TALLER: MECANISMOS DE REUSO EN OO – ARQUITECTURA, PATRONES Y FRAMEWORKS

Urciuolo Adriana, Sandoval Sandra
Universidad Nacional de la Patagonia San Juan Bosco – Sede Ushuaia, Darwin y Canga, (9410) Ushuaia
e-mail: urciuolo@tdfuego.com, sandrasandoval@ciudad.com.ar

Resumen

En el presente trabajo se describe la experiencia de dictado de un taller sobre mecanismos orientados a objetos para diseño de software reusable, dirigido a alumnos de la carrera Licenciatura en Informática (U.N.P.S.J.B. – Sede Ushuaia); el mismo ha sido implementado en forma preliminar, durante el presente ciclo lectivo, dada la necesidad de incorporar conceptos avanzados de diseño orientado a objetos, que permitan una mejor comprensión de las ventajas del paradigma. No siendo posible alcanzar este objetivo en las asignaturas del Plan de Estudios, se optó por la realización de un taller dirigido a los alumnos que hayan finalizado las cursadas de las asignaturas: Análisis y Diseño de Sistemas y Programación Avanzada. Se trabaja sobre la base de un proceso de análisis y diseño de sistemas centrado en la arquitectura, aplicando patrones en las diferentes etapas del desarrollo, dando por supuesto el conocimiento de métodos orientados a objetos. Se brindan los conceptos fundamentales relativos a desarrollo basado en componentes, arquitectura de software y a distintos tipos de patrones: arquitecturales, de análisis y de diseño, con aplicaciones apropiadas. Dados los resultados satisfactorios de la experiencia realizada, la presente propuesta ha sido formalizada ante el Consejo Académico, para la continuidad de su dictado.

Palabras clave: diseño, reuso, arquitectura, patrones, taller

Introducción

La enseñanza de la tecnología de Orientación a Objetos involucra el dictado de conceptos y métodos de *programación, análisis y diseño*, así como de *procesos de desarrollo de software* adecuados para el uso de esta metodología. Usualmente en las asignaturas de Programación OO se brindan los conceptos básicos del paradigma, sus ventajas, los problemas que resuelve y la implementación de dichos conceptos en un lenguaje OO. En las asignaturas relativas al Diseño de sistemas por otra parte, se presentan los beneficios de la tecnología OO para el desarrollo de software de fácil mantenimiento, extensibilidad y reuso, métodos de elicitación de requerimientos apropiados para el desarrollo OO y los métodos disponibles para realizar el análisis y diseño de la estructura y comportamiento de objetos, así como una introducción a los procesos de desarrollo apropiados para tal fin. En síntesis, el alumno aprende a analizar el contexto de un sistema, identificar clases, sus responsabilidades y asociaciones, a construir diagramas de clases, de interacciones, de eventos, etc., definir las clases del diseño con sus métodos e implementar un diseño en un lenguaje de programación OO. No obstante, se observa que si bien en dicho proceso de aprendizaje el alumno llega a comprender las ventajas del paradigma en la construcción de software fácil de mantener, no sucede lo mismo con relación a la extensibilidad y el reuso, ya que estas características implican otro grado de madurez en el estudio de la metodología.

Sólo después de haber realizado todo el proceso de desarrollo de sistemas OO (incluida la implementación), se comprenden los problemas recurrentes que aparecen en las distintas etapas y la necesidad de contar con soluciones flexibles.

La necesidad del dictado de un taller sobre reuso en OO, surgió como consecuencia de la experiencia realizada por un grupo de docentes en el dictado de las asignaturas Programación Avanzada y Análisis y Diseño de Sistemas (ambas de 2º año, cuatrimestrales), a partir de la implementación del nuevo plan de la Licenciatura en Informática en la sede Ushuaia. Dichas asignaturas abarcan gran cantidad de temas básicos, fundamentales para la formación en desarrollo de software del alumno. En particular, se brinda el primer contacto del alumno con el paradigma orientado a objetos, tanto desde la programación mediante la enseñanza del lenguaje JAVA, como desde el análisis y el diseño. En ambas materias, se incorporan los conocimientos básicos de tecnologías orientadas a objetos (OO) desde distintas perspectivas.

Vista la necesidad y conveniencia de incorporar conceptos relativos al reuso de software y los mecanismos que brindan las técnicas mencionadas para este fin y considerando la imposibilidad de hacerlo durante el transcurso de la cursada de las materias del Plan de Estudios, se optó por la alternativa de la implementación de un Taller optativo, donde se presenten conceptos relativos a reuso de software mediante la utilización de técnicas OO: arquitectura de software, componentes, patrones y frameworks.

Durante el mes de marzo del año 2002, se realizó la primera experiencia, dictando un taller sobre Patrones de diseño OO para software reusable, como clases complementarias de la asignatura Análisis y Diseño de sistemas. Los resultados positivos de la experiencia motivaron su formalización en una propuesta presentada al Consejo Académico de la Facultad de Ingeniería, para su análisis y aprobación.
Considerando la experiencia realizada, en el presente trabajo se presentan los contenidos de la propuesta formal aprobada para la implementación del taller, sus motivaciones y los beneficios obtenidos y esperados de su implementación.

Motivación

El desarrollo del taller se consideró necesario debido a los siguientes factores:

- En la actualidad el tema de reuso ha cobrado vital importancia en la Ingeniería de software, resultando por lo tanto conveniente incorporar a nivel de carreras de grado, conceptos avanzados de análisis y diseño que favorezcan el reuso: arquitectura de software, componentes, patrones y frameworks orientados a objetos.
- No resulta factible la incorporación de más contenidos en las asignaturas de formación básica en metodología OO (Análisis y Diseño de Sistemas y Programación Avanzada) debido a cuestiones de tiempo y al grado de madurez del alumno en temas de desarrollo de software.
- Los patrones ayudan a conocer y comprender problemas recurrentes de diseño y sus soluciones, resultando por lo tanto fundamental y enriquecedora su incorporación en el proceso de aprendizaje del diseño OO.

- Se ha observado que en un primer curso de objetos, los alumnos comprenden los objetivos del paradigma con relación al desarrollo de software reusable, pero la conceptualización que llegan a desarrollar, no les permite comprender fehacientemente cómo se alcanza dicho objetivo. Si bien el concepto de herencia les permite entender rápidamente cómo reusar código, no resulta tan obvia la manera en que los mecanismos de delegación y composición permiten llevar adelante alternativas de diseño más flexible. Por lo tanto, la sensación es de quedar a mitad de camino en la comprensión de los beneficios del uso de este paradigma.
- En la actualidad se reconoce a la Arquitectura de software como un componente clave para el desarrollo de sistemas y el vehículo apropiado para el reuso de conocimiento de diseño.

Objetivo general

Se pretende capacitar a los alumnos en el desarrollo de software reusable, mediante la enseñanza de técnicas orientadas a objetos, apropiadas para favorecer tal característica, en las distintas etapas del desarrollo de software.

Objetivos específicos

Se espera que el alumno logre:
- Comprender las ventajas del paradigma de orientación a objetos en el desarrollo de software reusable.
- Conocer los diferentes mecanismos de reuso que brinda el paradigma OO, sus ventajas y restricciones.
- Entender el concepto de arquitectura y la importancia de diseñar arquitecturas flexibles.
- Conocer a un nivel básico, diferentes estilos de arquitectura y aplicaciones.
- Comprender la relación entre arquitectura, patrones y frameworks.
- Analizar problemas recurrentes del diseño OO y las soluciones propuestas por los patrones.
- Decidir cuándo y cómo es conveniente aplicar un patrón.
- Aprender a utilizar catálogos de patrones.
- Implementar aplicaciones ejemplo en lenguaje Java, utilizando patrones de diseño OO
- Comprender las diferencias básicas entre patrones y frameworks.

Arquitectura de software. Ventajas de su enseñanza.

La enseñanza de un uso disciplinado de arquitecturas de software puede tener un impacto positivo en varios aspectos del aprendizaje de desarrollo de software, tales como (Booch et al, 1999):
- ✓ Comprensión de los sistemas (durante la etapa de desarrollo y a futuro)
- ✓ Organización del software
- ✓ Posibilidad de reuso
- ✓ Evolución de los sistemas desarrollados.

Si bien hasta el presente, no existe una definición clara y sin ambigüedades sobre el concepto de arquitectura de software, existe consenso alrededor de su relación con cuestiones tales como diseño de alto nivel y estructura general de los sistemas, incluyendo tanto las propiedades estáticas como dinámicas (Graham et al, 2001).

Larry Bass (Bass *et al*., 1998), brinda la siguiente definición:
"La arquitectura de software de un programa o sistema de computación es la estructura o estructuras del sistema, que comprende los componentes de software, las propiedades externamente visibles de esos componentes y las relaciones entre ellos".

En un trabajo fundamental sobre arquitectura de software, Garlan y Shaw (Garlan et al, 1994) plantean que a medida que el tamaño y complejidad del software crece, *el problema del diseño va más allá de los algoritmos y estructuras de datos de la computación*: diseñar y especificar la *estructura global de la arquitectura del sistema* emerge como un tipo de problema fundamental. Se incluyen cuestiones como organización y control global de la estructura; protocolos para comunicación, distribución física; composición de elementos de diseño; asignación de funcionalidad a dichos elementos, etc. Trabajos relativos a estos tópicos incluyen estilos arquitecturales, patrones y frameworks para sistemas de dominios específicos. Plantean además, que los orígenes de la arquitectura datan del momento en que los sistemas de software comenzaron a partirse en módulos y reconocen que históricamente la arquitectura ha existido en forma implícita. Introducen las importantes nociones de estilos y descripciones arquitecturales.

Las descripciones arquitecturales han sido reconocidas como esenciales para un sistema bien diseñado (Bass et al, 1999). En la actualidad dichas descripciones se reconocen como un vehículo apropiado para la codificación y el reuso de conocimiento de diseño, razón por la cual se considera que su incorporación al proceso de aprendizaje del mismo, aporta considerables ventajas y beneficios.

Por otra parte, uno de los procesos de desarrollo que emerge como estándar, el Proceso Unificado de Desarrollo de Software (SDUP) (Booch et al, 1999) se caracteriza por ser "*centrado en la arquitectura*". En la actualidad, si bien los procesos de desarrollo clásicos siguen dictándose, realizando un análisis comparativo de los mismos, SDPU se utiliza como conductor de la enseñanza del proceso de análisis y diseño OO, así como la notación UML (Booch et al, 1998) y las herramientas apropiadas para el mismo. Se plantea entonces la necesidad de incorporar los conceptos de arquitectura de software que permitan valorar y comprender las ventajas de utilizar un proceso con las características de SDUP.

En la figura 1 se ilustra, la diferencia entre la utilización de métodos de diseño y arquitectura de software (Garlan et al, 1994), a los fines de comparar la enseñanza del tradicional diseño OO y el diseño centrado en la arquitectura. Los métodos de diseño proveen un camino entre algunas clases de requerimientos de sistemas y algunas clases de implementaciones. En el ámbito de la arquitectura de software se estudia cómo definir el esqueleto o molde que cubrirá esta brecha, utilizando determinados estilos. Se puede entonces, enseñar una manera de transitar ese camino, definiendo un estilo arquitectural

dentro del cual los diferentes métodos estudiados pueden ser utilizados de forma conveniente.

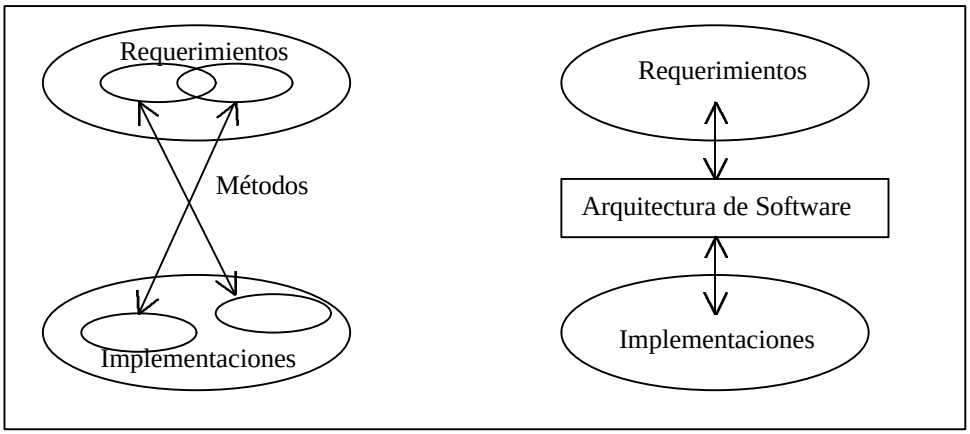

Figura 1.- Desde los requerimientos a la implementación (Adaptación Garlan & Perry, 1994)

En base a lo dicho, métodos de diseño y arquitectura se complementan mutuamente. Es conveniente por lo tanto, incorporar en una etapa posterior a la enseñanza de los métodos de diseño brindados en el primer curso (Odell et al, 1998), el uso de estilos arquitecturales y/o arquitecturas para dominios específicos en el proceso de análisis y diseño de sistemas.

Cuestiones importantes a enfatizar en el proceso de aprendizaje del diseño OO referidas a la arquitectura son entre otras: componentes (Wills, 96), desacoplamiento, estilos arquitecturales, uso de frameworks y patrones.

Ventajas de la enseñanza de patrones

Los patrones han sido utilizados en la Ingeniería de Software para permitir el reuso de soluciones que han probado ser exitosas para problemas recurrentes, en las diferentes etapas del proceso de desarrollo de software. Los patrones y los estilos arquitecturales constituyen mecanismos complementarios para capturar la experticia de diseño.

Normalmente el énfasis en la enseñanza de diseño de software ha sido puesto en los algoritmos y estructuras de datos, los cuales solucionan problemas computacionales concretos tales como ordenación y búsqueda (Appleton, 2000). La enseñanza de patrones permite al alumno comprender cuestiones relativas a encapsulación, comunicación, mantenibilidad y reutilización del software. Es posible brindar los conceptos relativos a los mecanismos para implementar esas estructuras de datos y funcionalidad, proporcionando al software las características mencionadas.

El uso de patrones en la enseñanza del diseño OO es importante además porque permiten especificar abstracciones por encima del nivel de componentes individuales, es decir colecciones de componentes, sus relaciones y responsabilidades. Por otra parte proveen un lenguaje común para la comunicación de ideas y experiencias acerca de problemas y sus

soluciones, ayudan a documentar arquitecturas de software y facilitan el reuso de diseños y arquitecturas probados en forma exitosa.

Existen diferentes tipos de patrones, los cuales pueden clasificarse de acuerdo a:
- ✓ Niveles de abstracción y rangos de escala
- ✓ Dependencia de un dominio de aplicación particular
- ✓ Etapa del proceso de desarrollo de software

Actualmente las categorías de patrones utilizadas pueden clasificarse de acuerdo al nivel de abstracción y rango de escala en: patrones arquitecturales, patrones de análisis o conceptuales, patrones de diseño, idiomas (o patrones de código) y patrones de procesos. Los tres primeros se refieren a patrones relacionados con el *proceso de análisis y diseño del software*, pero se diferencian por el nivel de detalle con el cual se aplican.
Los **Patrones Arquitecturales** expresan la organización estructural global fundamental o esquema, de un sistema de Software (Buschmann et al, 1996).
Los **Patrones de Análisis** capturan modelos conceptuales en un dominio de aplicación en orden a permitir su reuso a través de las aplicaciones (Fowler, 1997).
Los **Patrones de Diseño** describen estructuras comúnmente recurrentes de componentes comunicantes que solucionan un problema general de diseño dentro de un contexto particular (Gamma et al, 1995).

Si bien en el presente trabajo el énfasis está puesto en la enseñanza de los patrones de diseño, se considera conveniente además brindar los conceptos básicos relativos a patrones arquitecturales y conceptuales, a los fines de contar con mecanismos adecuados para el reuso en todas las etapas del proceso de análisis y diseño "centrado en la arquitectura".

Descripción del Taller

Se presenta a continuación, la propuesta formal elevada al Consejo Académico, reestructurada en base a los resultados obtenidos de la experiencia realizada.

Contenidos

MODULO 1 - CONCEPTOS BÁSICOS

Descripción: En este módulo se pretende lograr que el alumno relacione los conceptos básicos del Paradigma orientado a objetos adquiridos en las materias Análisis y Diseño de Sistemas y Programación Avanzada, con el desarrollo de software reusable. Para ello se presentan distintos mecanismos de reutilización basados en dicho paradigma. Se analizan los conceptos básicos relativos a Arquitectura del software y la importancia del uso de patrones, frameworks y componentes para el planteo de arquitecturas y/o microarquitecturas reusables. Se presentan los fundamentos de patrones y frameworks, analizando las diferencias entre ellos.
Temas:
- Necesidad de reuso. Características del software reusable.
- Concepto de arquitectura de software. Arquitecturas flexibles y reusables.
- Mecanismos de reutilización en OO: componentes, patrones, frameworks.
- Patrones. Definiciones y desarrollo histórico

- Tipos de Patrones: de arquitectura, conceptuales, de diseño, idiomas. Otros patrones: organizacionales y de procesos.
- Elementos de un patrón.
- Catálogos, lenguajes y sistemas de Patrones.
- Patrones y algoritmos.
- Diferencia entre patrones y frameworks.
- Componentes. Desarrollo de software basado en componentes.

MÓDULO 2 – ARQUITECTURA DE SOFTWARE

En este módulo se presentan los conceptos relativos a arquitectura de software y a la conveniencia de utilizar un proceso de desarrollo de software "centrado en la arquitectura", a los efectos de que el alumno conozca las ventajas del Proceso Unificado (Booch et al, 1999), cuyos conceptos básicos adquirió en la materia "Análisis y Diseño de Sistemas". Se analiza la relación entre arquitectura y patrones y la necesidad del uso de los mismos para la construcción de microarquitecturas en dominios específicos.

- Concepto de arquitectura. Definiciones según diferentes perspectivas.
- Estilos arquitecturales.
- Relación entre arquitectura y patrones.
- Ventajas del análisis de dominio.
- Desarrollo de arquitecturas para dominios específicos.
- Conveniencia del uso de un Proceso de desarrollo "centrado en la arquitectura" (Unified Process, Booch et al, 1999).
- Frameworks. Conceptos básicos.

MÓDULO 3 – PATRONES DE ARQUITECTURA

En este módulo se brindan los conceptos básicos de patrones de arquitectura (Buschmann et al, 1996), se describen algunos patrones arquitecturales y se realiza una aplicación ejemplo del patrón Layers, a los fines de que se comprendan las ventajas de diseñar una arquitectura utilizando patrones.

- Conceptos básicos. Clasificación. Propiedades. Utilización.
- Patrón Broker. Descripción. Aplicabilidad. Ejemplo
- Patrón Model-View-Controler. Descripción. Aplicabilidad. Ejemplo.
- Patrón Layers. Descripción Aplicabilidad. Ejemplo.
- Aplicación: Utilización de una arquitectura Layers en la especialización de un framework conceptual para aplicaciones geográficas para un Sistema de Información ambiental

MÓDULO 4 - PATRONES DE ANÁLISIS O CONCEPTUALES.

En este módulo se brindan los conceptos básicos de patrones de análisis, se describen algunos patrones de Fowler (Fowler, 1997; 2001) y se realiza una aplicación ejemplo integradora, a los fines de que se comprenda las ventajas del uso de estos patrones en el desarrollo de modelos conceptuales.

- Conceptos básicos. Definición. Propiedades.
- Modelos conceptuales y su relación con los patrones de análisis.
- Patrones de cálculos. Party. Organization Hierarchies. Post. Ejemplos.
- Patrones de Observaciones y Mediciones. Quantity. Measurement. Observation. Protocol. Ejemplos.

- Patrones temporales: TimePoint, Temporal Object, Temporal Property, Recurring events.
- Aplicación: Construcción de un modelo conceptual para aplicaciones ambientales en base a patrones de análisis

MÓDULO 5 - PATRONES DE DISEÑO

En este módulo se dictan los conceptos fundamentales de patrones de diseño, su clasificación, las guías básicas de diseño y los mecanismos que utilizan los patrones de diseño: fuerte uso de composición y delegación, interfaces por sobre herencia. Se brindan asimismo criterios para decidir cuándo utilizar un patrón.

- Cualidades de patrones de diseño.
- Guías de diseño. Herencia vs. Interfaces. Composición. Delegación.
- Clasificación. Patrones creacionales, estructurales y de comportamiento. Patrones de clase y de objetos.
- Catálogo de patrones. Patrones más utilizados.
- Cuando usar un patrón de diseño.
- Selección del patrón de diseño.

MODULO 6 - ANÁLISIS DE PATRONES CREACIONALES.

- Patrones creacionales. Ejemplos:
- Singleton. Objetivo. Aplicabilidad. Solución. Consecuencias. Usos en el API de Java. Patrones relacionados.
- Factory Method Objetivo. Aplicabilidad. Solución. Consecuencias. Implementación en Java: “Crear componente ambiental”. Patrones relacionados.
- Abstract Factory. Objetivo. Aplicabilidad. Solución. Consecuencias. Implementación en Java: “Crear imagen”. Patrones relacionados.
- Builder. Objetivo. Aplicabilidad. Solución. Consecuencias. Implementación en Java: “Crear imagen”. Patrones relacionados.

MODULO 7 - ANÁLISIS DE PATRONES ESTRUCTURALES

- Patrones estructurales. Ejemplos:
- Adapter. Aplicabilidad. Solución. Consecuencias. Usos en el API de Java. Patrones relacionados.
- Bridge. Aplicabilidad. Solución. Consecuencias. Usos en un modelo de representación de campos continuos. Patrones relacionados.
- Decorator. Aplicabilidad. Solución. Consecuencias. Usos en un Modelo de Representación geográfica. Patrones relacionados.
- Facade. Aplicabilidad. Solución. Consecuencias. Usos en una aplicación contable. Patrones relacionados.

MÓDULO 8 - ANÁLISIS DE PATRONES DE COMPORTAMIENTO

- Patrones estructurales. Ejemplos:
- Mediator. Aplicabilidad. Solución. Consecuencias. Usos en una aplicación de simulación ambiental. Patrones relacionados
- Observer. Aplicabilidad. Solución. Consecuencias. Usos en el API de Java. Patrones relacionados

- State. Aplicabilidad. Solución. Consecuencias. Usos en un modelo de simulación hidrológica. Patrones relacionados
- Strategy. Aplicabilidad. Solución. Consecuencias. Usos en el tratamiento estadístico de un sistema de información ambiental. Patrones relacionados

Duración: Cuatro (4) semanas de duración, organizado en dos clases semanales de 2,30 hs.

Condiciones: Los alumnos del taller deben tener aprobadas las cursadas de las asignaturas Análisis y Diseño de Sistemas y Programación Avanzada, ya que se dan por supuestos los conceptos básicos de Tecnologías orientadas a objetos.

Resultados

La experiencia se realizó durante el presente ciclo lectivo a un nivel preliminar, dictándose en forma parcial los contenidos de los módulos 1, 5, 6, 7 y 8 expuestos en el Programa de contenidos. Dicha experiencia resultó satisfactoria, dado que despertó gran interés entre quienes cursaron las asignaturas básicas, logrando un buen nivel de comprensión del tema por parte de los asistentes e iniciativa por seguir profundizando en el mismo.

La ejemplificación, basada en aplicaciones concretas desarrolladas en la materia Análisis y Diseño de Sistemas, resultó fundamental para la valorización de las ventajas del uso de patrones. Asimismo permitió a los asistentes comprobar mediante aplicaciones, la forma de identificar un problema de diseño, encontrar la solución y reutilizar un diseño.

Dado lo satisfactorio de la experiencia realizada y, teniendo en cuenta que se vio la necesidad de no limitar el tema del reuso a la enseñanza de patrones de diseño, se optó por incorporar al Taller para el próximo ciclo, el dictado de conceptos más completos de arquitectura de software, patrones arquitecturales y conceptuales. De esta forma, el aprendizaje del proceso de diseño de software reusable, puede realizarse de manera integradora, partiendo del análisis del estilo arquitectural y pasando por el análisis del modelo conceptual. El programa presentado se ha desarrollado en base a los resultados obtenidos a partir de la implementación del taller, considerando los aspectos positivos y las falencias identificadas.

Conclusiones

El dictado del Taller permitió comprobar que la enseñanza de técnicas apropiadas para el reuso de software, permiten al alumno profundizar en el conocimiento de diseño orientado a objetos, valorando los beneficios y ventajas de esta técnica, así como reconocer las limitaciones propias de la etapa inicial del aprendizaje del diseño y comprender la necesidad de soluciones flexibles para los problemas, basadas en la experiencia.

Por otra parte, se pudo observar que el grado de aprendizaje de los alumnos en métodos de análisis y diseño OO, resulta adecuado para la incorporación de los conceptos relativos a mecanismos de reuso y, por lo tanto, el nivel requerido es apropiado para su dictado, así como para la incorporación de los nuevos contenidos expuestos en el programa.

Bibliografía

Appleton B. *Patterns and Software: Essential Concepts and Terminology.* Página Web: www.enteract.com/~bradapp/, 2000

Bass L.,Clements P., Kazman R. *Software Architecture in Practice* Addison-Wesley, 1998

Bass L., Kazman R. *Architecture-Based Development.* Technical Report CMU-SEI-99-TR-007, 1999

Booch G., Jacobson I., Rumbaugh J. *The Unified Modeling Language.* Addison-Wesley Publications, 1998

Booch G., Jacobson I., Rumbaugh J. *The Unified Process Software Development.* Addison-Wesley Publications, 1999

Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P. and Stal, M. *Pattern-Oriented Software Architecture: A system of patterns*. New York: John Wiley & Sons, 1996.

Fowler, M. *Analysis patterns: reusable object models*. Menlo Park: Addison Wesley Longman, 1997

Fowler, M. *Patterns for things that change with time*. Fowler Web Page, 2001

Gamma, E. H *Design Patterns. Elements of Reusable OO Software.* Addison-Wesley, 1995

Garlan D., Shaw Mary. An Introduction to Software Architecture. Advances in Software Engineering and Knowledge Engineering, Volume I, editado por V.Ambriola and G.Tortora, World Scientific Publishing Company, New Jersey, 1994.

Garlan D., Perry D. “Introduction to the special issue on Software Architecture”, Draft version, 1994.

Graham Ian with O’Callaghan and Alan Cameron Wills, “Object Oriented Methods, 3rd. Edition”, Cap. Architecture, patterns and components, 2001.

Odell J., Fowler M. *Advanced Object-Oriented Analysis and Design Using UML.* SIGS Books & Multimedia; ISBN: 052164819X, 1998

Wills A. *Frameworks and component-based development.* Publicado en OOIS '96, eds Sun, Patel, Sun [Springer]