Res. CFE Nro. 129/11

Anexo V

Marco de referencia para procesos de homologación de títulos del nivel superior

---

Sector Informático - Desarrollo de Software

Índice

1. Identificación del título profesional y trayectoria formativa

1.1 Sector/es de actividad socio productiva: Informática[1] (Software y servicios informáticos)

1.2 Denominación del perfil profesional: Desarrollador de Software

1.3 Familia profesional: Informática

1.4 Denominación del título: Técnico Superior en Desarrollo de Software

1.5 Nivel y ámbito de la trayectoria formativa: Nivel Superior en la modalidad Técnica.

2. Referencial al Perfil Profesional[2]

2.1. Alcance del Perfil Profesional.

> El Técnico Superior en Desarrollo de Software estará capacitado para producir artefactos de software[3], lo que comprende su diseño detallado, construcción -reutilizando elementos existentes o programándolos enteramente- y verificación unitaria, así como su depuración, optimización y mantenimiento; desarrollando las actividades descriptas en el perfil profesional y cumpliendo con los criterios de realización establecidos para las mismas en el marco de un equipo de trabajo organizado por proyecto.

El proceso de desarrollo de software es una tarea grupal, o también individual y muchas veces multidisciplinaria que se organiza por proyectos. Cada proyecto es negociado y acordado con el cliente o usuario y llevado a cabo por un equipo de trabajo constituido "ad-hoc", conducido y administrado por un líder que mantiene la relación diaria con el cliente o usuario y asume la responsabilidad operativa del proyecto.

El software debe satisfacer especificaciones de requerimientos, ya sean éstas formales o informales, las que pueden venir dadas por el cliente, algún consultor especializado en el tipo de problemas que aborda la aplicación o ser elaboradas por algún analista funcional integrante del equipo de trabajo del proyecto. El equipo de desarrollo suele estar integrado por un arquitecto de software, que establece el diseño general del sistema y especificaciones de calidad de la solución, un grupo de desarrolladores de software, que son quienes lo construyen y otro de "testing", que son los encargados de verificar que el software producido cumpla los requisitos, tanto funcionales como de comportamiento, oportunamente establecidos. Del equipo de trabajo pueden participar uno o más analistas técnicos que se ocupan de detalles relativos a aspectos de tecnología, seguridad, bases de datos o estándares de programación y asesoran y dan apoyo técnico a los desarrolladores. Eventualmente pueden participar diseñadores gráficos y especialistas en otros aspectos específicos.

A partir de especificaciones de diseño y del conocimiento de la arquitectura del sistema, los desarrolladores de software (también denominados analistas programadores o programadores) diseñan en forma detallada la parte del software que les correspondiere, la construyen, preferiblemente en base a artefactos de software ya existentes y adaptando o escribiendo lo que sea necesario, así como documentándola para facilitar su posterior mantenimiento por otros, verifican unitariamente lo producido y lo entregan para ser probado integralmente e integrado al resto. Habitualmente, los desarrolladores, que pueden estar especializados en una tecnología determinada, trabajan individualmente o de a pares dentro de un grupo más numeroso, brindándose mutuamente colaboración para resolver los problemas que deben enfrentar y los que tienen mayor experiencia suelen brindar orientación (coaching) a los más noveles.

En algunos casos, sobre todo en lugares en los que organizaciones de escasa dimensión y recursos no necesitan de software sofisticado o no pueden plantear requisitos de calidad para el software que necesitan, el desarrollo de software suele ser efectuado por realizadores independientes que asumen todas las funciones del equipo de desarrollo.

---

[1] Este técnico desempeña sus actividades en proyectos de desarrollo de software. La instrumentación del Catálogo Nacional de Títulos (Ley 26.058) determinará la denominación final del sector en el cual esta formación deba incluirse.

[2] Refiere al perfil profesional desarrollado por el PET del INET a partir de una tarea de análisis ocupacional realizado en colaboración con cámaras, polos y asociaciones del sector y validado por sus representantes.

[3] **Artefacto de software**: cualquier parte de software (es decir modelos/descripciones) desarrollado y utilizado durante el desarrollo y mantenimiento de software. Ejemplos de artefactos son especificaciones de requerimientos, modelos de arquitectura y de diseño, código fuente y ejecutable (programas), instrucciones de configuración, datos de prueba, scripts de prueba, modelos de proceso, planes de proyecto, otra documentación pertinente. [Según el glosario sobre Ingenería de Software mantenido por el Prof. Raider Corradi, del Dpto. de Informática, Universidad Noruega de Ciencia y Tecnología, Trondheim]

El Técnico Superior en Desarrollo de Software participa en proyectos de desarrollo de software desempeñando roles que tienen por objeto producir artefactos de software (programas, módulos, objetos). Estos artefactos suelen integrarse en aplicaciones o subsistemas que interactúan entre sí, con otras aplicaciones ya existentes desarrolladas con la misma o distinta tecnología, con el sistema operativo del computador u otro software de base (motor de base de datos, navegador, monitor de comunicaciones) configurando distintas capas de software que pueden estar distribuidas en diversas máquinas situadas en la misma o distintas ubicaciones.

La actividad del desarrollador de software es no rutinaria a pesar de que muchas veces se reutilicen partes ya existentes. Cada asignación representa la necesidad de dar satisfacción a determinados requisitos. Ello requiere comprender el problema y la arquitectura en la que estará inserta la solución, idear estrategias de resolución y dominar el lenguaje y ambiente de programación a emplear, así como aplicar buenas prácticas de programación, lo que incluye documentar decisiones significativas de diseño y las limitaciones que tendrá el artefacto construido.

Para poder desarrollar plenamente su profesionalidad, el técnico tiene que poseer ciertas capacidades que resultan transversales a todas sus funciones y tienen que ser desarrolladas durante el transcurso de su formación. Estas son:

Abstracción - Implica descartar o reducir detalles poco significativos de la información sobre un problema para concentrarse en pocos elementos por vez, lo que resulta en una reducción de la complejidad que permita conceptualizar de modo más simple un dominio de problemas para facilitar su comprensión y manejo en forma genérica de sus posibles soluciones.

Pensamiento combinatorio - Conduce a la consideración sistemática de un conjunto de alternativas, lo que incluye el manejo mental de muchas variables o detalles del problema sin perder nunca de vista el concepto o la estrategia general de resolución.

Autorregulación - Implica manejarse respetando reglas y limitaciones, tanto explícitas como implícitas, sean éstas propias o del equipo de trabajo; actuar ateniéndose a un orden propio que le facilite el acceso a lo que puede necesitar, reconocer y guardar; referenciar la información y registrarla de tal manera que le facilite acceder posteriormente en forma rápida para evaluarla y recuperarla.

Comunicación apropiada - Implica una disposición a reconocer que existen otros que pueden aportar información útil o a quienes puede interesarle lo que hace. Supone reconocer su rol y el de cada integrante del proyecto, transmitir la información necesaria en forma precisa y en un lenguaje apropiado para el entendimiento mutuo en interacciones individuales o grupales, o en forma escrita, utilizando, si es necesario para ello, el idioma inglés, que debe interpretar con propiedad a nivel técnico.

Trabajo en equipo - Implica adoptar una actitud abierta, estar dispuesto a compartir información y conocimientos, a tomar en cuenta a los usuarios del producto que está construyendo, a brindar, pedir y aceptar ayuda cuando ésta resulte necesaria para facilitar su propia labor o la de otro integrante del equipo. Comprende al equipo del proyecto, incluyendo a los usuarios que participan del mismo.

Además, se requiere:

Actitud de aprendizaje permanente - Implica aprender a capitalizar experiencias a partir de su propio trabajo, a tomar iniciativas para actualizar o profundizar sus conocimientos y habilidades, investigar fuentes de información o herramientas que le puedan ser útiles. Aplica metodologías de investigación y dedica tiempo a este fin.

Actitud ética - Implica el ejercicio profesional respetando principios éticos y adecuación al marco legal, como así también conocer y aplicar la normativa legal vigente.

2.2. Funciones que ejerce el profesional

A continuación se presentan funciones y subfunciones del perfil profesional de este técnico superior en las cuales se pueden identificar las siguientes actividades:

Modelizar artefactos de software a partir de especificaciones, refinándolas en caso necesario, para determinar el diseño detallado y las características de una solución que las satisfaga en el contexto de la arquitectura del sistema de software del cual van a formar parte.

Esto comprende:

Interpretar críticamente las especificaciones recibidas.

Interpretar la arquitectura del sistema en el cual se inserta la asignación.

Aplicar patrones de diseño si corresponde.

Diseñar la solución.

Representar el diseño.

Verificar el diseño.

Para realizar esto el técnico utiliza lenguajes y herramientas de representación y modelización de sistemas, como UML y otras técnicas de graficación y especificación, incluyendo diccionarios de datos del proyecto y catálogos de patrones de diseño. También considera las características de la tecnología a utilizar y consulta a pares y al líder del equipo de trabajo para interpretar los problemas a resolver y verificar sus conclusiones y enfoques. Al realizar esto procura atenerse a los lineamientos de la arquitectura establecida para el proyecto y respeta criterios de seguridad informática, confidencialidad y las políticas vigentes en la organización en la cual se desempeña, así como las prácticas establecidas para el proyecto.

Construir los artefactos de software que implementen el diseño realizado, aplicando patrones o reutilizando código en la medida en que resulte posible. Al hacer esto, y en función de lo acordado para el proyecto, optimizará el desempeño de lo que construya aplicando buenas prácticas de programación y documentación.

Esto comprende:

Reutilizar elementos ya existentes.

Redactar código.

Optimizar el código.

Controlar cambios y versiones.

Utilizar ambientes de desarrollo.

Para realizar esto el técnico utiliza patrones, reutiliza código existente adaptándolo o complementándolo a su nueva función o redacta código nuevo aplicando sus conocimientos de programación, respetando buenas prácticas y las normas establecidas para asegurar la calidad del proyecto. Esto implica el dominio del lenguaje y del ambiente de desarrollo utilizados en el proyecto, así como la tecnología en la cual va a ser implementada la solución. También consulta a pares y al líder del equipo de trabajo para reflexionar y recibir ayuda que le permita resolver problemas encontrados o aporta sus conocimientos y capacidad de reflexión a otros, y participa de foros y listas temáticas para encontrar soluciones o elementos reutilizables.

Verificar los artefactos de software construidos considerando las necesidades de cobertura de la prueba. Para ello diseña los casos considerando el entorno de pruebas y ejecuta pruebas unitarias, así como registra los datos y resultados. De ser necesario, realiza acciones correctivas sobre el código hasta satisfacerse de que cumpla con las especificaciones recibidas.

Esto comprende:

Considerar las necesidades de cobertura de la prueba.

Diseñar los casos de prueba.

Preparar el entorno de pruebas.

Realizar pruebas unitarias.

Registrar casos de prueba, datos y resultados de pruebas y acciones correctivas.

Para realizar esto el técnico determina las necesidades de cobertura en función de las características de su asignación y normas establecidas para asegurar la calidad del proyecto, identifica las clases de equivalencia de datos utilizados internamente o intercambiados y diseña los casos de prueba, tomando en cuenta la estructura del artefacto y las condiciones de borde, así como prepara el entorno de pruebas, incluyendo los scripts y datos necesarios. Esto implica el dominio de conceptos de

"testing" y de herramientas utilizadas para establecer el ambiente de "testing". Realiza las pruebas unitarias, registrando los datos y resultados alcanzados, así como las acciones correctivas realizadas para solucionar las fallas encontradas.

Revisar el código de artefactos de software para resolver defectos o mejorarlo. Este código puede ser propio o ajeno. Esta actividad comprende revisiones cruzadas con otros integrantes del proyecto para asegurar la calidad del producto. Algunas asignaciones requieren una revisión de código ya existente para poder ampliar funcionalidades o refactorizarlo.

Esto comprende:

Interpretar código.

Diagnosticar defectos.

Depurar defectos.

Al realizar esto el técnico analiza sistemáticamente el código para identificar partes relacionadas con posibles malfuncionamientos y revisa meticulosamente esas partes para determinar las causas de posibles defectos a fin de corregirlos, así como replantea, si resulta necesario, aspectos estructurales y cuida de no introducir otros defectos al efectuar modificaciones en el código. También analiza tanto el cumplimiento de buenas prácticas de programación, como la eficiencia del código.

Documentar sus actividades y los resultados obtenidos aportando elementos para asegurar la calidad de los proyectos de acuerdo a normas y estándares establecidos.

Registrar actividades realizadas.

Documentar todos los productos de su labor.

Las normas de calidad del proceso de desarrollo de software exigen una adecuada documentación del mismo, así como del producto resultante. Para que el técnico pueda realizar un aporte efectivo a estas exigencias de calidad y para facilitar el mantenimiento de lo que desarrolle, tiene que justificar las decisiones relevantes de diseño que tome, así como las limitaciones que tienen los artefactos que produzca, de acuerdo a criterios de legibilidad por parte de otros y a las normas de documentación establecidas para el proyecto.

Gestionar sus propias actividades dentro del equipo de trabajo del proyecto. Ello comprende la planificación (organización y control) de las tareas a realizar, el oportuno reporte de avances y dificultades y el registro y reflexión sobre lo realizado para capitalizar experiencias y estimar métricas aplicables a su actividad.

Obtener métricas a partir de los registros de actividades.

Reportar avances y dificultades.

Planificar sus actividades.

Controlar sus actividades.

La construcción de software es una actividad que se desarrolla por proyectos, los que son llevados a cabo por un equipo de trabajo y el técnico tiene que realizar un aporte efectivo al trabajo conjunto. Para ello tiene que mantener una comunicación efectiva con quien lidere el grupo o lo asesore y con el resto de su equipo de trabajo, informando y consultando sobre problemas que observe al enfrentar sus asignaciones. También debe desarrollar su propia profesionalidad estimando tiempos y comparando resultados, extrayendo conclusiones formales o informales que le permitan establecer sus propias métricas de rendimiento y calidad, así como un estrecho autocontrol que le facilite una mayor predictibilidad de sus resultados.

Para lograr un desempeño competente en sus actividades profesionales, el desarrollador de software, además de realizar las actividades previstas en su perfil profesional e incluidas aquí en la descripción de las funciones que realiza, tiene que dominar ciertos aspectos de la tecnología de la información que le sirven de base para poder desarrollar competentemente sus funciones profesionales. Al dominio de estos aspectos lo hemos denominado:

Desempeño de base – Esto implica conocer y saber utilizar con propiedad y en condiciones de seguridad recursos de hardware, software y redes para emplear los ambientes que necesite para el

desarrollo y la verificación del software, mantener los repositorios de información que necesite utilizar y disponer de los productos de su trabajo en condiciones de confiabilidad.

2.3. Área Ocupacional

Este técnico se ocupa en organizaciones de diversos tipos. Empresas que realizan desarrollo de software por encargo de organizaciones locales o extranjeras, que proveen software junto con otros servicios de asesoramiento y consultoría, y, en menor número, que desarrollan sus propios productos de software para vender en el país o en el exterior. También en organizaciones dedicadas a otras actividades, pero que producen el software que necesitan para desarrollar sus propias actividades o que integran en productos que venden.

Su posición ocupacional suele denominarse analista programador o programador, aunque últimamente se está generalizando una denominación más abarcativa y menos categorizante, desarrollador de software. Integra equipos de proyecto dedicados al desarrollo o mantenimiento de software y recibe asignaciones específicas que tiene que resolver en lapsos que suelen medirse en términos de días o semanas, produciendo artefactos que satisfagan especificaciones y se integren al sistema objeto del proyecto.

Resuelve estas asignaciones individualmente o trabajando en pares, recibiendo la supervisión y asesoramiento de un líder de proyecto o de grupo, con quien consulta dudas y decisiones significativas o comunica inconvenientes. También recibe apoyo y brinda colaboración a otros miembros del grupo. Su trabajo es verificado por un grupo de "testing" y eventuales controles cruzados de código importante. Con una mayor experiencia o especialización en determinadas tecnologías o metodologías, posibles evoluciones ocupacionales del Técnico Superior en Desarrollo de Software son el liderar grupos de trabajo o asumir roles de analista técnico en la materia de su especialidad.

Asimismo, puede desempeñarse en forma autónoma, asumiendo la mayor parte de las tareas propias del proceso, sobre todo trabajando en forma independiente resolviendo problemas de pequeñas organizaciones que requieren sistemas de baja complejidad y reducida dimensión. Por otra parte, Técnicos Superiores en Desarrollo de Software o profesionales equivalentes con capacidad emprendedora pueden y suelen asociarse entre ellos para generar sus propias empresas para brindar servicios de desarrollo y proveer software a terceros.

2.4. Habilitaciones profesionales

Las actividades que realiza y para las cuales está capacitado el Técnico Superior en Desarrollo de Software, así como el ámbito de su desempeño y el campo y condiciones de su ejercicio profesional son los descriptos en el Perfil Profesional correspondiente.

Si bien las actividades de este técnico superior no están orientadas a un tipo de software en particular, conviene tener en cuenta que el software es utilizado crecientemente en sistemas que afectan a la seguridad pública. Estos sistemas, denominados *críticos para la seguridad*, son lo que, en un sentido general, involucran riesgos que conllevan la posibilidad de pérdidas inaceptables (daños para la salud o aún la vida humana, daños a la propiedad, contaminación ambiental, conflictos sociales, grandes pérdidas monetarias).

En función de estos riesgos, se establecen las siguientes habilitaciones profesionales para el Técnico Superior en Desarrollo de Software, con las limitaciones o exclusiones que se indican en cada caso. Estas habilitaciones tienen efecto para su desempeño en forma autónoma o asumiendo plenamente la responsabilidad por los resultados que obtenga su grupo de trabajo.

- Diseñar, construir y verificar artefactos de software de complejidad media, correspondientes a sistemas de información o vinculados indirectamente al hardware o a sistemas de comunicación de datos, respondiendo a especificaciones.

Queda excluido de esta habilitación el software correspondiente a sistemas críticos para la seguridad, como es el caso de los que involucren el procesamiento de información que conlleve riesgos efectivos para terceros. Particularmente, queda excluido el software destinado a:

  - control de equipos y procesos médicos, industriales o de domótica que puedan poner en riesgo inmediato o mediato la salud de personas;

- procesamiento de información crítica para los individuos, como ser la que sirva para corroborar su identidad o características de su estado de salud, para demostrar situaciones legal, fiscal, patrimonial u otras que afecten a su patrimonio o a sus libertades;
- procesamiento en línea de transacciones financieras importantes.

En estos casos, requerirá la supervisión de profesionales habilitados.

- Controlar la calidad de artefactos de software para resolver defectos o mejorarlos, lo que incluye revisar especificaciones, diseños y código.

Esto se realiza bajo supervisión en el marco de equipos de desarrollo de software.

3. En relación con la Trayectoria Formativa

Los planes de estudio a ser presentados para su homologación deberán evidenciar el trayecto formativo completo que conduce a la emisión del título técnico superior, independientemente de la organización institucional y curricular adoptada, de manera tal que permitan identificar los distintos tipos de contenidos a los que hace referencia.

Deberán identificarse los campos de formación general, de formación de fundamento, de formación específica y de prácticas profesionalizantes.

De la totalidad de la trayectoria formativa y a los fines de homologar títulos de un mismo sector profesional y sus correspondientes ofertas formativas, que operan sobre una misma dimensión de ejercicio profesional, se prestará especial atención a los campos de formación de fundamento, de formación específica y de prácticas profesionalizantes. Cabe destacar que estos contenidos son necesarios e indispensables pero no suficientes para la formación integral.

3.1. Formación general

El campo de formación general está destinado a abordar los saberes que posibiliten la participación activa, reflexiva y crítica en los diversos ámbitos de la vida laboral y sociocultural y el desarrollo de una actitud ética respecto del continuo cambio tecnológico y social.

Este campo de la formación del técnico sirve de nexo entre la tecnología, el trabajo y el ciudadano. A los fines del proceso de homologación, aunque no se analizarán específicamente sus contenidos, este campo debe ser identificable en el plan de estudios a homologar y la carga horaria total de este campo tendrá que respetar la acordada para los títulos de educación técnica superior.

Ejemplo de contenidos que pueden formar parte de este campo son los relativos a la ética y la responsabilidad social del técnico; la problemática sociocultural del trabajo; la comunicación, incluyendo la comprensión y producción de textos; las técnicas de indagación destinadas tanto a la búsqueda de información y conocimientos como a interpretar requerimientos de terceros; el conocimiento básico de lenguas extranjeras, en particular el idioma inglés que es el lenguaje en que se documenta la tecnología de la información y permite acceder a la información necesaria. Estos ejemplos no pretenden ser completos ni excluyentes.

3.2. Formación de Fundamento

Está destinado a abordar los saberes científico-tecnológicos y socioculturales que otorgan sostén a los conocimientos, habilidades, destrezas, valores y actitudes propios del campo profesional.

Provenientes del campo de la Matemática y la Lógica.

Funciones; tipos: inyectivas, sobreyectivas, inversas, composición. Relaciones; tipos: reflexividad, simetría, transitividad, equivalencia. Conjuntos; diagramas de Venn, operaciones, complementos, producto cartesiano, conjunto potencia. Numerabilidad y cardinalidad. Aritmética modular. Relaciones de congruencia. Sistemas de numeración.

Elementos de lógica. Lógica proposicional, conectivos lógicos. Tablas de verdad. Formas normales; conjuntiva y disyuntiva. Validez. Lógica de predicados; cuantificadores universal y existencial. Modus ponens y modus tollens. Limitaciones de la lógica de predicados.

Técnicas de demostración. Nociones de implicación, conversa, inversa, contrapositivo, negación y contradicción. La estructura de las demostraciones matemáticas. Demostración directa, por

contraejemplo, por contradicción. Inducción matemática. Inducción fuerte. Definiciones matemáticas recursivas. Buen ordenamiento.

Bases de conteo. Argumentos de conteo, regla de la suma y el producto. Principio de inclusión – exclusión. Sucesiones aritméticas y geométricas, números de Fibonacci. Principio de Dirichlet. Permutaciones y combinaciones, propiedades de los números combinatorios. El teorema binomial. Resolución de relaciones de equivalencia, el teorema maestro.

Grafos. Conceptos básicos, recorridos, coloreado de vértices. Árboles y bosques. Grafos dirigidos y redes. Aplicaciones de árboles y grafos (algoritmos de recorrida, organización de índices, topología de redes, cálculo del camino crítico). Matrices y vectores como representación de cambios de estado.

Espacio finito de probabilidades, medida de probabilidad, eventos. Probabilidad condicional, independencia, teorema de Bayes. Esperanza matemática, variables aleatorias enteras. Ley de los grandes números.

Números reales. Funciones reales de una variable. Límite y continuidad. Cálculo diferencial. Aplicaciones del cálculo diferencial.

Estadística descriptiva. Medidas de posición y de dispersión. Efectos del descarte de datos extremos en las diferentes medidas de posición y dispersión. Distribuciones discretas y continuas. Acumulación por rangos. Recolección de datos para análisis estadísticos, clasificación e interpretación. Series temporales.

Nota: el propósito de los contenidos de este campo consiste en desarrollar capacidad de razonamiento y de resolución de problemas para fortalecer bases necesarias para el pensamiento computacional.

Provenientes del campo de la Tecnología de la Información

Conceptos de tecnología de la información, evolución histórica, tecnologías que la integran, disciplinas que forman parte (ciencia de la computación, ingeniería de software, sistemas de información, ingeniería en computación) o aportan a la misma. El problema de la complejidad. Concepto de computación paralela, concurrente, multinúcleos.

Evolución del computador, su organización y unidades funcionales que lo componen. Arquitectura interna de computadores, unidad central de procesamiento, instrucciones y flujo de la información Tipos y niveles de organización de la memoria interna y externa (sistemas de memoria, tecnologías y jerarquías, memoria caché, memoria virtual, dispositivos de almacenamiento secundario). Periféricos, clasificación y utilización. Funcionamiento del programa a nivel de la máquina (principalmente como medio de comprender características de su funcionamiento).

Introducción a la lógica digital, compuertas lógicas, flip-flops, circuitos. Expresiones lógicas y funciones booleanas. Representación de datos numéricos, aritmética con y sin signo, concepto de overflow. Rango, precisión y errores en aritmética de punto flotante. Representación de caracteres, audio e imágenes. Compresión de datos.

Orígenes y evolución de Internet y las comunicaciones digitales. Arquitecturas de red. Especializaciones en la computación centrada en redes. Redes y protocolos. Computación distribuida. Paradigmas cliente/servidor y peer to peer. Computación sin cables y móvil.

Estándares de redes y cuerpos de estandarización. Modelos de referencia: modelo de capas, TCP/IP. Espacio de direcciones del protocolo, categorías de direcciones. Máscara de red. Elementos de ruteo e interconexión. Aspectos de seguridad de redes.

Provenientes del campo de la Administración.

Elementos de teoría general de los sistemas, enfoque sistémico de la organización. Elementos de estructura y comportamiento de las organizaciones, organización estructurada por funciones o líneas de productos, el manejo de sedes.

Concepto de proceso. Procesos del ciclo de ventas y cobranzas; del ciclo de compras y pagos. Nociones de procesos de gestión y transformación de materiales y su organización. Comprobantes usuales, requerimientos legales y fiscales. Concepto de recurso y su gestión. El papel de los

sistemas de información en la organización. Nociones de control interno. La contabilidad como sistema de información. Algunas características de organizaciones y procesos de servicios.

Los niveles de la administración: la planificación estratégica, el control de gestión, el control operativo y el tipo de sistemas de información asociados a los mismos.

Provenientes del campo de la Ética y del Derecho.

Importancia social y económica de los servicios de tecnología de la información, significado de Internet, valor de la información almacenada para las organizaciones, seguridad. Valor de la información para los individuos, normativa relativa a privacidad y "habeas data". Bases de datos públicas y privadas. Propiedad de datos empresarios. Secretos comerciales e industriales.

Contexto normativo: responsabilidades empresarias, contratos, responsabilidades del trabajador, leyes de protección de datos personales, propiedad intelectual del software y de contenidos, conceptos jurídicos aplicables a delitos informáticos.

Privacidad de datos personales. Normas que rigen el correo electrónico. Protección legal de la propiedad intelectual (incluyendo software), derechos de reproducción y derechos sobre marcas y patentes. Licencias de fabricación, de uso, GNU y "creative commons".

Derechos y obligaciones derivados de relaciones laborales o profesionales. Derecho contractual y normas de ética profesional.

Provenientes del campo del Idioma Inglés.

Inglés técnico. Lectura e interpretación de textos e información técnica en inglés. Comprensión y producción de textos de complejidad creciente en inglés para comunicarse solicitando o aportando información técnica por e-mail o en foros y listas de discusión.

3.3. Formación Específica

La formación específica del Técnico Superior en Desarrollo de Software es la directamente relacionada con las actividades propias de su Perfil Profesional, por ello los contenidos correspondientes a este campo están agrupados en forma tal que puedan relacionarse fácilmente con las actividades propias del técnico. Para poner en perspectiva y señalar el nivel de los contenidos, se los acompaña con ejemplos de ejercicios prácticos que contribuyan a la formación a través de desempeños que preparen al estudiante para su trabajo futuro.

Aspectos formativos

A los fines de la homologación y con referencia al perfil profesional se considerarán los siguientes aspectos de la trayectoria formativa del técnico:

Aspecto formativo referido modelizar artefactos de software a partir de especificaciones, refinándolas en caso necesario, para determinar el diseño detallado y las características de una solución que las satisfaga en el contexto de la arquitectura del sistema de software del cual van a formar parte.

Relativos a interpretar críticamente especificaciones.

El software de aplicaciones intenta resolver necesidades de información o automación establecidas en necesidades planteadas por usuarios u otros interesados, las que, una vez acordadas, son plasmadas en especificaciones de requerimientos, ya sean estas formales o informales.

Estas especificaciones se refieren a las funciones que debe realizar el software, a interacciones con usuarios y otros sistemas, requisitos de calidad y comportamiento y son el punto de partida para lo que va a desarrollar, por lo que el desarrollador de software debe ser capaz de interpretarlas, analizándolas críticamente, detectando posibles lagunas o incoherencias, preguntando por aspectos faltantes o incoherentes y validar su propia interpretación con quienes lideran el proyecto.

Contenidos relacionados al análisis y especificación de requerimientos:

Análisis de requerimientos de software, el proceso, partes interesadas. Requerimientos funcionales, prioridades y criterios de realización. Análisis orientado a objetos y UML. Diagramas de clase. Escenarios, historias y casos de uso; diseño centrado en el usuario. Representación del comportamiento: diagramas de secuencia, máquinas de estado, diagramas de actividad. Redes de Petri. Pre y post condiciones.

Análisis de datos: datos de referencia y de operaciones; datos de nivel de recursos y de volumen de actividad. Modelo Entidad/Relación. Principales formas normales. Diccionario de datos.

Requerimientos no funcionales, ejemplos y su influencia en el diseño del software. Herramientas de modelización. Validación de requerimientos. Estándares de documentos de requerimientos.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que:

Producir diagramas de clase a partir de problemas correspondientes a diversos dominios. Analizar y discutir sus propiedades y corrección. Representar situaciones determinadas utilizando diagramas UML u otras técnicas. Analizar y discutir sus características y defectos. Modelizar y especificar casos de uso a partir de descripciones de situaciones realistas. Documentar escenarios. Revisar documentos de requerimientos de software utilizando buenas prácticas para determinar su calidad. Realizar revisiones cruzadas de especificaciones.

Relativos a diseñar artefactos de software.

Los programas, subsistemas y otros artefactos de software tienen que diseñarse respetando buenas prácticas y manteniendo coherencia con la arquitectura existente o prevista del sistema de software en el que estarán insertos o tendrán que interactuar.

Lograr esto requiere no sólo conocer técnicas de diseño de software sino también comprender principios de arquitectura de sistemas de software, propiedades de calidad del software y técnicas de representación.

Contenidos relacionados al diseño de artefactos de software:

Principios generales de diseño: descomposición, desacoplamiento, cohesión, reusabilidad, portabilidad, testeabilidad, flexibilidad, escalabilidad. Patrones de diseño. Arquitecturas de software: concepto de vistas, arquitecturas distribuidas, “pipe-and-filter”, “model-view-controller”. Diseño orientado a objetos. Diseño estructurado. Diseño orientado al reuso de componentes, incorporación de elementos disponibles al diseño. Diseño de interfaces con el usuario. Prototipos rápidos.

Concepto de base de datos, tipos de bases de datos. El modelo relacional, transformación del modelo E/R a relacional.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Diseñar artefactos de software (clases, objetos, métodos, algoritmos, tablas) que resuelvan problemas planteados. Analizar críticamente la eficiencia y mantenibilidad de diseños alternativos. Relacionar situaciones con patrones de diseño. Analizar diversos tipos de arquitectura de sistemas de software, discutiendo sus propiedades de calidad (escalabilidad, portabilidad, seguridad, mantenibilidad). Construir prototipos rápidos con herramientas sencillas. Diseñar tablas y bases de datos relacionales. Analizar y discutir su eficiencia y escalabilidad. A partir de un diseño, analizar clases de equivalencia y diseñar esquemas de prueba.

Aspecto formativo referido a construir los artefactos de software que implementen el diseño realizado, aplicando patrones o reutilizando código en la medida en que resulte posible. Esto incluye revisar y depurar código propio o ajeno para corregirle defectos, optimizarlo o adaptarlo a nuevas funcionalidades que sean requeridas. Al hacer esto se aplican buenas prácticas de programación y documentación, conforme a procedimientos de calidad establecidos. También hay que participar en revisiones cruzadas de artefactos de software para asegurar la calidad del producto.

Relativos a la redacción y depuración de código que responda al diseño propuesto

El diseño tiene que convertirse en programas que satisfagan efectiva y eficientemente los requisitos planteados dentro de la arquitectura prevista para el sistema, respondan a buenas prácticas, siendo

comprensibles y fáciles de modificar, y que presenten robustez ante situaciones no previstas. Esto se logra no sólo redactando código, hay que encontrar y adaptar módulos o clases ya existentes para utilizarlas en lo que se está construyendo, verificar lo construido mediante diverso tipo de pruebas y volver a trabajar sobre lo hecho para depurar errores o malfuncionamientos encontrados, así como para optimizarlo.

Por otra parte, nuevos negocios, necesidades de usuarios o regulaciones de las autoridades plantean la necesidad de modificar aplicaciones existentes, con lo cual algún desarrollador tiene que tomar ese programa, interpretar su código para comprenderlo y ubicar dónde ese programa realiza lo que hay que cambiar. Una vez localizado el punto a modificar, tiene que plantear la forma de resolver la situación e introducir los cambios necesarios, probándolo nuevamente para verificar que haga lo esperado y que tampoco hayan cambiado funcionalidades que tenía previamente.

Esto implica dominar programación aplicando conceptos de abstracción, descomposición, algoritmia, estructuras de datos, recursividad, herencia y polimorfismo. Por otra parte, hay que aplicar buenas prácticas de programación y documentación, conocimientos de "testing" y tener conciencia del proceso completo de desarrollo, lo que es independiente de la tecnología utilizada.

Contenidos relacionados a algoritmos y estructuras de datos:

Concepto de algoritmo, resolución algorítmica de problemas, estrategias de diseño, de implementación, de depuración. Algoritmos fundamentales, algoritmos numéricos simples.

Estructuras fundamentales, variables, tipos, expresiones y asignaciones, entrada/salida, estructuras de control condicionales e iterativas, funciones y pasaje de parámetros, descomposición estructurada.

Concepto de lenguaje de alto nivel y la necesidad de traducción, comparación entre compiladores e intérpretes, aspectos de la traducción dependientes y no dependientes de la máquina. Programas generadores de código.

Máquinas virtuales, concepto, jerarquía de máquinas virtuales, lenguajes intermedios, asuntos de seguridad que surgen al ejecutar código en una máquina diferente.

Representación de datos numéricos, rango, precisión y errores de redondeo. Arreglos. Representación de datos de caracteres, listas y su procesamiento. Manejo de memoria en tiempo de ejecución, punteros y referencias, estructuras encadenadas, pilas, colas y tablas de hashing. Recolección de espacios no utilizados. La elección de una estructura de datos adecuada.

Diseño orientado a objetos, encapsulamiento y ocultamiento de información, separación entre comportamiento e implementación, clases y subclases, herencia (sustitución), polimorfismo (subtipos vs. herencia), jerarquías de clases, clases colección y protocolos de iteración.

Verificación unitaria de unidades de código, concepto de cubrimiento, organización, ejecución y documentación de la prueba.

Recursión, concepto, funciones matemáticas recursivas, funciones recursivas simples, estrategia de dividir y conquistar, backtracking recursivo.

Algoritmos de búsqueda sucesiva y binaria, de ordenamiento con tiempos cuadráticos (selección, inserción), con tiempos O(NlogN) (quicksort, heapsort, mergesort). Tablas de hashing, estrategias para evitar colisiones. Árboles de búsqueda binaria. Representación de grafos. Algoritmos de camino mínimo. Concepto de autómata. Elementos de complejidad de algoritmos.

Declaraciones y tipos, la concepción de tipos como conjunto de valores junto con operaciones, modelos de declaración, elementos de verificación de tipos, tipos y polimorfismo paramétrico.

Estándares de nomenclatura y formato en programación, encabezado de módulos u objetos con comentarios que expliciten sus alcances y limitaciones, inserción de comentarios o advertencias en el código, documentación adicional.

Programación conducida por eventos, métodos para manejo de eventos, propagación de eventos, manejo de excepciones.

Programación defensiva, importancia de verificar para evitar el overflow de arreglos y listas. Alternativas o dispositivos de lenguajes de programación para evitarlo. Cómo atacantes pueden utilizar el overflow para destruir el stack en tiempo de ejecución.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Resolver ejercicios de programación, tanto con lápiz y papel como en computador. Se espera que al concluir el ciclo los estudiantes dominen al menos dos de los tres paradigmas de programación (objetos, imperativa-estructurada o funcional) y varios lenguajes (por lo menos uno correspondiente a cada paradigma, pero también otros, en particular los que tienen aplicación en páginas web). (Se entiende que el tener que adaptarse a diversos tipos de lenguajes de programación y resolver diversa clase de problemas utilizándolos ayuda al proceso de desarrollar capacidad de abstracción.) Revisar y corregir programas dados. Resolver diversos tipos de problemas comenzando por especificar su propia comprensión de la asignación, diseñar una solución, programar o integrar partes de código utilizando el ambiente de programación indicado, documentándola de acuerdo a buenas prácticas y realizar la verificación unitaria de lo realizado. Intercambiar artefactos de software asumiendo la obligación de interpretar y criticar o mejorar lo recibido. Desarrollar proyectos grupales durante los cuales se simulen condiciones similares a las del trabajo profesional y en los que cada uno aporte componentes que deben integrarse en el producto final.

Relativos a desarrollar software que utilice bases de datos

El código de los programas se utiliza para computar datos, los que pueden ser internos del programa o, más generalmente, encontrarse o tener que ser almacenados en archivos y bases de datos. En consecuencia, el desarrollador no sólo tiene que conocer de algoritmos y lenguajes, sino también de manejo de la información.

Esto implica conocer de modelos de información que faciliten su almacenamiento y recupero, modelos de datos, indexación, lenguajes de consulta y características de los principales modelos y sistemas de bases de datos.

Actualmente, con sistemas de información distribuidos hace falta obtener o intercambiar datos con otros sistemas a través de Internet y, eventualmente, hacer uso o interactuar con herramientas externas de búsqueda.

Contenidos relacionados con bases de datos:

Concepto de almacenamiento y recuperación de información, captura y representación, aplicaciones, búsqueda, recuperación, vinculación, navegación. Metadatos o esquemas asociados con los datos objeto del procesamiento. Problemas de escalabilidad, eficiencia y efectividad. Privacidad, integridad, seguridad y preservación de la información. La persistencia e integridad de los datos.

Modelización de datos, modelos conceptuales (E/R, UML), modelo orientado a objetos, modelo relacional, modelos semiestructurados (XML). Concepto y evolución de los sistemas de bases de datos, sus componentes, funciones de un sistema de base de datos.

Lenguajes de consulta (SQL, QBE), definición de datos, álgebra relacional, formulación de consultas, lenguaje de actualización, restricciones, integridad. SQL embebido en un lenguaje imperativo. “Scripts”. Introducción a un lenguaje de consulta de objetos. Procedimientos almacenados.

Diseño de bases de datos, dependencia funcional, formas normales, descomposición de un esquema, claves primarias y secundarias. Procesamiento de transacciones, fallas y recuperación, control de concurrencia. Bases de datos distribuidas, problemas que surgen con su explotación.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Resolver ejercicios de álgebra relacional. Se espera que al concluir el ciclo los estudiantes resulten capaces de explotar una base de datos relacional. Revisar y corregir programas dados. Resolver diversos tipos de problemas comenzando por especificar consultas a bases de datos dadas, programar actualizaciones de datos en base a cálculos con nuevos datos, preocupándose tanto por la integridad de la información como por la eficiencia de los procesos. Diseñar tablas y bases de datos, incorporar procedimientos. Desarrollar proyectos grupales durante los cuales se simulen condiciones similares a las del trabajo profesional y cada uno aporte componentes que deben integrarse en el producto final.

Relativos a producir interfaces adecuadas para el usuario

En los sistemas de información el usuario suele proveer datos al sistema y utilizar la información que brinda el sistema para tomar decisiones de diverso tipo. En tal sentido, la calidad de las interfases y la interacción del usuario con el sistema resultan muy importantes, ya que interfases pobremente diseñadas pueden llevar a registrar mal los datos o a dificultar el uso del sistema. En particular, cuando se producen situaciones de excepción (datos o comandos incorrectos por parte del usuario o la solicitud de algo que el sistema no puede realizar) es conveniente planificar un diálogo adecuado para resolver la situación, incluyendo ayudas para el usuario.

En consecuencia, el desarrollador, a pesar de que inscriba su componente en un diseño más general, debe conocer distintos tipos de interfases con el usuario, principios de diseño de interfases visuales, verificaciones básicas a realizar sobre los datos de entrada y manejo de ayudas y del diálogo para superar las dificultades que pueda encontrar el usuario.

En la actualidad se han difundido una serie de dispositivos (móviles, GPSs, tabletas de diversas características, pantallas que reaccionan al contacto, recolectores de datos) que amplían el espectro de las interfases con los usuarios, lo que genera una gama de tecnologías y modelos de interacción que un buen desarrollador de software debe dominar para su trabajo o, al menos, estar en condiciones de adaptarse rápidamente.

Contenidos relacionados con interacción ser humano-máquina:

Interacción ser humano-máquina, conceptos básicos. Distintos contextos para interfaces: visuales o de texto en aplicaciones habituales, interfaces web con dispositivos para navegación, sistemas colaborativos, juegos y otras aplicaciones multimediales, interfaces con o por medio de diversos dispositivos, lo que pueden incluir teléfonos móviles y TV digital.

Proceso de desarrollo centrado en el usuario: foco temprano en los usuarios, prueba empírica de la calidad, diseño iterativo. Medidas de evaluación: utilidad, eficiencia, facilidad de aprendizaje, satisfacción del usuario. Modelos de diseño de la interacción: atención, movimiento, cognición, percepción y reconocimiento.

Diseño para el cambio: adaptación a otras lenguas o localismos, adaptación a la diversidad de condiciones humanas. Notación para especificar interfaces. El manejo de los errores del usuario o del sistema. Técnicas y herramientas de prototipado.

Principios de interfaces gráficas, *acción-objeto vs. objeto-acción*, eventos en interfaces de usuario, estándares, errores más comunes. Interfaces para un sistema nativo, uso del browser para sistemas que operen en la web.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Considerar, discutir y diseñar interacciones software-usuario. Diseñar diversas pantallas que respondan a determinadas propuestas y evaluar conjuntamente lo obtenido. Diseñar interfaces para la web con ayudas para la navegación. Diseñar interfaces para alguna norma estándar (USB, “bluetooth”) para dispositivos.

Relativos a desarrollar software que opere en ambientes distribuidos

En la actualidad la mayor parte de los sistemas operan en forma distribuida a través de redes locales o de Internet, utilizando en muchos casos como interfase el software de navegación (browser) de la máquina cliente. Esto implica mantener un diálogo cliente-servidor que intercambie datos y permita acceder y actualizar bases de datos situadas a distancia.

El desarrollador tiene que dominar la programación en un ambiente cliente-servidor, para lo cual tiene que comprender conceptos de arquitectura de sistemas web, aspectos de seguridad y de comportamiento.

Contenidos relacionados con computación centrada en redes:

Aplicaciones en redes. Protocolos a nivel de la capa de aplicación. Interfaces web: “browsers” y APIs. Subprotocolos TCP y UDP. El “socket” como abstracción.

Modelo cliente servidor. Primeras acciones de ambos. Creación de “sockets” y ligado de direcciones. Par cliente/servidor TCP. Concepto de sesión. Par cliente/servidor UDP. Concepto de paquete. “Polling” con primitivas bloqueantes. RCP. “Object brokers”.

Tecnologías web, modelos de computación distribuida en la red. Protocolos web. Lenguajes de programación utilizados para el desarrollo de páginas y sistemas web.

Principios de ingeniería web. Sitios web estructurados mediante bases de datos. Tecnologías de búsqueda en web. El papel del "middleware", herramientas de apoyo.

Aplicaciones basadas en tecnologías para toda la empresa. Aplicaciones cooperativas. Sistemas de "workflow". Herramientas para desarrollo en ambientes web. "Frameworks" de aplicaciones y su utilización.

Creación y administración de sitios web.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Diseñar y programar aplicaciones sencillas que interactúen en un ambiente cliente-servidor. Diseñar sitios web organizados como bases de datos para que el usuario pueda actualizarlos sin intervención de desarrolladores. Utilizar ambientes de programación para web, programar aplicaciones interactivas que actualicen bases de datos, considerar y discutir aspectos de seguridad relativos a las mismas.

Verificar los artefactos de software construidos considerando las necesidades de cobertura de la prueba. Para ello diseña los casos considerando el entorno de pruebas y ejecuta pruebas unitarias, así como registra los datos y resultados. De ser necesario, realiza acciones correctivas sobre el código hasta satisfacerse de que cumpla con las especificaciones recibidas.

Relativos a verificar el buen funcionamiento de los programas

En el software es tan alta la distancia entre el diseño y la construcción, que resulta totalmente improbable producir inicialmente programas sin defectos. Así es que los productos tienen que ser verificados mediante pruebas que comprueben su calidad. Para ello hay que diseñar conjuntos de datos de prueba y realizar procesos en condiciones controladas que den cuenta de diversos aspectos. En primer lugar, que el programa satisfaga los requerimientos planteados. También que tenga robustez y no acepte datos incorrectos o no realice acciones imprevistas cuando un usuario se equivoca en un comando.

Esto se inscribe en el concepto de verificación unitaria, que debe realizar el desarrollador de software para satisfacerse que ha realizado lo requerido. Sin embargo, la buena práctica implica que un grupo independiente debe integrar lo realizado por cada desarrollador y someterlo a prueba conjunta, lo que puede poner de relieve fallas originadas en la interacción. La detección de fallas motiva que el desarrollador vuelva sobre el código para encontrar los defectos y los resuelva.

Para aplicar con propiedad técnicas de "testing" un desarrollador de software tiene que conocer principios generales, los diversos tipos de "testing" que se utilizan en el proceso de desarrollo de software y dominar la utilización de ambientes y herramientas específicos de "testing".

Contenidos relacionados con "testing"

Distinción entre validación y verificación. Enfoques estáticos y dinámicos. Fundamentos de "testing", testeo de caja negra y de caja blanca. Pruebas funcionales: generación de casos o datos de prueba, clases de equivalencia. Pruebas estructurales: pruebas estáticas, pruebas dinámicas, cobertura de la prueba. Otro tipo de objetivos: verificación de usabilidad, confiabilidad, seguridad. Registro de fallas, seguimiento de fallas e informes técnicos.

Prueba unitaria, de integración, validación y prueba del sistema. Desarrollo conducido por el testeo. Refactorización del código. Testeo de regresión. Verificación y validación de artefactos que no constituyen código: documentación, archivos de ayuda, material de capacitación. Inspecciones, revisiones cruzadas, auditorías.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Procesar pruebas e identificar defectos en artefactos producidos por otros. Planificar y diseñar casos y conjuntos de datos para prueba de artefactos dados, respondiendo a objetivos y requisitos de cobertura. Implementar pruebas de programas y pequeños sistemas utilizando herramientas y creando ambientes necesarios, realizar los procesos y revisar los resultados para generar informes de fallas.

Gestionar sus propias actividades dentro del equipo de trabajo del proyecto. Ello comprende la planificación (organización y control) de las tareas a realizar, el oportuno reporte de avances y dificultades y el registro y reflexión sobre lo realizado para capitalizar experiencias y estimar métricas aplicables a su actividad.

Relativos a la gestión personal dentro de un contexto de procesos de ingeniería de software

El desarrollador tiene que desenvolverse en el marco de un equipo de trabajo organizado en función del proyecto a encarar. En consecuencia, asume responsabilidad por su asignación dentro del proyecto pero interactúa con pares y líderes del equipo para lograr un mejor proceso conjunto.

En consecuencia, además de dominar las técnicas con que va a realizar la parte asignada, tiene que tener una comprensión del sistema y de la totalidad del proceso, tiene que comprender y cumplir estándares establecidos para el proyecto tratando de aportar lo mejor de su parte, aceptar soluciones resueltas grupalmente o por el liderazgo y tiene que colaborar con otros pares y juniors en la solución de los problemas.

Contenidos relacionados con el proceso de ingeniería de software

Conceptos de dinámica de grupos, grupo y equipos de trabajo, características distintivas. La tarea como eje de la convocatoria de todo grupo/equipo. Tarea explicita e implícita. Dinámica de lo grupal. La mutua representación interna, espacio y tiempo. Objetivos grupales y metas individuales. Lo individual versus lo grupal. Roles y estereotipos, rotación de roles. La comunicación, medios, ruidos que afectan a la comunicación. Importancia de la retroalimentación. La empatía. La escucha activa. Conflictos, técnicas de resolución alternativa.

El equipo de proyectos de desarrollo de software, roles y responsabilidades de sus integrantes. Programas de trabajo y resolución conjunta de problemas. Modelos de ciclo de vida del software y de procesos de desarrollo. El problema del mantenimiento y las migraciones de plataforma.

Metodologías tradicionales y ágiles. Metodologías ágiles, concepto de “sprint”, fraccionamiento del producto en unidades realizables en un “sprint”, cola de pendientes, mejora de productos provisorios (refactoring), variación de los roles y la documentación en el marco de un proceso en el que se aplican metodologías ágiles.

Gestión de los cambios, conceptos de versión, “Guild”, producto de la asignación. Concepto de componente. Elementos de administración de la configuración y control de versiones de software. Herramientas de versionado. Otras herramientas (bibliotecas, diccionarios, repositorios) del proyecto.

El problema de asegurar la calidad y elementos de métricas. Modelos de madurez de la capacidad de desarrollo. Enfoques para la mejora del proceso, métricas. El proceso personal de software, estadísticas personales y capitalización de experiencias.

Como parte de la forma de adquirir estos aprendizajes y para demostrar prácticamente los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Participar de proyectos conjuntos de desarrollo de artefactos de software en los que se pongan en práctica diferentes metodologías. Poner en práctica estadísticas elementales propias del proceso personal de software. Realizar revisiones cruzadas de código proponiendo mejoras. Organizar la documentación de un proyecto. Utilizar herramientas de versionado y administración de la configuración. Reflexionar en forma conjunta sobre experiencias y conclusiones obtenidas.

Desempeño de base

Conocer y saber utilizar con propiedad y en condiciones de seguridad recursos de hardware, software y redes para emplear los ambientes que necesite para el desarrollo y la verificación del software, mantener los repositorios de información que necesite utilizar y disponer de los productos de su trabajo en condiciones de confiabilidad.

Relativos al ambiente de desarrollo

El desarrollador no sólo tiene que tener capacidades como para resolver los problemas que presenta el diseñar y programar artefactos de software que satisfagan las asignaciones planteadas en el contexto de la arquitectura propuesta. Tiene que configurar el ambiente de programación y el de “testing” que va a utilizar en su trabajo, generar o extraer datos para producir los que necesite para

probar lo que realizó. Eventualmente, tratar de interpretar fallas en función de posibles problemas de compatibilidad con otro software.

Para realizar esto el desarrollador debe conocer sobre sistemas operativos y debe ser capaz de manejarse hábilmente con diversos editores, configurar aspectos de software y hardware y explotar con habilidad recursos de programación, incluyendo entre los mismos bibliotecas de objetos y programas propias, de su organización o disponibles a través de Internet, así como plantear y resolver consultas de problemas de programación a través de foros y listas públicas o privadas.

Contenidos relacionados con sistemas operativos, editores y bibliotecas de programas

Los sistemas operativos, su papel y propósito, la historia de su desarrollo, funcionalidades típicas. Mecanismos que soportan los modelos cliente-servidor y otros dispositivos. Características y objetivos de su diseño y su influencia en la seguridad, interoperabilidad, capacidad multimedial.

Estructuras de sistemas operativos (monolíticos, modulares y de "micro kernel"). Abstracciones, procesos y recursos. Organización de los dispositivos, interrupciones: métodos e implementación. Concepto de estados usuario/supervisor y protección, transición al modo supervisor.

Estados y transiciones; cola de procesos, bloque de control de procesos. Despacho, "switching" de contexto, "switching" cooperativo y "preempted". Ejecución concurrente: ventajas y desventajas. El problema de la exclusión mutua y algunas soluciones. Bloqueos: causas, condiciones, prevención. Paso de mensajes sincrónico y asincrónico. Problema de consumidor-productor y sincronización (mutex, semáforos). Multiprocesamiento (interrupción de ciclos, reentrada).

Políticas de despacho de procesos; programación con y sin prioridades de interrupción. Procesos y "threads". Elementos de tiempo real y tiempos límite.

Administración de memoria. Revisión de memoria física y hardware de administración de memoria. Paginamiento y memoria virtual. "Working sets" y "trashing". "Cacheo".

Administración de dispositivos, características de dispositivos seriales y paralelos. Abstracción de diferencias entre dispositivos. Estrategias de "buffering". Acceso directo a memoria. Recuperación de fallas.

Seguridad y protección. Políticas y mecanismos de separación. Métodos y dispositivos de seguridad. Protección, control de acceso y autenticación. Backups.

Sistemas de archivo (datos, metadatos, operaciones, organización, "buffering", secuenciales y no secuenciales). Índices: contenido y estructura. Técnicas estándares de implementación. Archivos de mapeo de memoria. Sistemas de archivo para propósitos especiales. Denominación, búsqueda, acceso, backups.

"Scripting". Comandos básicos del sistema, creación de "scripts", pasaje de parámetros. Ejecución de un "script".

Ambientes gráficos para edición, editores inteligentes. Herramientas integradas disponibles para la edición en distintos lenguajes y ambientes. Bibliotecas de clases, programas y rutinas.

Aspectos de administración de redes, uso de contraseñas y mecanismos de control de acceso, servidores de nombres de dominos y de servicios, proveedores de servicios en Internet. Aspectos de seguridad y firewalls. Asuntos de calidad de servicio: comportamiento, recuperación de fallos.

Como parte de la forma de adquirir estos aprendizajes y demostración práctica de los resultados alcanzados, en el curso de su formación los estudiantes tienen que realizar:

Localizar y seleccionar artefactos de software, libre o bajo otras licencias, que respondan a ciertas características. Instalar, configurar y personalizar sistemas operativos, compiladores de lenguajes, editores y otros ambientes de programación o de prueba de programas. Crear y organizar repositorios de documentación y programas para uso personal o de proyectos.

3.4 Práctica profesionalizante

El mundo del trabajo, las relaciones que se generan dentro de él, sus formas de organización y funcionamiento y la interacción de las actividades productivas en contextos socio económicos locales y regionales, conjugan un conjunto de relaciones tanto socio culturales como económico productivas

que sólo puede ser aprehendido a través de una participación activa de los estudiantes en distintas actividades de un proceso de producción de bienes o servicios.

La adquisición de capacidades para desempeñarse en situaciones sociolaborales concretas sólo es posible si se generan en los procesos educativos actividades formativas de acción y reflexión sobre situaciones reales de trabajo.

En este sentido, el campo de formación de la práctica profesionalizante está destinado a posibilitar la integración y contrastación de los saberes construidos en la formación de los otros campos, y garantizar la articulación teoría-práctica en los procesos formativos a través del acercamiento de los estudiantes a situaciones reales de trabajo, propiciando una aproximación progresiva al campo ocupacional hacia el cual se orienta la formación y poniendo a los estudiantes en contacto con diferentes situaciones y problemáticas que permitan tanto la identificación del objeto de la práctica profesional como la del conjunto de procesos técnicos, tecnológicos, científicos, culturales, sociales y jurídicos que se involucran en la diversidad de situaciones socioculturales y productivas que se relacionan con un posible desempeño profesional.

Un espacio de práctica profesionalizante tiene que permitir la integración de un conjunto significativo de funciones primordiales del perfil profesional en el marco de un ambiente de trabajo real o simulado. En ese sentido, las actividades formativas grupales e individuales tienen que integrar prácticas como la interpretación crítica de especificaciones de artefactos de software, el diseño de la solución, su justificación y validación; la construcción de partes no triviales, revisión, verificación unitaria y depuración, aplicando buenas prácticas de programación y documentación; así como también su integración con otros artefactos ya existentes o desarrollados por otros para conformar versiones, incluyendo la depuración de los errores encontrados. Esto requiere un conocimiento y apropiación del campo profesional y la interacción con sus distintos actores.

Esto se puede lograr en el sector productivo, realizando acuerdos en los que se planifique y verifique que el estudiante realice un conjunto de tareas del tipo de las descriptas, o en la institución educativa, creando ámbitos de desarrollo de software, típicamente denominados *software factory*, que reproduzcan las condiciones en las que desarrollan proyectos las empresas del sector, organizando equipos de desarrollo y contando con figuras docentes que asuman papeles como gerentes de desarrollo o responsables por la calidad. También resulta importante contar con un cliente creíble que plantee demandas realistas y que se preste al juego de modificar algunos de los requerimientos durante el proceso.

Esta actividad formativa debe ser cumplida por todos los estudiantes, con supervisión docente, y la institución educativa debe garantizarla durante y a lo largo de la trayectoria formativa.

3.5. Carga horaria mínima

La carga horaria mínima total es de 1600 horas reloj[4]. Al menos la tercera parte de dicha carga horaria es de práctica de distinta índole.

La distribución de carga horaria mínima total de la trayectoria por campo formativo, según lo establecido en párrafo 3.2.3 de los Lineamientos para la organización institucional y curricular de la Educación Técnica Profesional de la Educación Secundaria y Superior aprobado por Res. CFE Nro. 47/08, es:

- Formación general: mínimo el 10% del total,
- Formación de fundamento: mínimo el 20% del total,
- Formación técnica específica: mínimo el 30% del total,
- Prácticas profesionalizantes: mínimo el 20% del total.

----------o----------

[4] Esta carga horaria mínima está determinada por la Res. CFE N° 47/08 "Lineamientos para la organización institucional y curricular de la Educación Técnica Profesional de la Educación Secundaria y Superior".