métodos, herramientas y modelos adicionales a las de su formación pregradual y fruto de la experiencia laboral. Deben ser conscientes que, dado el carácter altamente dinámico y cambiante de la disciplina, siempre existirá algo más que aprender.
2. Comprensión	Capacidad de explicar, traducir o interpretar el significado de información basada en el aprendizaje previo.	Describir, generalizar, ilustrar, resumir y parafrasear, entre otros.	Los estudiantes deben comprender: • El proceso de software tanto en sus modelos abstractos como en sus ejemplificaciones que se aplican en la industria. • Los eventos que han motivado el crecimiento y evolución de la disciplina. • Los aspectos de construcción de software bajo limitantes de las actividades de gestión y control. • Un conjunto razonable de principios, modelos, metamodelos, representaciones, métodos y herramientas. • El papel de analista y evaluador en la Ingeniería de Software. • Los paradigmas de diseño existentes para sistemas bien conocidos. • Las razones y contenidos de los estándares que se aplican en la disciplina. • Asuntos éticos, legales y económicos relacionados con la disciplina.
3. Aplicación	Se corresponde con el uso de abstracciones en situaciones concretas; es decir, el(la) estudiante es capaz de seleccionar, transferir y usar datos y principios para resolver un problema o tarea con mínima dirección o tutoría.	Determinar, ejecutar, graficar, implementar, resolver, preparar, calcular, usar, construir, resolver y desarrollar, entre otros.	Los estudiantes deben ser hábiles para: • Aplicar los principios fundamentales a las diferentes actividades de desarrollo de software. • Aplicar métodos formales para lograr resultados. • Usar las herramientas apropiadas a las diversas actividades del proceso de desarrollo de software. • Recoger datos apropiados con el propósito de mejorar la gestión de proyectos y ejecutar análisis y valoración del proceso y el producto. • Ejecutar planes de pruebas, de aseguramiento de la calidad y de gestión de la configuración de software. • Usar de manera adecuada estándares de documentación recomendados para la Ingeniería de Software.

NIVEL COGNITIVO	DESCRIPCIÓN	VERBOS RELACIONADOS	CONTEXTUALIZACIÓN A LA ING. DE SOFTWARE[31]
4. Análisis	Se refiere a dividir un todo en sus partes teniendo en cuenta sus relaciones; es decir, el(la) estudiante distingue, clasifica y relaciona suposiciones, hipótesis, evidencias o estructuras de un enunciado, instrucción o cuestionamiento (Pregunta).	Puntualizar, remarcar, señalar, diferenciar, distinguir, discriminar y comparar, entre otros.	Los estudiantes podrán participar en revisiones técnicas formales de diversos productos relacionados con los entregables (Documentos, diseños, planes, código, entre otros) de un proyecto de desarrollo de software. También tendrán la capacidad de efectuar un análisis metódico de las necesidades de los clientes.
5. Síntesis	Está relacionado con colocar partes juntas que conforman un todo con un propósito funcional intencionado; en este nivel el(la) estudiante origina, integra y combina ideas en el logro de un producto, plan o propuesta que es nueva para él o ella. Está fuertemente relacionado con el *pensamiento creativo*.	Crear, elaborar, diseñar, planificar, organizar, generar y escribir, entre otros.	Los estudiantes tendrán la capacidad de: • Generar los productos o entregables de diversas actividades del proceso de desarrollo de software, tales como: especificaciones de requerimientos, diseños, código y documentación. • Elaborar planes de proyecto, planes de pruebas y planes de aseguramiento de la calidad. • Diseñar modelos de dominio para memoria volátil y modelos de persistencia para memoria de largo plazo. • Diseñar pruebas de software. • Preparar presentaciones técnicas relacionadas con proyectos de software. • Elaborar planes de revisiones técnicas e inspecciones de software.
6. Evaluación	Aborda la elaboración y emisión de juicios sustentados sobre ideas, materiales o fenómenos; en este caso el(la) estudiante aprecia, valora o critica basado en estándares específicos o criterios desarrollados durante su formación académico-científica. Este nivel se relaciona directamente con lo que se ha denominado el *pensamiento crítico*.	Evaluar, valorar, tasar, criticar, juzgar, sopesar, validar, verificar y seleccionar, entre otros.	Los estudiantes tendrán habilidades para: • Evaluar entregables del proceso de software respecto al cumplimiento de estándares. • Usar medidas cuantitativas y cualitativas en la evaluación de productos de trabajo del proceso de desarrollo de software. • Ejecutar procesos de validación y verificación de software. • Aplicar y validar modelos predictivos tales como los modelos de estimación de software. • Valorar herramientas y tecnologías para usarlas en su trabajo.

Tabla 4.2. Niveles cognitivos de la taxonomía de Bloom.

Esta taxonomía de aprendizaje ha sido discutida ampliamente frente a su utilidad en las disciplinas ingenieriles relacionadas con las Ciencias de la Computación [37] y la Ingeniería de Software [38]; una propuesta revisada de esta fue efectuada por David Krathwohl [39]. La revisión propuso los siguientes ajustes al modelo original:

1. Reemplazar algunos de los términos que nombran los niveles de conocimiento de Bloom pasándolos a formas infinitivas verbales, así: Conocimiento lo renombra como Recordar; Comprensión lo renombra como Entender o Comprender; Aplicación lo renombra como Aplicar; Análisis lo renombra como Analizar; Síntesis lo renombra como Crear y Evaluación lo renombra como Evaluar. Este cambio hace los términos más cercanos a la manera como los docentes se refieren a la evolución en la formación de sus estudiantes.

2. Se agrega una nueva dimensión a la de los procesos cognitivos, la cual se denomina Conocimiento. Se puede dar a esta dimensión un nombre que la acerca a una semántica más precisa en nuestro idioma español: naturaleza del conocimiento. En este sentido, se proponen cuatro posibles tipos de conocimiento ortogonales a los niveles de Bloom:

   a) **Conocimiento fáctico**. Comprende los elementos básicos que un estudiante debe conocer respecto a una disciplina. Esto incluye:
      - La terminología propia de la disciplina.
      - Los elementos y detalles específicos de la disciplina.

   b) **Conocimiento conceptual**. Comprende las interrelaciones entre los elementos específicos de la disciplina que los habilitan a ser combinados, integrados o mezclados para que funcionen apropiadamente. Esto incluye el conocimiento de:
      - Categorías y clasificación.
      - Principios y generalizaciones.
      - Teorías, modelos y estructuras.

   c) **Conocimiento procedimental**. Se refiere al *cómo hacer algo*; métodos y modelos de proceso junto con los criterios para aplicar las herramientas, destrezas, algoritmos, técnicas para obtener entregables de ingeniería. Este conocimiento incluye:
      - Destrezas específicas y algoritmos para obtener resultados.
      - Técnicas y métodos específicos asociados al contexto disciplinar.
      - Criterios de selección de técnicas, métodos, algoritmos, herramientas o destrezas específicas para solucionar los problemas que se aborden en el contexto disciplinar.

   d) **Conocimiento meta-cognitivo**. Para el contexto de la Ingeniería de Software se subdividirá en metamodelado y meta-metamodelado del conocimiento propio de la disciplina. En el nivel doctoral y de

maestría en investigación, es requisito mínimo de formación final para el egresado. Esto incluye:

- Comprensión y manejo del nivel de metamodelado en las áreas de interés de la disciplina (obligatorio para la modalidad de profundización e investigación).
- Comprensión y manejo del nivel de meta-metamodelado para las áreas de interés de la disciplina (obligatorio para la modalidad de investigación).

Krathwohl propone un arreglo bidimensional para determinar los objetivos de cada unidad programática de formación; algunas universidades la han utilizado para ubicar los objetivos en la celda que corresponde al cruce entre la naturaleza de conocimiento abordado y el nivel cognitivo de Bloom esperado [39], como se presenta en la Tabla 4.3.

N. cognitivo / Nat. conocimiento	1. Recordar	2. Entender	3.Aplicar	4. Analizar	5. Crear	6. Evaluar
NF. FÁCTICO	Objetivo 1					
NC. CONCEPTUAL		Objetivo 2				
NP. PROCEDIMENTAL			Objetivo 3	Objetivo 4		
NMC. METACOGNITIVO					Objetivo 5	Objetivo 6

Tabla 4.3. Ejemplo de tabla de objetivos usando la taxonomía de Bloom revisada.

Este enfoque podría ser útil al aplicarlo a la planeación curricular de cada área de conocimiento cubierta en las modalidades objetivo de una maestría con el énfasis en Ingeniería de Software; cada titular de asignatura debe diligenciar la configuración tabular presentada en la Tabla 4.3 teniendo en cuenta que para la modalidad de profundización la naturaleza del conocimiento debe cubrir hasta el nivel NP (procedimental) y en alcance cognitivo debe llegarse hasta el nivel 5 (crear). En el caso de la modalidad de investigación la naturaleza del conocimiento debe llegar hasta el meta-metamodelado NMC (metacognitivo) y el nivel cognitivo estará en el 6 (evaluar). El cumplimiento de un diseño curricular con este enfoque puede satisfacer los resultados de formación ilustrados en la Tabla 4.1.

La propuesta presentada por Azuma et al. en [38] combina la taxonomía de Bloom con seis categorías aplicables a las áreas de conocimiento. Las categorías son:

A. Concepto, marco de trabajo o modelo de referencia.

B. Principio o teoría.

C. Método o técnica.

D. Medición o valoración.

E. Caso de estudio o de aplicación.

F. Herramienta o manejo tecnológico de una herramienta.

Estas categorías son útiles en cuanto reunen un agrupamiento conceptual de los niveles de alcance asociados a la naturaleza de la disciplina. La forma de aplicar estas categorías a algunas áreas de conocimiento son ejemplificadas por Azuma et.al. en [38]. En el caso que nos ocupa, se presenta a continuación un modelo que combina la taxonomía de Bloom [35],[36], la naturaleza del conocimiento abordado [39] y las categorías de las áreas de conocimiento para la Ingeniería de Software [38]; el objetivo primario es el de facilitar la determinación de alcance de programas pregraduales y posgraduales en Ingeniería de Software con una representación gráfica que sintetice decisiones curriculares de este ámbito.

4.2.2 Los prismas de alcance de formación

La propuesta para determinar alcances de formación generales en programas académicos se ilustra a través de las Figuras 4.1 - 4.6, las cuales combinan las tres propuestas antes revisadas con el objeto de ubicar claramente el alcance de formación pregradual, maestría de profundización, maestría de investigación y doctorado. Esta presentación tridimensional determina prismas rectangulares que establecen claramente la cobertura de formación máxima y un volumen dentro del cual se debe mover cualquier programa pregradual, de maestría o de doctorado. Se postula, en cada categoría del área de conocimiento, el alcance de formación a nivel cognitivo en el eje de Bloom y la naturaleza del conocimiento a abordar usando el eje vertical (naturaleza del conocimiento). Se puede notar, en el eje vertical, que el tipo de conocimiento meta-cognitivo se ha discretizado en dos alcances: metamodelado y meta-metamodelado. Esta división permite ubicar con mayor certeza el alcance de formación diferenciado en los énfasis de investigación y de profundización[3].

La naturaleza incluyente de los niveles superiores respecto a sus antecesores en los ejes horizontal y vertical establece la cobertura de alcance de formación por cada categoría de área de conocimiento de la ingeniería de software. En la práctica, cada docente o grupo docente de área de conocimiento diseña los contenidos programáticos de las asignaturas ubicando los temas en alguna de las categorías de áreas de conocimiento[4]. Los ejes de cobertura de niveles de Bloom y tipos de conocimiento determinan el alcance de formación en cada temática; es decir, este modelo permite establecer de manera gráfica el alcance de formación para el programa académico.

El modelo aquí propuesto es aplicable a otros programas académicos que asuman como premisa de alcance de formación la taxonomía de Bloom y la tipología de la naturaleza del conocimiento de Krathwohl. Sólo se debe variar el eje de categorías por área de conocimiento que debe ser coherente con la naturaleza de la disciplina.

[3]Para una interpretación más detallada del papel de estos niveles de abstracción en la Ingeniería de Software se puede consultar a [40], una reflexión efectuada al respecto por el autor.

[4]Cada temática podría ubicarse en alguna de las categorías. Estas categorías podrían ampliarse o precisarse mejor a criterio de los conocedores de la disciplina

Es importante resaltar que la discretización de cada categoría de área de conocimiento se puede dar en la medida que existan temáticas pertenecientes a cada una de ellas; estas a su vez se mapean al correspondiente nivel cognitivo de Bloom deseado y tipo de conocimiento a abordar. Lo anterior generaría una serie de vectores de satisfacción de formación por temática.

La interpretación de las gráficas de alcance de formación sería la siguiente(para el ejemplo presentado): En el caso de la Figura 4.1, la formación en conceptos básicos, modelos de referencia o marcos de trabajo a nivel pregradual tendrá un alcance cognitivo de formación pregradual que permitirá a los estudiantes Aplicar[5] lo aprendido, la cobertura mínima de tipo de conocimiento será Fáctico + Conceptual. Para el énfasis de profundización de la maestría en esta categoría, se espera que el egresado satisfaga el nivel cognitivo de Analizar abarcando tipos de conocimiento Fáctico + Conceptual + Procedimental + Metamodelado. Para el énfasis de investigación de la maestría en esta categoría, se espera que el egresado satisfaga el nivel cognitivo de Evaluar abarcando tipos de conocimiento Fáctico + Conceptual + Procedimental + Metamodelado + Meta-metamodelado. De manera similar, se interpretan las Figuras 4.2 - 4.6.

Una interpretación sintética del alcance de formación posgradual en una maestría para las modalidades de profundización e investigación, ilustrada en las Figuras 4.1 - 4.6, lleva a concluir que los niveles cognitivos mínimos a alcanzar en la modalidad de profundización se mueve entre Aplicar - Analizar - Crear con el abordaje de tipos de conocimiento que se mueven entre el Procedimental y el Metamodelado. Para el caso de la modalidad de investigación, los niveles cognitivos mínimos a lograr se mueven entre Crear - Evaluar y el tipo de conocimiento que debe acompañar el discurso del egresado es de Meta-metamodelado; es decir, formalismos de lenguajes para diseñar lenguajes que permitan elaborar modelos.

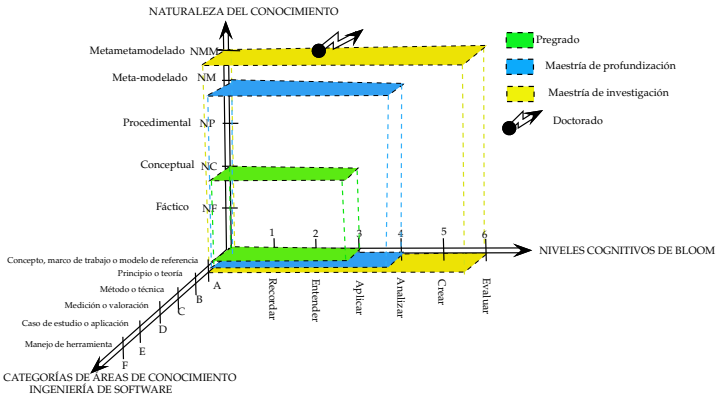


Figura 4.1. Alcance de formación para categoría A de temáticas.

[5]Recuérdese que la taxonomía de Bloom supone que los niveles cognitivos anteriores se han satisfecho.

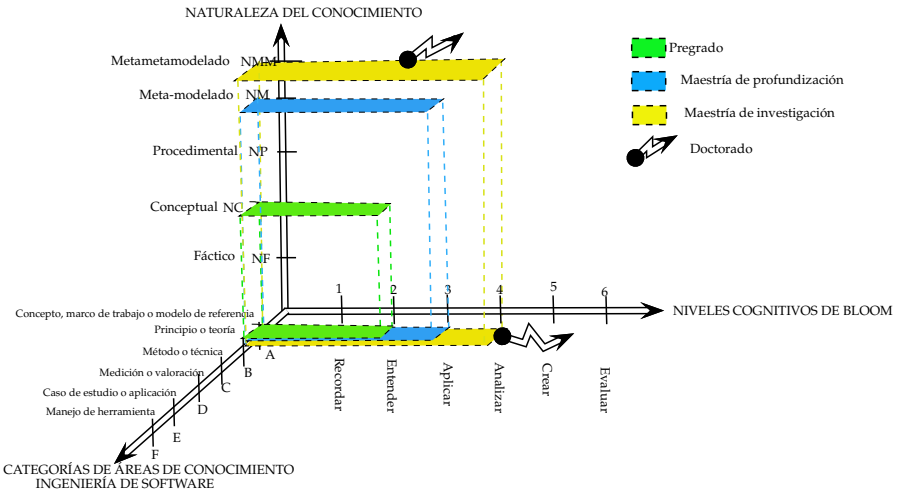


Figura 4.2. Alcance de formación para categoría B de temáticas.

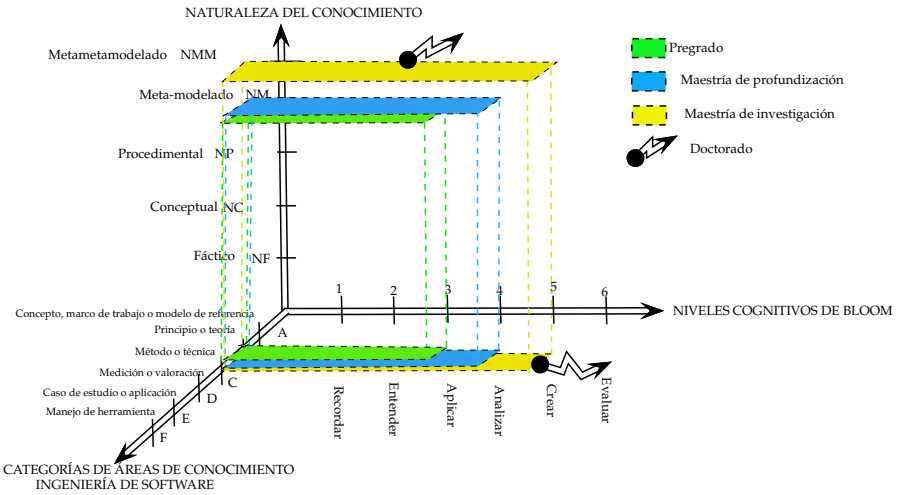


Figura 4.3. Alcance de formación para categoría C de temáticas.

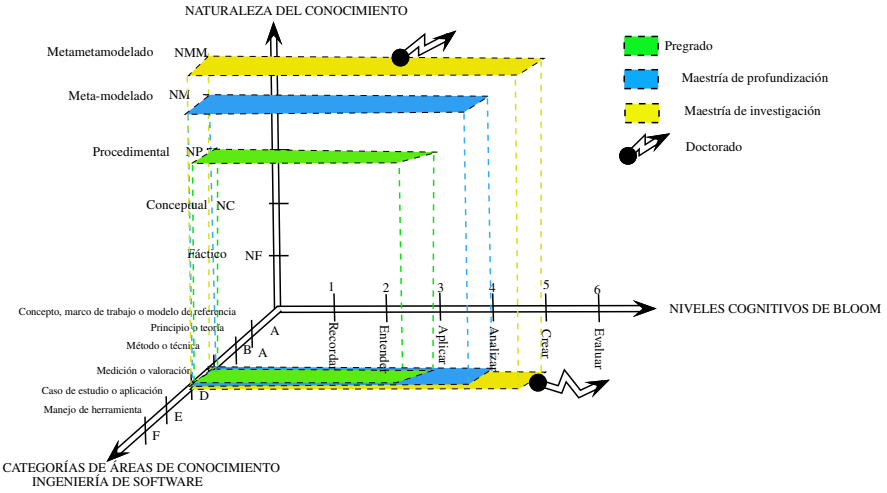


Figura 4.4. Alcance de formación para categoría D de temáticas.

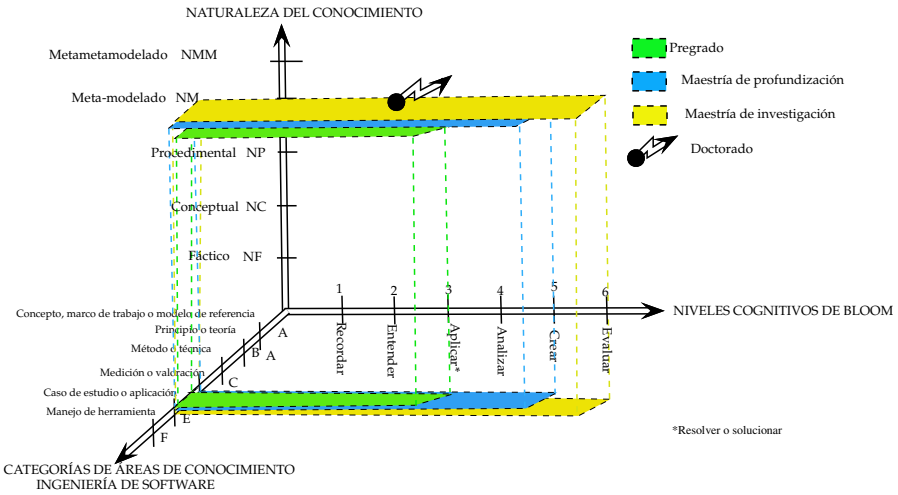


Figura 4.5. Alcance de formación para categoría E de temáticas.

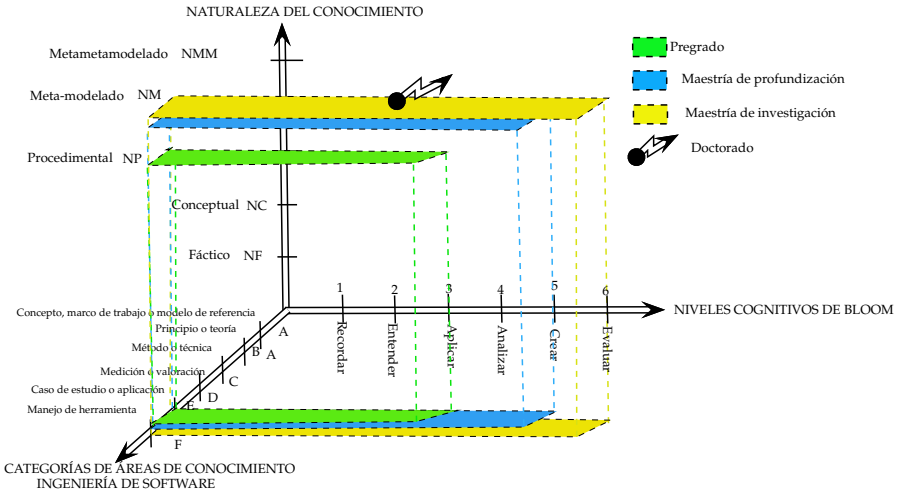


Figura 4.6. Alcance de formación para categoría F de temáticas.

Como se puede observar, el nivel doctoral exige aportar a un avance en la frontera del conocimiento en el tipo de conocimiento de meta-metamodelado.

Capítulo 5

Un ejemplo de arquitectura de currículo posgradual

Son compromisos naturales de la universidad: La búsqueda rigurosa de nuevos saberes y la consolidación o re-evaluación de los existentes. En el cumplimiento de esos compromisos confía, a veces ciegamente, la sociedad.

La Figura 5.1 sintetiza una propuesta de currículo para un programa de formación de Maestría en Ingeniería de Software que acoge las recomendaciones del proyecto iSSSec [7]. Se puede resumir la estructura en cinco niveles de formación:

N0 . Nivel de formación previa de aspirantes.

N1 . Nivel de cursos fundamentales.

N2 . Nivel de formación específica.

N3 . Nivel de cursos electivos.

N4 . Nivel de proyecto de grado (modalidad de profundización) o tesis (modalidad de investigación).

La línea base se corresponde con aspirantes, cuyo título de pregrado permitiría su aceptación sin abordar cursos nivelatorios o material preparatorio para asumir la formación que ofrece el programa académico. Este requisito no excluye un proceso de admisión donde se escojan los mejores aspirantes.

Nivel de formación previa de aspirantes. Será referenciado en adelante como el **Nivel 0 de formación**. En el caso de otros perfiles profesionales, con título pregradual en disciplinas diferentes a las de la línea base, se requerirá un proceso preparatorio que implica satisfacer los contenidos temáticos propuestos en la Tabla 3.1 con el cumplimiento de los niveles cognitivos de Bloom definidos. El aspirante podrá demostrar el manejo de estos contenidos temáticos a través de un examen o la participación en cursos preparatorios que la universidad facilitará semestralmente a través de cursos pregraduales. Estos cursos preparatorios se orientarán a las temáticas-requisito del ingreso al programa académico.

Nivel de cursos fundamentales. Será referenciado en adelante como el **Nivel 1 de formación**. A partir del ingreso al programa de maestría, con la

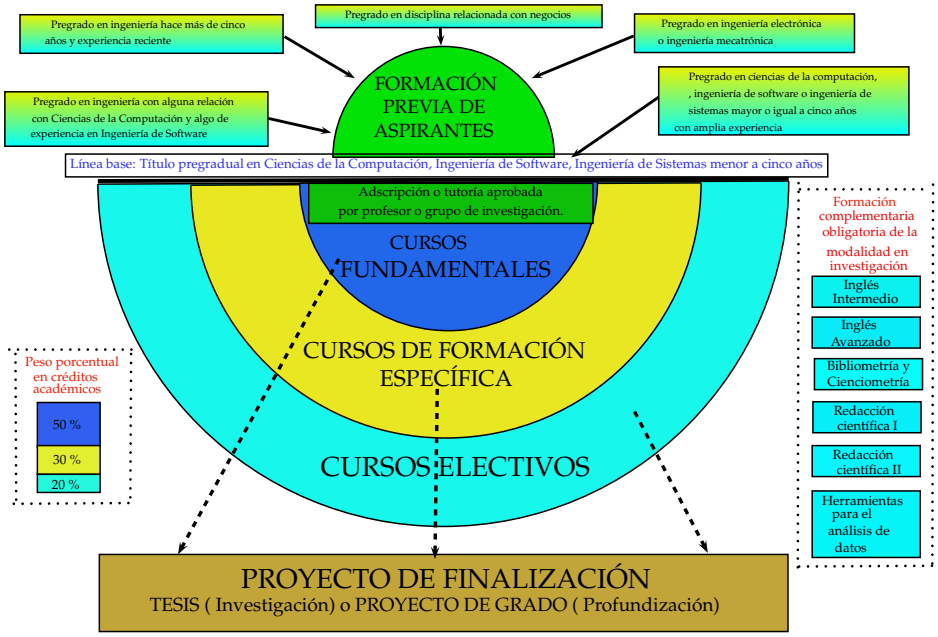


Figura 5.1. Arquitectura de currículo posgradual en Ingeniería de Software.

garantía de idoneidad que debe dar el proceso de admisión, el estudiante asumirá una serie de cursos fundamentales de la disciplina de Ingeniería de Software. Estos cursos fundamentales brindarán una panorámica de la disciplina y orientarán al estudiante hacia su especialización en una de las áreas de conocimiento en las que los profesores o grupos de investigación que soportan el énfasis desarrollan su trabajo.

Nivel de formación específica. Será referenciado en adelante como el **Nivel 2 de formación**. Los cursos de formación específica son los ofertados por los profesores o grupos de investigación que soportan el énfasis. Estos cursos se enfocan a la formación profunda sobre temas de la ingeniería de software con alcance investigativo o profesionalizante en los que dichos profesores o grupos de investigación centran su interés cotidiano y de proyección de su trabajo. El programa académico ofertará una bolsa de cursos por modalidad que permitirá al estudiante con su director de proyecto de grado o tesis elegir los más adecuados para lograr llevar a buen término su proyecto de grado o tesis.

Nivel de cursos electivos. Será referenciado en adelante como el **Nivel 3 de formación**. Los cursos electivos, ofertados desde los profesores, grupos de investigación o eventos institucionales diseñados para tal fin (aplica a cursos complementarios) estarán en concordancia con los intereses de profesionalización e investigación de estos. Estos cursos se ofertarán como una bolsa de cursos opcionales que el estudiante tomará orientado por su director de tesis o de proyecto de grado según sea la modalidad respectiva de formación. Los cursos comple-

mentarios obligatorios pueden hacer parte de estos cursos electivos, excepto los correspondientes a la formación en un idioma extranjero.

Para el caso de la modalidad de investigación se exigirá el cumplimiento de la formación complementaria obligatoria presentada al lado derecho de la Figura 5.1. Esta formación complementaria se considera como parte de los requerimientos mínimos de un investigador en cualquier disciplina. La universidad o IES ofrecerá semestralmente los cursos complementarios obligatorios o facilitará la participación en eventos de formación orientados hacia estos complementos. Adicionalmente, los estudiantes deberán estar adscritos a un grupo de investigación desde el que se orientará el proyecto de tesis de interés para el estudiante y la comunidad académica que lo acoge.

Nivel de proyecto de grado (modalidad de profundización) o tesis (modalidad de investigación). Será referenciado en adelante como el **Nivel 4 de formación**. El proyecto de finalización para la modalidad de investigación consistirá en una tesis desarrollada de manera individual y cuyos resultados deben abrir el camino a la tesis doctoral. En el ejercicio de tesis, el futuro investigador tendrá que demostrar sus capacidades de pensamiento crítico y creativo además de las competencias científicas que son inherentes a un investigador (ver Tabla 4.1). El discurso del estudiante, al defender su tesis de maestría en esta modalidad, debe estar en el nivel del meta-metamodelado.

Para el caso de la modalidad profesionalizante o de profundización bastará con un proyecto de grado consistente en la aplicación de los conocimientos para solucionar un problema de la industria (puede ser con una estadía de mínimo un semestre en una compañia de software nacional) o un problema de desarrollo de software abordado con la tutoría de un profesor o grupo de investigación que implique el desarrollo de una solución basada en software y donde, por lo menos, se ejercite intensivamente una de las áreas de conocimiento de la Ingeniería de Software. El discurso del estudiante, al defender su tesis de maestría en esta modalidad debe ser consistente con el nivel de apropiación tecnológica y de meta-modelado.

5.1 La naturaleza transversal de las temáticas

Uno de los inconvenientes con un modelo que presenta la formación por capas o en niveles jerárquicos es que oscurece la naturaleza transversal de ciertos elementos del cuerpo de conocimiento de la Ingeniería de Software [7]. Áreas de conocimiento como la de aspectos de calidad del software no se pueden separar o desligar de otras áreas como diseño, construcción, procesos, entre otras. Esta naturaleza transversal no debe perderse de vista y ayuda a comprender mejor que los alcances temáticos pueden cubrirse de manera combinada o integrada en diferentes cursos; es decir, aunque la propuesta aquí presentada discretice de manera clara los temas, estos pueden cubrirse a través de varios cursos que involucran su estudio. La Ingeniería de Sistemas es una de las áreas de conocimiento transversales de mayor importancia, realmente es una de las disciplinas soporte a la que aquí se aborda; sus aportes serán irrigados a través

de todo el ciclo de formación, tanto de profundización como de investigación. Es por esta razón que los ingenieros de sistemas son buenos candidatos para ingresar a este énfasis de formación posgradual.

5.2 Propuesta de alcance de los cursos fundamentales

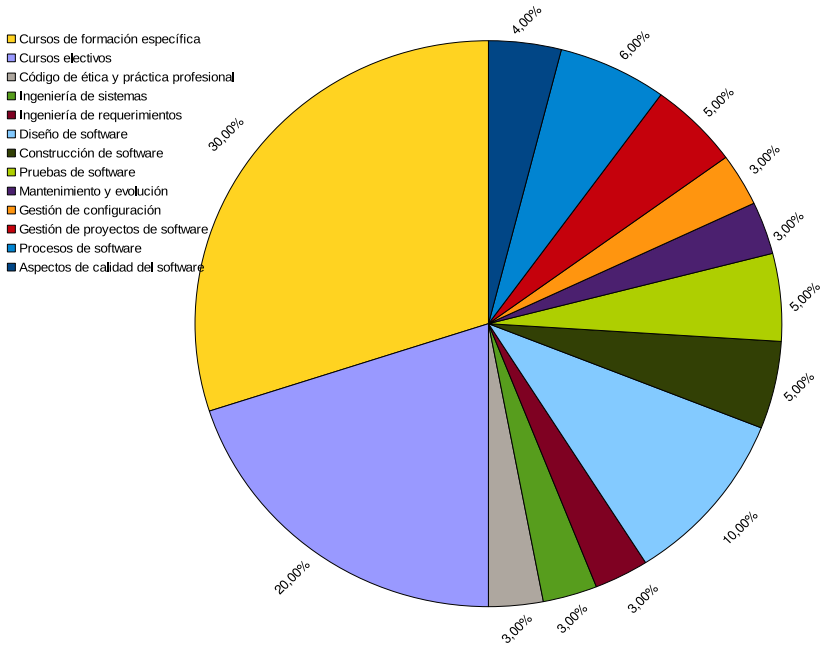


Figura 5.2. Peso porcentual de áreas de conocimiento en cursos fundamentales vs. niveles de formación flexible del currículo.

La propuesta de cursos fundamentales del énfasis en Ingeniería de Software es consistente con la propuesta del iSSEc [7] y está actualizada respecto a la última propuesta del cuerpo de conocimiento [29], teniendo en cuenta los conocimientos esenciales de la disciplina y suponiendo que los fundamentos de computación, matemáticos, de gestión económica de proyectos y de ingeniería son satisfechos por la formación del estudiante que ingresa. Cubrir, en su totalidad temática, las áreas de conocimiento en un programa académico de formación posgradual no es factible. Por esta razón, se espera que el programa académico defina un peso mayor o menor de cada área de conocimiento de acuerdo con los objetivos de posicionamiento en la disciplina a nivel de innovación e investigación en el ámbito nacional e internacional. La propuesta aquí presentada da mayor énfasis

a las áreas de conocimiento en las que la trayectoria de los grupos de investigación y profesores del programa han trabajado más intensamente. Los cursos fundamentales tienen relación con alguna de las siguientes áreas de conocimiento en el peso porcentual que ilustra la Figura 5.2:

1. **Código de ética y conducta profesional.**
2. **Ingeniería de Sistemas.**
3. **Ingeniería de requerimientos.**
4. **Diseño de software.**
5. **Construcción de software.**
6. **Pruebas de software.**
7. **Mantenimiento y evolución de software.**
8. **Gestión de configuración.**
9. **Gestión de proyectos de Ingeniería de Software.**
10. **Procesos en la Ingeniería de Software.**
11. **Aspectos de calidad del software.**

Dado que los cursos fundamentales no deben incurrir en el error de repetir los contenidos programáticos de cursos de pregrado, se presenta en la Tabla 5.1 una propuesta de alcance temático mínimo en cada una de las áreas de conocimiento arriba enunciadas. No hay que descartar que se deba hacer un cruce periódico de contenidos temáticos entre los cursos pregraduales y posgraduales para evitar redundancia de formación académica. La Figura 5.2 puede ajustarse, en los pesos porcentuales de cada área de conocimiento, en concordancia con los énfasis de formación que el programa académico determine respecto a ciertas áreas de conocimiento; lo anterior, basado en las fortalezas internas e intereses de la comunidad académica; es decir, los pesos porcentuales propuestos son aproximados y no son una camisa de fuerza. El ejercicio de definición de los pesos porcentuales determina directamente la densidad temática por cubrir en créditos académicos; por lo tanto, es importante que estas determinaciones las hagan los profesores de más alta idoneidad en la disciplina.

Adicionalmente, la Tabla 5.1 propone un número de créditos aproximado dedicado a cada temática. El valor de referencia del número total de créditos del programa posgradual es de cuarenta, en concordancia con la media de programas académicos actualmente aprobados por el Consejo Nacional de Acreditación en Colombia[1]. El número de créditos de referencia arriba mencionado deja entre 4 a 8 créditos para el desarrollo del proyecto de grado final (4 créditos) o la tesis (8 créditos).

[1]Es de anotar que el número de créditos es alto para lo que a nivel internacional se considera formación de maestría, que normalmente oscila entre los 30 y 34 créditos para la modalidad profesionalizante y hasta 40 créditos para la de investigación.

Área de conocimiento	Créditos
Ética y conducta profesional	
1. Aspectos sociales, legales e históricos.	**0.5**
2. Código de ética y conducta profesional.	**0.5**
3. Una revisión a los estándares relevantes.	**0.2**
SUBTOTAL CRÉDITOS	**1.2**
Una revisión conceptual a la Ingeniería de Sistemas	
1. Contexto de sistema, relación de seres humanos y sistemas, jerarquía de sistemas.	**0.5**
2. Ciclo de vida en la Ingeniería de Sistemas: Fases, actividades y tareas.	***0.5***
3. Relación del ciclo de vida con los procesos en la Ingeniería de Software.	**0.2**
SUBTOTAL CRÉDITOS	**1.2**
Ingeniería de requerimientos	
1. Técnicas para la determinación de requerimientos (Entrevistas, escenarios, prototipos, encuentros facilitadores, observación, historias de usuario).	**0.5**
2. Análisis de requerimientos (Clasificación, modelado conceptual, análisis formal).	**0.5**
3. Especificación de requerimientos (Especificación del sistema, especificación de requerimientos del sistema, especificación de requerimientos del software).	**0.2**
SUBTOTAL CRÉDITOS	**1.2**
Diseño de software	
1. Modelado funcional, estructural y comportamental de software usando un metamodelo en el marco del paradigma de computación orientado a objetos (Semántica, notación y elaboración de modelos con un ejercicio práctico).	**2**
2. Diferenciación de metamodelo y meta-metamodelo.	**0.5**
3. Mapeo de modelos a código fuente (Codificación de modelos, documentación técnica de software en la etapa de diseño, RTF's para la etapa de diseño).	**1**
4. Diseño básico de interfaz gráfica de usuario.	**0.5**
SUBTOTAL CRÉDITOS	**4**
Construcción de software	
1. Gestión de la construcción de software (Métodos, planeación, medición y estimación).	**0.8**
2. Estándares de programación, pruebas unitarias, pruebas de integración.	**0.8**
3. Revisiones de calidad de la codificación.	**0.4**
SUBTOTAL CRÉDITOS	**2**
Pruebas de software	
1. Niveles de pruebas de software (Objetivos, pruebas de componentes, pruebas de sistema y pruebas de aceptación).	**0.4**
2. Técnicas de prueba (Basadas en experiencia de probadores, basadas en especificación, basadas en codificación, basadas en fallas, basadas en uso, basadas en la naturaleza del software, criterios de selección y combinación de pruebas).	**0.8**
3. Gestión y medición del proceso de pruebas.	**0.8**
SUBTOTAL CRÉDITOS	**2**
Mantenimiento y evolución del de software	
1. Aspectos técnicos, de gestión y de estimación de costos.	**0.5**
2. Actividades para el mantenimiento de software.	**0.5**

Área de conocimiento	Créditos
3. Técnicas de mantenimiento (Comprensión de programas, ingeniería inversa, re-ingeniería .	**0.2**
SUBTOTAL CRÉDITOS	**1.2**
Gestión de configuración de software	
1. Identificación de elementos de configuración (Ítems de configuración, versionado, línea base y bibliotecas de trabajo, librerías de software).	**0.5**
2. Control de elementos de configuración (Aprobación de cambios y control de versiones).	**0.5**
3. Herramientas para apoyar sistemas de gestión documental y de control de configuración.	**0.2**
SUBTOTAL CRÉDITOS	**1.2**
Gestión de proyectos de software	
1. Estimación de costos, agenda y esfuerzo.	**0.8**
2. Plan presupuestal y organización de personal de proyecto.	**0.8**
3. Fundamentos de gestión de riesgo.	**0.4**
SUBTOTAL CRÉDITOS	**2**
Procesos de software	
1. Modelos de proceso prescriptivos y ágiles.	**1**
2. Medición de producto y proceso.	**1**
3. Estimación ajustada a modelo de proceso genérico.	**0.4**
SUBTOTAL CRÉDITOS	**2.4**
Aspectos de calidad del software	
1. Calidad de proceso y calidad de producto.	**0.5**
2. Verificación de software para apoyar la calidad.	**0.5**
3. Gestión de la calidad del software.	**0.6**
SUBTOTAL CRÉDITOS	**1.6**
NÚMERO TOTAL DE CRÉDITOS	**20**

Tabla 5.1. Contenidos temáticos mínimos para el nivel de formación de cursos fundamentales.

5.3 Propuesta de posibles cursos de formación específica

La propuesta de cursos se enmarca en las temáticas de actualidad y se presenta en la Tabla 5.2. Estos cursos se ofertan como una bolsa opcional de créditos que el estudiante escoge de acuerdo con sus intereses profesionalizantes o de investigación bajo la orientación de su director de proyecto de grado o de tesis. La bolsa de créditos o cursos presentados en la Tabla 5.2, que está en consonancia con el estado del arte internacional temático del cuerpo de conocimiento de la Ingeniería de Software [29], es solo una propuesta que debe evaluarse por parte de cada programa académico para determinar cuáles pueden ser ofertados desde las capacidades académicas e investigativas de la facultad de ingeniería en cada universidad. Adicionalmente, quedaría la tarea de armonizar el alcance temático e intensidad de trabajo en horas con el número de créditos aquí recomendado.

POSIBLES CURSOS DE FORMACIÓN ESPECÍFICA	CRÉDITOS
Métodos formales para arquitecturas basadas en componentes.	4
Ingeniería de software dirigida por modelos.	4
Teoría de concurrencia y distribución.	4
Desarrollo de software basado en componentes.	4
Patrones de diseño y arquitecturas de software.	4
Bases de datos avanzadas (Relacionales y NoSQL).	4
Computación distribuida y paralela.	4
Formulación de lenguajes y semánticas de lenguajes de programación.	4
Factores humanos en el desarrollo de Interfaz Gráfica de Usuarios.	4
Verificación y validación de software.	4
Métodos formales en la Ingeniería de Software.	4
Tecnologías y herramientas para la construcción de software.	4
Diseño de software conducido por aspectos de calidad	4
Consideraciones teóricas y prácticas de la Ingeniería de Requerimientos	3
Medición y estimación en Ingeniería de Software	3
Desarrollo de software para sistemas de tiempo real	4
Técnicas de programación concurrente y lenguajes de implementación	4

Tabla 5.2. Bolsa opcional de cursos para el segundo nivel de formación.

5.4 Propuesta de posibles cursos electivos

Los cursos electivos contienen los de orientación complementaria y otras opciones generadas desde los grupos de investigación o profesores adscritos al programa académico. La Tabla 5.3 presenta una primera propuesta de bolsa de cursos opcionales con sus pesos en créditos; el estudiante escogerá de esta bolsa de créditos aquellos cursos útiles para sus propósitos profesionalizantes o de investigación bajo la orientación de su director de proyecto de grado o de tesis. La bolsa de créditos o cursos presentados en la Tabla 5.3 es solo una propuesta que debe actualizarse según las capacidades académicas e investigativas de la universidad o IES. Queda la tarea de armonizar su alcance temático e intensidad de trabajo en horas con el número de créditos aquí recomendado.

MODALIDAD	POSIBLES CURSOS ELECTIVOS
INVESTIGACIÓN	• Cienciometría, bibliometría y revisiones sistemáticas de literatura (4 créditos). • Redacción científica I (2 créditos). • Redacción científica II (2 créditos). • Herramientas para el análisis de datos (4 créditos). • Chequeo de modelos (4 créditos). • Redes de Petri para chequeo formal y prototipado en el desarrollo de software (4 créditos). • Álgebras de proceso para el modelado formal (4 créditos). • Meta-metamodelado de UML (4 créditos). • Computación evolutiva y arquitecturas de software basadas en componentes (Arquitecturas de software emergentes) (4 créditos). • Teoría de la complejidad e ingeniería (4 créditos).
PROFUNDIZACIÓN	• Herramientas para el análisis de datos (4 créditos). • Desarrollo de software especializado (4 créditos). • Diseño de software conducido por aspectos de calidad (4 créditos). • Documentación de arquitecturas de software (4 créditos). • Minería de datos y "Machine Learning" (4 créditos). • Desarrollo y mantenimiento de software seguro (2 créditos). • Evolución y mantenimiento de software (4 créditos). • Práctica profesional de la Ingeniería de Software (2 créditos). • Fundamentos de gestión económica de proyectos de Ingeniería de Software (2 créditos).

Tabla 5.3. Bolsa opcional de cursos en las dos modalidades para el tercer nivel de formación.

5.5 Mapeo de la propuesta a un plan de estudios actual

La propuesta presentada en los capítulos anteriores puede traducirse o mapearse a la propuesta de un plan de estudios actual. Las Figuras 5.3 y 5.4 se corresponden con la estructura del plan de estudios para la modalidad de investigación y profundización de la Maestría en Ciencias de la Información y las Comunicaciones de la Universidad Distrital Francisco José de Caldas, la cual posee un énfasis en Ingeniería de Software. Como se puede observar en la arquitectura del currículo presentada al comienzo de este capítulo, un 50% del plan de estudios es flexible. Para un programa de maestría de cuarenta créditos (excluyendo los créditos del proyecto final o tesis) se ocuparían 20 créditos en lo correspondiente

a cursos fundamentales, 12 créditos a la formación específica y 8 créditos a los cursos electivos.

Para el caso de la modalidad de investigación y profundización, se puede abordar el enfoque de formación temática que esta propuesta presenta usando dos cursos del núcleo común y tres cursos de profundización, la sumatoria de créditos (20) coincide con lo requerido para esta formación fundamental. Los demás cursos pueden tomarse por parte de los estudiantes, para el nivel de profundización y de electivas en cada modalidad, de la bolsa de créditos y asignaturas ofertadas por parte del programa académico (ver Tablas 5.2 y 5.3). La selección de dichos cursos se hará de acuerdo con las necesidades de formación que los tutores de proyecto de grado o tesis requieren de los estudiantes para llevar a buen término su formación posgradual.

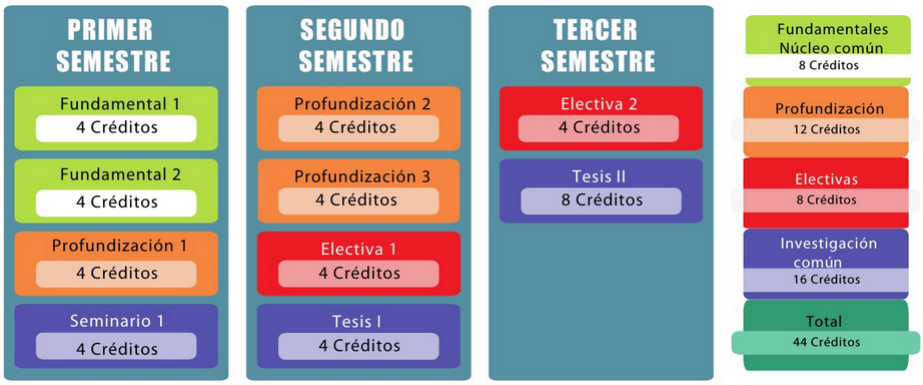


Figura 5.3. Plan de estudios de la modalidad de investigación.
Fuente [41].

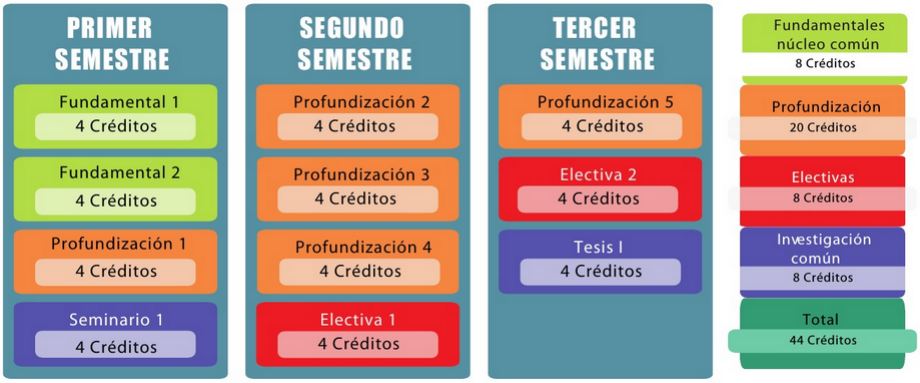


Figura 5.4. Plan de estudios de la modalidad de profundización.
Fuente [41].

Capítulo 6

Reflexiones finales

Algunas recomendaciones, fruto de la experiencia, ameritan ser consideradas:

- La Figura 6.1 presenta una cronología de documentos relacionados con reflexiones sobre el currículo en programas académicos de Ingeniería de Software en una ventana de 25 años; los documentos y publicaciones escogidas han influenciado de alguna manera la propuesta aquí desarrollada. Esta es solo una muestra de las publicaciones y reportes de trabajo que sobre este asunto se pueden encontrar en las diversas librerías digitales del mundo y en los repositorios documentales de centros e institutos de investigación.

  La intencionalidad de esta línea de tiempo es efectuar un corto análisis sobre el proceso de maduración de la disciplina. Como se puede observar, aparecen los títulos de los documentos en dos colores, se diferencian con color azul aquellas reflexiones y aportes que son de individuos o entidades particulares respecto a los documentos generados por organizaciones o consorcios de universidades, centros de investigación y empresas (color rojo). La muestra presentada nos permite visualizar que la década del noventa se dedicó principalmente a generar reflexiones individuales sobre la naturaleza de la disciplina y sobre cómo educar en Ingeniería de Sofware. Una de las universidades que ha aportado sustancialmente a estas reflexiones es Carnegie Mellon University con su Instituto de Ingeniería de Software.

  Para la primera década del siglo XXI, se evidencian los primeros esfuerzos consorciados en definir claramente el cuerpo de conocimiento de la disciplina y se establece la diferenciación de la Ingeniería de Software respecto a las otras disciplinas relacionadas. Estas circunstancias consolidan la disciplina de la Ingeniería de Software con un cuerpo de conocimiento definido y permite acordar unos mínimos de formación en programas pregraduales y posgraduales para los profesionales que soportarán en la sociedad su aplicación y desarrollo. Uno de los aspectos más útiles en este

tipo de reflexiones soportadas en organizaciones reconocidas internacionalmente es la revisión que se efectúa a diversos programas pregraduales y posgraduales en diversas partes del mundo, mientras que otros trabajos de revisión como el efectuado por Pyster et al. [43] aportan una visión crítica al cuerpo de conocimiento formulado. Estos aportes dan un sustento práctico, conceptual y de estado del arte que para la formulación de programas académicos pregraduales y posgraduales son pertinentes.

Esta última década se ha logrado refinar el trabajo de los consorcios como iSSEc y las fuerzas de tarea de la IEEE Computer Society así como de la ACM[1] junto con los consorcios europeos y asiáticos. Este trabajo conjunto en la definición de guías y soportes curriculares es bondadoso para la comunidad académica e industrial de ingenieros de software. Estos acuerdos conceptuales y del cuerpo de conocimiento que soporta la disciplina son tremendamente útiles y no deben obviarse en cualquier formulación de programas académicos relacionados con la disciplina.

Los marcos de trabajo establecidos por los documentos técnicos que han formulado guías para el currículo y cuerpos de conocimiento en las disciplinas pueden contextualizarse a las necesidades e intereses propios de cada universidad. El documento conceptual que aquí se presenta ha abordado esta tarea y facilitará a futuro mantener una línea base a partir de la cual la formulación de un programa académico en Ingeniería de Software evolucione de manera reflexivamente ordenada y acorde con el contexto internacional.

- Otro soporte importante para el desarrollo de programas posgraduales es el ambiente institucional, la infraestructura y logística que acompaña el desarrollo cotidiano de estos. Aunque en el capítulo 3 se enunciaron los principios institucionales y pedagógicos que la universidad debe garantizar, no se hizo hincapié en algunos aspectos de infraestructura necesarios y que se presentan a continuación:

  - Laboratorio de Ingeniería de Software para apoyar la experimentación directa con herramientas y el trabajo cotidiano de I + D + i de los estudiantes y profesores.
  - Se debe contar con una unidad de apoyo a la divulgación académica en idiomas foráneos. Inicialmente, esta unidad de apoyo puede concentrarse en el idioma inglés y debe servir para soportar la revisión y corrección de artículos y documentos técnicos generados en dicho idioma por parte de estudiantes y profesores. Adicionalmente, esta unidad de apoyo debe encargarse de diseñar y ejecutar los cursos complementarios de redacción académico científica en inglés u otros idiomas y apoyar el proceso de publicación.
  - Salas de reuniones y oficinas para estudiantes de tiempo completo (investigación).

[1] Association for Computing Machinery.

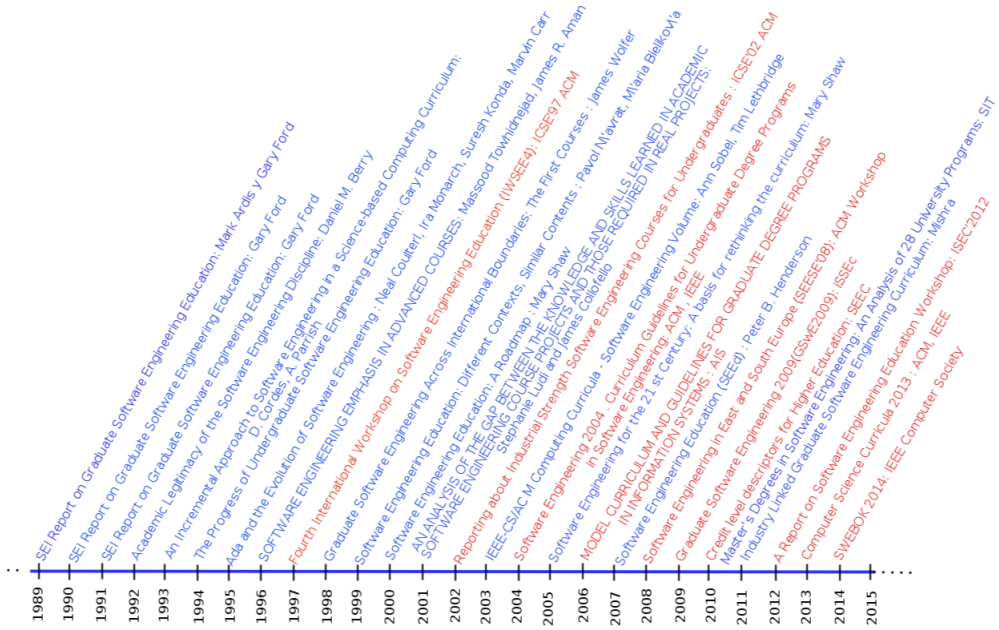


Figura 6.1. Cronología de publicaciones sobre cuerpo de conocimiento de la Ingeniería de Software.

- Bases de datos científicas con acceso ilimitado. Se proponen las siguientes: *Scopus, ACM Digital Library, Science Direct, IEEE Digital Library y Springer Link*, por lo menos.
- Subsidio imparcial y periódico de estadías académicas de profesores en institutos de investigación de universidades de prestigio, con el fin de actualizar sus conocimientos, fortalecer relaciones académico-científicas o tomar cursos especializados de interés para los grupos de investigación adscritos al programa académico.
- Subsidio imparcial y periódico para la asistencia de los profesores adscritos al programa a congresos internacionales relacionados con las temáticas de investigación que estén trabajando.
- Soporte con recursos de comunicación virtual para el desarrollo de los cursos propios del programa académico.
- Programa de apoyo con becas de tiempo completo para estudiantes de la modalidad de investigación.
- Programa de apoyo a la movilidad estudiantil para estudiantes pasantes en otras universidades y de académicos visitantes a nuestra universidad.
- Programa de bienestar institucional diferenciado para estudiantes posgraduales.

- Mantener la calidad en un programa académico requiere un compromiso serio, honesto y ético de la comunidad adscrita a este. La existencia de mecanismos de participación reflexiva y control argumentado para la incorporación de cambios al plan de estudios o procesos que afectan la calidad académica permite que la identidad con la institucionalidad de quienes participan en la cotidianidad del trabajo se fortalezca. Por esta razón, se propone que programas académicos posgraduales de esta naturaleza cuenten con Comités Académico-Científicos (CAC) legalmente establecidos. La labor principal de estos entes organizacionales será recomendar cambios e incorporación de temáticas a los cursos del programa académico. Así mismo, podrán reprobar propuestas de cambios al currículo con posturas argumentadas que darán soporte al Consejo de Programa o de Carrera para la legalización de las decisiones tomadas con base en las recomendaciones del CAC. Los integrantes del CAC podrían ser los directores de los grupos de investigación que brinden soporte al énfasis.

Referencias

[1] S. Zubiría. *Universidad, cultura y emancipación en América Latina*. Ediciones Izquierda Viva.Fundación Walter Benjamin, Bogotá D.C., 2013.

[2] El Tiempo. "crece la industria de 'software' colombiano". [En línea]. Disponible en: http://www.eltiempo.com/tecnosfera/novedades-tecnologia/aumento-de-la-industria-de-software-colombiano/15445677, Julio 2015.

[3] FEDESOFT. "FEDESOFT, busca generar estrategias para disminuir el déficit en profesionales de la Industria de Software del País". [En línea]. Disponible en: http://fedesoft.org/fedesoft-busca-generar-estrategias-para-disminuir-el-deficit\-en-profesionales-de-la-industria-de-software-del-pais/, Febrero 2015.

[4] La República. "El potencial del software colombiano en el mundo". [En línea]. Disponible en: http://www.larepublica.co/el-potencial-del-software-colombiano-en-el-mundo_228816, Marzo 2015.

[5] Gartner. "Gartner Says Worldwide Software Market Grew 4.8 Percent in 2013". [En línea]. Disponible en: http://www.gartner.com/newsroom/id/2696317, Marzo 2014.

[6] FEDESOFT. "Aliado Tecnológico vs Precio, factores determinantes al comprar un software". [En línea]. Disponible en: http://fedesoft.org/aliado-tecnologico-vs-precio-factores-determinantes-al-comprar\-un-software/, Febrero 2015.

[7] Integrated Software & Systems Engineering Curriculum (iSSEc) Project. Curriculum Guidelines for Graduate Degree Programs in Software Engineering. Technical report, Stevens Institute of Technology, 2009.

[8] Ministerio de Educación Nacional. Resolución No. 05485 del MEN. [En línea]. Disponible en: http://www.udistrital.edu.co:8080/en/web/mcic/, Abril 2015.

[9] Joint Task Force for Computing Curricula 2004. A Guide to Undergraduate Degree Programs in Computing. Technical report, *The Association for Computing (ACM), The Association for Information Systems (AIS), The Computer Society (IEEE-CS)*, 2004.

[10] The Joint Task Force on Computing Curricula. Computer Engineering 2004.Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering. Final Report. Technical report, *IEEE Computer Society and Association for Computing Machinery*, Diciembre 2004.

[11] T. Hastie, R. Tibshirani y J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* Springer, second edition edition, 2009.

[12] J. Tucker y A. Mackness. *Human-Centered System Development*, volume Computer Science Handbook, chapter 47, pages 1–26. CRC Press, second edition, 2004.

[13] C. Herbert, J. Wang y J. Liu. *Information Retrieval and Data Mining*, volume Computer Science Handbook, chapter 75, pages 1–16. CRC Press, second edition, 2004.

[14] A. Silberschatz, H. Korth S. Sudarshan. *Data Models*, volume Computer Science Handbook, chapter 52, pages 1–20. CRC Press, second edition, 2004.

[15] S. Bellovin. *Network and Internet Security*, volume Computer Science Handbook, chapter 74, pages 1–14. CRC Press, second edition, 2004.

[16] D. House. *Overview of Three-Dimensional Computer Graphics*, volume Computer Science Handbook, chapter 35, pages 1–19. CRC Press, second edition, 2004.

[17] E. Allender, M. Loui y K. Regan. *Complexity Theory*, volume Computer Science Handbook, chapter 5, pages 1–30. CRC Press, second edition, 2004.

[18] T. Jiang, M. Li y B. Ravikumar. *Formal models and computability*, volume Computer Science Handbook, chapter 6, pages 1–34. CRC Press, second edition, 2004.

[19] D. Kopec, T. Marsland y J.L. Cox. *Search*, volume Computer Science Handbook, chapter 63, pages 1–26. CRC Press, second edition, 2004.

[20] The Joint Task Force on Computing Curricula. Computer Science Curricula 2013. Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. Final Report. Technical report, *IEEE Computer Society and Association for Computing Machinery*, Diciembre 2013.

[21] H. Topi, J. Valacich, R. Wright y Kate M. et al. IS 2010. Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. Technical report, *Association for Computing Machinery and Association for Information Systems*, Noviembre 2010.

[22] B. Lunt, J. Ekstrom, S. Gorka y G. Hislop et al. Information Technology 2008. Curriculum Guidelines for Undergraduate Degree Programs in Information Technology. Technical report, *Association for Computing Machinery and IEEE Computer Society*, Noviembre 2008.

[23] IEEE Computer Society. IEEE Standard Glossary of Software Engineering Terminology. [En línea]. Disponible en: `http://www.computer.org/sevocab`, Julio 2015.

[24] F. Bauer. Software Engineering. *Information Processing*, 71, 1972.

[25] 1990 SEI Report on Undergraduate Software Engineering Education. Technical report, *Software Engineering Institute. Carnegie Mellon University*, 1990.

[26] R. Pressman. *Ingeniería del software. Un enfoque práctico.* McGraw Hill, 1221 Avenue of the Americas, New York. NY 10020, Seventh edition, 2010.

[27] Software Engineering for the 21st Century: A basis for rethinking the curriculum. Technical report, *Software Engineering Institute. Carnegie Mellon University*, 2005.

[28] The Joint Task Force on Computing Curricula. Software Engineering 2004. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Technical report, *IEEE Computer Society and Association for Computing Machinery*, Agosto 2004.

[29] Guide to the Software Engineering Body of Knowledge. Version 3.0, SWEBOK. Technical report, *IEEE Computer Society*, 2014.

[30] "Job Outlook: The Candidate Skills/Qualities Employers Want, the Influence of Attributes". [En línea]. Disponible en: `http://www.naceweb.org/s11122014/job-outlook-skills-qualities-employers-want.aspx`, November 2014.

[31] 1991 SEI Report on Graduate Software Engineering Education. Technical report, *Software Engineering Institute. Carnegie Mellon University*, 1991.

[32] O. Eljabiri y F. Deek. *Specialized System Development*, volume Computer Science Handbook, chapter 110, pages 1–18. CRC Press, second edition, 2004.

[33] B. Kitchenham. Procedures for Performing Systematic Reviews. Technical report, Software Engineering Group Department of Computer Science. Keele University, 2004.

[34] Members of Southern England Consortium for Credit Accumulation and Transfer. SEEC Credit level descriptors for Higher Education. Technical report, Southern England Consortium for Credit Accumulation and Transfer, 2010.

[35] B. Bloom. *Taxonomy of educational objectives: The classification of educational goals: Handbook I,cognitive domain*. Longmans, New York: David McKay Company., 1956.

[36] Bloom et al.'s taxonomy of the cognitive domain. [En línea]. Disponible en: `http://www.edpsycinteractive.org/topics/cogsys/bloom.html`, 2011.

[37] "Developing a Computer Science-specific Learning Taxonomy". [En línea]. Disponible en: `https://kar.kent.ac.uk/23997/1/TaxonomyFuller.pdf`, Junio 2014.

[38] M. Azuma, F. Coallier y J. Garbajosa. How to Apply the Bloom Taxonomy to Software Engineering. IEEE. Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP'04), 2004.

[39] D. R. Krathwohl. A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice*, 41(4):212–218, 2002.

[40] H. A., Diosa. ¿Cómo enseñar a diseñar software?.[Relación con los niveles de abstracción en ingeniería]. En: *Experiencias educativas y prácticas pedagógicas en la Universidad Distrital. Memorias del Primer Encuentro*, 2013.

[41] Subcomité de autoevaluación y acreditación. Proyecto curricular. Maestría en Ciencias de la Información y las Comunicaciones. Documento maestro, 2014.

[42] Subcomité de autoevaluación y acreditación. Maestría en Ciencias de la Información y las Comunicaciones.Pénsum 2015-III. Programa académico por modalidad.[En línea]. Disponible en: http://www1.udistrital.edu.co:8080/web/mcic/plan-nuevo-2015, 2018.

[43] A. Pyster, R. Turner, H. Devanandham y K. Lasfer et al. Master's Degrees in Software Engineering: An Analysis of 28 University Programs. *IEEE Software. Software Education Engineering*, pages 94–101, sept-oct 2009.

Autor

Ingeniero de Sistemas de la Universidad Nacional de Colombia; magíster en Teleinformática de la Universidad Distrital Francisco José de Caldas, y doctor en Ingeniería con énfasis en Ciencias de la Computación de la Universidad del Valle. Actualmente se desempeña como docente de planta de la Facultad de Ingeniería de la Universidad Distrital y es director del Grupo de Investigación en Arquitecturas de Software (ARQUISOFT). Para conocer más de los proyectos y actividades del autor en el ámbito académico se puede visitar el portal `http://arquisoft.udistrital.edu.co`.