

# CPSA Certified Professional for Software Architecture<sup>®</sup>

## - Programa de Estudio de Foundation Level -

Versión 2019.2-ES; 16 de junio de 2020



## Índice

Información legal .....	1
Relación de objetivos de aprendizaje .....	2
Introducción .....	4
¿Qué aporta una Formación a Foundation Level? .....	4
Fuera de alcance .....	5
Prerrequisitos .....	6
Estructura, duración y métodos de enseñanza .....	7
Objetivos de aprendizaje y su relevancia con respecto al examen .....	8
1. Conceptos básicos de la arquitectura de software .....	9
Términos fundamentales.....	9
Objetivos .....	9
2. Diseño y desarrollo de arquitecturas de software .....	12
Términos fundamentales.....	12
Objetivos .....	12
Referencias.....	15
3. Especificación y comunicación de arquitecturas de software.....	16
Términos fundamentales.....	16
Objetivos .....	16
Referencias.....	18
4. Arquitectura de software y calidad .....	19
Términos fundamentales.....	19
Objetivos .....	19
Referencias.....	20
5. Ejemplos de arquitecturas de software .....	21
Objetivos .....	21
Referencias.....	22

## Información legal

© (Copyright), International Software Architecture Qualification Board e. V. (iSAQB® e. V.) 2019

Este programa de estudio sólo podrá utilizarse si se cumplen las siguientes condiciones:

1. Para obtener el certificado "CPSA Certified Professional for Software Architecture Foundation Level®". Se permite el uso de este documento de texto y/o programa de estudio para la obtención del certificado mediante la creación de copias de trabajo en un ordenador de uso privado. Si se pretende cualquier otro uso este documento y/o programa de estudio, por ejemplo para su difusión a terceros, para publicidad, etc., por favor escriba a [info@isaqb.org](mailto:info@isaqb.org) para preguntar si esto está permitido. En ese caso, habría que firmar un acuerdo para obtener la licencia de uso.
2. Si usted es un formador o proveedor de formación, le será posible utilizar los documentos y/o planes de estudio una vez que haya obtenido una licencia de uso. Por favor, dirija cualquier consulta a [info@isaqb.org](mailto:info@isaqb.org). Existen acuerdos de licencia de uso con disposiciones detalladas que contemplan múltiples aspectos.
3. Si no pertenece a la categoría 1 ni a la categoría 2, pero desea utilizar estos documentos y/o programas de estudio de todas formas, póngase en contacto con el iSAQB e. V. escribiendo a [info@isaqb.org](mailto:info@isaqb.org). A continuación se le informará sobre la posibilidad de adquirir las licencias pertinentes mediante los acuerdos de licencia existentes, lo que le permitirá obtener las autorizaciones de uso que desee.

### Aviso importante

**Se destaca que, por principio, este programa de estudio está protegido por el derecho de autor. El International Software Architecture Qualification Board e. V. (iSAQB® e. V.) tiene el derecho exclusivo de estos derechos de autor.**

La abreviatura "e. V." forma parte del nombre oficial de iSAQB y significa "eingetragener Verein" (asociación registrada), que describe su condición de entidad legal según la ley alemana. En aras de la simplicidad, en adelante se denominará iSAQB e. V. sin el uso de dicha abreviatura.

## Relación de objetivos de aprendizaje

- OA 1-1: Debater acerca de las definiciones de la arquitectura de software (R1)
- OA 1-2: Comprender e identificar los beneficios de la arquitectura de software (R1)
- OA 1-3: Entender la arquitectura de software como parte del ciclo de vida del software (R2)
- OA 1-4: Comprender las tareas y responsabilidades de los arquitectos de software (R1)
- OA 1-5: Relacionar el papel de los arquitectos de software con otros implicados (R1)
- OA 1-6: Ser capaz de explicar la correlación entre los enfoques de desarrollo y la arquitectura de software (R1)
- OA 1-7: Diferenciar entre los objetivos a corto y largo plazo (R1)
- OA 1-8: Distinguir afirmaciones explícitas y supuestos implícitos (R1)
- OA 1-9: Responsabilidades de los arquitectos de software dentro de un contexto arquitectónico más amplio (R3)
- OA 1-10: Diferenciar los tipos de sistemas TI (R3)
- OA 2-1: Seleccionar y utilizar enfoques y heurísticas para el desarrollo de la arquitectura (R1- R3)
- OA 2-2: Diseñar arquitecturas de software (R1)
- OA 2-3: Identificar y considerar los factores que influyen en la arquitectura del software (R1-R2)
- OA 2-4: Diseñar e implementar asuntos de interés transversales (R1)
- OA 2-5: Describir, explicar y aplicar de forma adecuada patrones arquitectónicos relevantes (R1-R3)
- OA 2-6: Explicar y aplicar los principios de diseño (R1)
- OA 2-7: Planificar dependencias entre bloques constructivos (R1)
- OA 2-8: Lograr los requisitos de calidad con enfoques y técnicas apropiados (R1)
- OA 2-9: Diseñar y definir interfaces (R1-R3)
- OA 3-1: Explicar y considerar la calidad de la documentación técnica (R1)
- OA 3-2: Describir y comunicar arquitecturas de software (R1)
- OA 3-3: Explicar y aplicar notaciones/modelos para describir una arquitectura de software (R2)
- OA 3-4: Explicar y utilizar vistas arquitectónicas (R1)
- OA 3-5: Explicar y aplicar la vista de contexto de los sistemas (R1)
- OA 3-6: Documentar y comunicar asuntos transversales (R1)
- OA 3-7: Describir interfaces (R1)
- OA 3-8: Explicar y documentar las decisiones arquitectónicas (R2)
- OA 3-9: Utilizar la documentación como comunicación escrita (R2)
- OA 3-10: Conocer recursos y herramientas adicionales para la documentación (R3)
- OA 4-1: Debater sobre modelos de calidad y características de calidad (R1)
- OA 4-2: Precisar los requisitos de calidad de las arquitecturas de software (R1)

- OA 4-3: Llevar a cabo un análisis cualitativo y una evaluación de las arquitecturas de software (R2-R3)
- OA 4-4: Llevar a cabo una evaluación cuantitativa de las arquitecturas de software (R2)
- OA 5-1: Conocer la relación entre las necesidades, restricciones y soluciones (R3)
- OA 5-2: Conocer los fundamentos de la implementación técnica de una solución (R3)

## Introducción

### ¿Qué aporta una Formación a Foundation Level?

Los cursos de formación con licencia *Certified Professional for Software Architecture - Foundation Level* (CPSA-F) proporcionarán a los participantes los conocimientos y aptitudes necesarios para diseñar, especificar y documentar una arquitectura de software adecuada para cumplir los requisitos correspondientes a sistemas de tamaño pequeño y mediano. Los participantes aprenderán a tomar decisiones arquitectónicas en función de su experiencia práctica individual y de sus competencias actuales, a partir de una visión de sistema existente y de los requisitos adecuadamente detallados. En los cursos de formación del CPSA-F se enseñan métodos y principios para el diseño, la documentación y la evaluación de arquitecturas de software, independientemente de los procesos de desarrollo específicos.

La atención se centra en la educación y la formación de las siguientes competencias:

- Debatar y conciliar las decisiones arquitectónicas fundamentales con los implicados provenientes de requisitos, gestión, desarrollo, operaciones y prueba
- Comprender las actividades esenciales de la arquitectura de software y llevar a cabo aquellas correspondientes a los sistemas de tamaño pequeño y mediano
- Documentar y comunicar arquitecturas de software basadas en vistas arquitectónicas, patrones de arquitectura y conceptos técnicos

Adicionalmente, esta formación abarca:

- El concepto y el significado de la arquitectura de software
- Las tareas y responsabilidades de los arquitectos de software
- El papel de los arquitectos de software en los proyectos de desarrollo
- Métodos y técnicas correspondientes al estado del arte para desarrollar arquitecturas de software

## Fuera de alcance

Este programa de estudio refleja los contenidos que los miembros del iSAQB consideran, en la actualidad, necesarios y útiles para alcanzar los objetivos de aprendizaje del CPSA-F. No es una descripción exhaustiva de todo el dominio de la "arquitectura de software".

Los siguientes temas y conceptos **no forman parte del CPSA-F**:

- Tecnologías, marcos de trabajo o librerías de implementación específicos
- Programación o lenguajes de programación
- Modelos de proceso específicos
- Fundamentos de las notaciones de modelado (como el UML) o fundamentos del propio modelado
- Análisis de sistemas e ingeniería de requisitos (por favor, consulte el programa de estudio y certificación de IREB e. V., <http://ireb.org>, International Requirements Engineering Board)
- Prueba de software (por favor, consulte el programa de estudio y certificación de ISTQB e.V., <http://istqb.org>, International Software Testing Qualification Board)
- Gestión de proyectos o producto
- Introducción a herramientas de software específicas

El objetivo de la formación es transmitir los fundamentos para la adquisición de los conocimientos y habilidades avanzados necesarios para el caso de aplicación que corresponda.

## Prerrequisitos

El iSAQB e. V. puede comprobar los siguientes prerrequisitos en los exámenes de certificación mediante las preguntas correspondientes.

Los participantes deberán contar con los siguientes conocimientos y/o experiencia. En particular, la experiencia práctica sustancial en el desarrollo de software en equipo es un prerrequisito importante para la comprensión del material de aprendizaje y la obtención de la certificación.

- Más de 18 meses de experiencia práctica en el desarrollo de software, obtenida mediante el desarrollo en equipo de varios sistemas fuera de la formación académica.
- Conocimiento y experiencia práctica con al menos un lenguaje de programación de alto nivel, especialmente:
  - Conceptos de
    - Modularización (paquetes, espacio de nombres, etc)
    - Paso de parámetros (llamada por valor, llamada por referencia)
    - Alcance (es decir tipo), declaración y definición de variables
  - Fundamentos de sistemas de tipos (tipos estáticos frente a tipos dinámicos, tipos genéricos de datos)
  - Tratamiento de errores y excepciones en el software
  - Problemas potenciales de estado global y variables globales
- Conocimientos básicos de:
  - Modelado y abstracción
  - Algoritmos y estructuras de datos (por ejemplo, Listas, Árboles, Tablas Hash, Diccionario/Mapa)
  - UML (diagramas de clase, de paquetes, de componentes y de secuencia) y su relación con el código fuente

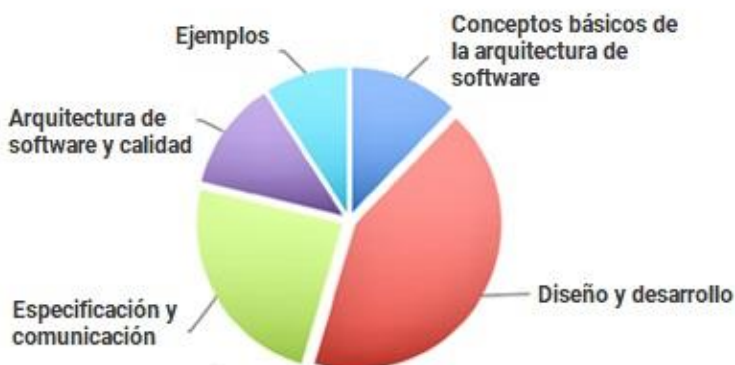
Además, los siguientes temas serán útiles para comprender diferentes conceptos:

- Fundamentos y diferencias de la programación imperativa, declarativa, orientada a objetos y funcional
- Experiencia práctica en
  - Un lenguaje de programación orientado a objetos (es decir, Java o C#)
  - Diseñar e implementar aplicaciones distribuidas, como sistemas cliente-servidor o aplicaciones web
  - Documentación técnica, especialmente la documentación del código fuente, el diseño del sistema o conceptos técnicos



## Estructura, duración y métodos de enseñanza

Los tiempos lectivos que se indican en las siguientes secciones del programa de estudio son sólo recomendaciones. La duración de un curso de formación debe ser de, al menos, tres días, pero también puede ser más prolongada. Los proveedores pueden variar la duración, los métodos de enseñanza, el tipo y la estructura de los ejercicios, así como el esquema detallado del curso. Los tipos (dominios y tecnologías) de los ejemplos y ejercicios pueden ser determinados individualmente por los proveedores de formación.



*Distribución recomendada del tiempo*

Índice	Duración recomendada (min)
1. Conceptos básicos de la arquitectura de software	120
2. Diseño y desarrollo	420
3. Especificación y comunicación	240
4. Arquitectura de software y calidad	120
5. Ejemplos	90
<b>Total</b>	<b>990</b>

## Objetivos de aprendizaje y su relevancia con respecto al examen

La estructura de los capítulos del programa de estudio sigue un conjunto de objetivos de aprendizaje priorizados. Para cada objetivo de aprendizaje, se indica claramente la relevancia para el examen de este objetivo de aprendizaje o sus subelementos (según la clasificación R1, R2, R3, véase el cuadro que figura a continuación). Cada objetivo de aprendizaje describe los contenidos que deben enseñarse, incluidos sus términos y conceptos clave.

En lo que respecta a la relevancia para el examen, en este plan de estudios se utilizan las siguientes categorías:

ID	Categoría del objetivo de aprendizaje	Significado	Relevancia para el examen
R1	Ser capaz de	Estos son los contenidos que se espera que los participantes puedan poner en práctica de forma independiente al finalizar el curso. Dentro del curso, estos contenidos serán cubiertos a través de ejercicios y debates.	El contenido <b>será</b> parte del examen.
R2	Comprensión	Estos son los contenidos que se espera que los participantes comprendan en principio. Normalmente no serán el foco principal de los ejercicios de la formación.	El contenido <b>puede ser</b> parte del examen.
R3	Conocimiento	Estos contenidos (términos, conceptos, métodos, prácticas o similares) pueden mejorar la comprensión y motivar el tema. Pueden ser cubiertos en la capacitación si fuera necesario.	El contenido <b>no será</b> parte del examen.

Cuando se considere necesario, los objetivos de aprendizaje incluirán referencias a lecturas adicionales, normas u otras fuentes. En las secciones "Términos y conceptos" de cada capítulo se enumeran las palabras que están asociadas al contenido del capítulo. Algunas de ellas se utilizan en las descripciones de los objetivos de aprendizaje.

## 1. Conceptos básicos de la arquitectura de software

Duración: 120 min.	Tiempo propuesto para ejercicios: sin tiempo
--------------------	--

### Términos fundamentales

**Arquitectura de software**; dominios de arquitectura; **estructura**; **bloque estructural**; **componente**; **interfaz**; **relación**; asunto transversal; arquitectos de software y sus responsabilidades; tareas y competencias necesarias; implicados y sus intereses; requisitos funcionales y de calidad de los sistemas; **restricción**; factor de influencia; tipos de sistemas TI (sistemas embebidos, sistemas en tiempo real, sistemas de información, etc.)

### Objetivos

#### OA 1-1: Debater acerca de las definiciones de la arquitectura de software (R1)

Los arquitectos de software tienen conocimiento de diversas definiciones en el ámbito de la arquitectura de software (incluyendo ISO 42010/IEEE 1471, SEI, Booch, etc.) y pueden señalar sus similitudes:

- Componentes/bloques estructurales con interfaces y relaciones
- Bloques estructurales como un término general, componentes como una forma especial de los mismos
- Estructuras, conceptos transversales, principios
- Decisiones respecto a la arquitectura y sus consecuencias sobre la totalidad de los sistemas y sus ciclos de vida

#### OA 1-2: Comprender e identificar los beneficios de la arquitectura de software (R1)

Los objetivos básicos de la arquitectura de software son:

- La consecución de los requisitos de calidad, como la fiabilidad, mantenibilidad, capacidad para ser modificado, seguridad, etc., y los requisitos funcionales
- Apoyar la creación y el mantenimiento de software, en particular su implementación
- Transmitir la comprensión de las estructuras y conceptos del sistema, en relación con todos los implicados relevantes

#### OA 1-3: Entender la arquitectura de software como parte del ciclo de vida del software (R2)

Los arquitectos de software entienden sus tareas y pueden integrar sus resultados a lo largo de todo el ciclo de vida de los sistemas TI. Ellos pueden:

- Identificar las consecuencias de los cambios en requisitos funcionales, requisitos de calidad, tecnologías o el entorno del sistema con respecto a la arquitectura de software
- Explicar detalladamente las relaciones entre los sistemas TI y los procesos de negocio y los procesos operativos soportados

#### OA 1-4: Comprender las tareas y responsabilidades de los arquitectos de software (R1)

Los arquitectos de software son responsables de lograr la calidad solicitada o necesaria y de crear el diseño de la arquitectura de una solución. Dependiendo del enfoque real o del modelo de

proceso utilizado, deben alinear esta responsabilidad con las responsabilidades globales de la gestión del proyecto y/u otros roles.

Las tareas y la responsabilidad de los arquitectos de software:

- Aclarar y examinar los requisitos y restricciones, y mejorarlos si fuera necesario. Junto con los requisitos funcionales (prestaciones necesarias), esto incluye las características de calidad requeridas (restricciones necesarias)
- Decidir cómo descomponer el sistema en bloques estructurales, al tiempo que se determinan las dependencias e interfaces entre los bloques estructurales
- Determinar y decidir respecto de asuntos transversales (por ejemplo, la persistencia, la comunicación, la interfaz gráfica de usuario, etc.)
- Comunicar y documentar la arquitectura de software basada en vistas, patrones arquitectónicos, conceptos transversales y técnicos
- Acompañar la realización e implementación de la arquitectura; integrar la retroalimentación de los implicados relevantes en la arquitectura si fuera necesario; revisar y asegurar la consistencia del código fuente y la arquitectura del software
- Analizar y evaluar la arquitectura de software, especialmente con respecto a los riesgos que implican los requisitos de calidad
- Identificar, resaltar y justificar las consecuencias de las decisiones arquitectónicas a otros implicados

Deberían reconocer, de forma independiente, la necesidad de las iteraciones en todas las tareas y señalar las posibilidades de una retroalimentación adecuada y relevante.

#### **OA 1-5: Relacionar el papel de los arquitectos de software con otros implicados (R1)**

Los arquitectos de software son capaces de explicar su rol. Deben adaptar su contribución al desarrollo de software en un contexto específico y en relación con otros implicados, en particular a:

- Análisis de los requisitos (análisis de sistemas, gestión de los requisitos, campo de especialización)
- Implementación
- Dirección y gestión del proyecto
- Gestión de producto, propietario de producto
- Aseguramiento de la calidad
- Las operaciones TI (producción, centros de datos), se aplican principalmente a los sistemas de información
- Desarrollo de hardware
- Arquitectura corporativa, comité de arquitectura

#### **OA 1-6: Ser capaz de explicar la correlación entre los enfoques de desarrollo y la arquitectura de software (R1)**

- Los arquitectos de software pueden explicar la influencia de los enfoques iterativos en las decisiones arquitectónicas (con respecto a los riesgos y la previsibilidad).
- Los arquitectos de software, a menudo, tienen que trabajar y tomar decisiones de forma iterativa debido a la incertidumbre inherente. Para ello, tienen que obtener retroalimentación en forma sistemática por parte de otros implicados.

**OA 1-7: Diferenciar entre los objetivos a corto y largo plazo (R1)**

Los arquitectos de software pueden:

- Explicar los requisitos de calidad a largo plazo y sus diferencias en relación con los objetivos del proyecto (a corto plazo)
- Explicar los posibles conflictos entre los objetivos a corto y a largo plazo, a fin de encontrar una solución adecuada para todos los implicados
- Identificar y especificar los requisitos de calidad

**OA 1-8: Distinguir afirmaciones explícitas y supuestos implícitos (R1)**

Los arquitectos de software:

- Deben exponer explícitamente los supuestos o prerrequisitos y evitar supuestos implícitos
- Deben saber que las suposiciones implícitas pueden dar lugar a posibles malentendidos entre los implicados

**OA 1-9: Responsabilidades de los arquitectos de software dentro de un contexto arquitectónico más amplio (R3)**

La atención del programa de estudio de Profesional Certificado del iSAQB® en Arquitectura de Software se centra en las estructuras y conceptos de sistemas de software individuales.

Adicionalmente, los arquitectos de software están familiarizados con otros dominios arquitectónicos, por ejemplo:

- Arquitectura ti corporativa: estructura de panoramas de aplicación
- Arquitectura de negocio y de procesos: estructura de, entre otras cosas, los procesos de negocio
- Arquitectura de la información: estructura transversal de los sistemas y uso de la información y los datos
- Infraestructura o arquitectura tecnológica: estructura de la infraestructura técnica, hardware, redes, etc
- Arquitectura del hardware o del procesador (para sistemas relacionados con el hardware)

Estos dominios arquitectónicos no son el foco de contenido del CPSA-F.

**OA 1-10: Diferenciar los tipos de sistemas TI (R3)**

Los arquitectos de software conocen diferentes tipos de sistemas informáticos, por ejemplo:

- Sistemas de información
- Sistemas de soporte a decisiones, almacenes de datos o inteligencia de negocio
- Sistemas móviles
- Procesos o sistemas por lotes
- Sistemas relacionados con el hardware; aquí entienden la necesidad de diseño del código de hardware/software (dependencias temporales y relacionadas con el contenido del diseño del hardware y el software)

## 2. Diseño y desarrollo de arquitecturas de software

Duración: 330 min.

Tiempo propuesto para ejercicios: 90 min.

### Términos fundamentales

Patrón arquitectónico; enfoque ascendente; cohesión; acoplamiento; interés transversal; dependencia; diseño; enfoque de diseño; decisión de diseño; principio de diseño; diseño guiado por el dominio; arquitectura funcional y técnica; arquitectura funcional; interfaz; diseño iterativo/incremental; diseño basado en modelos; lenguaje de patrón; estereotipo; intereses técnicos y transversales; arquitectura técnica; interés técnico; enfoques descendente y ascendente; enfoque descendente; vista

### Objetivos

#### OA 2-1: Seleccionar y utilizar enfoques y heurísticas para el desarrollo de la arquitectura (R1-R3)

Los arquitectos de software son capaces de mencionar, explicar y utilizar los enfoques básicos del desarrollo de la arquitectura, por ejemplo:

- Enfoques de diseño descendente y ascendente
- Desarrollo de arquitectura basada en vistas
- Diseño iterativo e incremental
  - Necesidad de las iteraciones, especialmente cuando la toma de decisiones se ve afectada por incertidumbres
  - Necesidad de retroalimentación sobre las decisiones de diseño
- Diseño guiado por el dominio (R3)
- Arquitectura guiada por modelos (R3)

#### OA 2-2: Diseñar arquitecturas de software (R1)

Los arquitectos de software tienen la capacidad para:

- Diseñar y comunicar y documentar, de forma adecuada, arquitecturas de software basadas en requisitos funcionales y de calidad conocidos para sistemas de software que no son críticos para la seguridad ni para el negocio
- Tomar decisiones relevantes para la estructura con respecto a la descomposición del sistema, la estructura de los bloques estructurales y diseñar, de forma deliberada, las dependencias entre los bloques estructurales
- Reconocer y justificar las interdependencias y compromisos de las decisiones de diseño
- Explicar los términos *caja negra* y *caja blanca* y aplicarlos de forma deliberada
- Aplicar un refinamiento de forma gradual y especificar los bloques estructurales
- Diseñar vistas de arquitectura, especialmente la vista de bloques estructurales, la vista de tiempo de ejecución y la vista de despliegue
- Aclarar las consecuencias que tienen estas decisiones de cara al código fuente
- Separar los elementos técnicos y los relacionados con el dominio de las arquitecturas y justificar estas decisiones

- Identificar los riesgos de las decisiones de diseño

### **OA 2-3: Identificar y considerar los factores que influyen en la arquitectura del software (R1-R2)**

Los arquitectos de software son capaces de recopilar y tener en cuenta las restricciones y factores de influencia que limiten la libertad en las decisiones de diseño.

Deben reconocer y tener en cuenta:

- El impacto de los requisitos de calidad
- Impacto de las decisiones y conceptos técnicos
- Impacto (potencial) de las restricciones propias de la organización, legales y de otro tipo (R2)
- Impacto de los intereses de los implicados (R2)

### **OA 2-4: Diseñar e implementar asuntos de interés transversales (R1)**

Los arquitectos de software son capaces de:

- Explicar el significado de tales asuntos de interés transversales
- Decidir y diseñar con respecto a los asuntos de interés transversales, por ejemplo, la persistencia, la comunicación, la interfaz gráfica de usuario, el tratamiento de los errores, la concurrencia
- Identificar y valorar las posibles interdependencias entre estas decisiones

Los arquitectos de software saben que estos conceptos transversales pueden ser reutilizados en todo el sistema.

### **OA 2-5: Describir, explicar y aplicar de forma adecuada patrones arquitectónicos relevantes (R1-R3)**

Los arquitectos de software conocen diversos patrones arquitectónicos y pueden utilizarlos según sea necesario:

- Capas (R1)
- Tuberías y filtros (R1)
- Cliente/servidor (R1)
- Adaptador y fachada (R1)
- Subrogación (R1)
- Extensión (R1)
- Pizarra (R2)
- Controlador de vista de modelo y variantes (R2)
- Intermediario (R2)
- Llamada de procedimiento remoto (R2)
- Mensajería (R2), por ejemplo, con eventos y comandos

Los arquitectos de software conocen fuentes fundamentales para los patrones arquitectónicos, como POSA (por ejemplo, [Buschmann+1996]) y PoEAA ([Fowler 2003]) (para los sistemas de información) (R3)

### OA 2-6: Explicar y aplicar los principios de diseño (R1)

Los arquitectos de software son capaces de explicar y aplicar los siguientes principios de diseño:

- Abstracción
- Modularización, con respecto a:
  - Ocultación y encapsulación de información
  - Separación de asuntos de interés
  - Alta cohesión
- Responsabilidad única
- Principio de apertura y cierre
- Bajo acoplamiento, pero funcionalmente suficiente de los bloques estructurales (véase [OA 2-7](#))
- Inversión de la dependencia mediante interfaces o abstracciones similares
- Integridad conceptual para lograr la uniformidad (homogeneidad, consistencia) de las soluciones para problemas similares (R2)

Los arquitectos de software comprenden el impacto de los principios de diseño en el código fuente y son capaces de aplicarlos de forma adecuada.

### OA 2-7: Planificar dependencias entre bloques constructivos (R1)

Los arquitectos de software comprenden las dependencias y el acoplamiento entre los bloques estructurales y pueden utilizarlos de forma controlada. Ellos:

- Conocen diferentes tipos de dependencias de los bloques estructurales (por ejemplo, el acoplamiento estructural a través de la utilización/delegación, la anidación, la propiedad, la generación, la herencia, el acoplamiento temporal, el acoplamiento a través de los tipos de datos o a través del hardware)
- Pueden utilizar esos tipos de acoplamiento de manera selectiva y pueden evaluar las consecuencias de esas dependencias
- Conocen y pueden aplicar las posibilidades de reducir o eliminar el acoplamiento, por ejemplo:
  - Patrones (véase [OA 2-5](#))
  - Principios básicos del diseño (véase [OA 2-6](#))
  - Externalización de dependencias, es decir, definir dependencias concretas en el momento de la instalación o en tiempo de ejecución, por ejemplo, mediante la inyección de dependencias.

### OA 2-8: Lograr los requisitos de calidad con enfoques y técnicas apropiados (R1)

Los arquitectos de software comprenden y tienen en cuenta la considerable influencia de los requisitos de calidad en las decisiones de arquitectura y diseño, por ejemplo, para:

- La eficiencia o el rendimiento
- La disponibilidad
- Mantenibilidad, capacidad de ser modificado, extensibilidad, adaptabilidad



Ellos pueden:

- Explicar y aplicar las opciones de solución, las tácticas de diseño, las prácticas adecuadas, así como las posibilidades técnicas para lograr importantes requisitos de calidad de los sistemas software (diferentes para los sistemas embebidos o los sistemas de información)
- Identificar y comunicar los posibles compromisos entre esas soluciones y sus riesgos asociados

### **OA 2-9: Diseñar y definir interfaces (R1-R3)**

Los arquitectos de software conocen la importancia de las interfaces. Son capaces de diseñar o especificar interfaces entre bloques estructurales arquitectónicos, así como interfaces externas entre el sistema y elementos ajenos al sistema.

Ellos conocen:

- Las características deseadas de las interfaces y pueden utilizarlas en el diseño:
  - Fácil de aprender, fácil de utilizar, fácil de ampliar
  - Difícil de utilizar de forma incorrecta
  - Funcionalmente completo desde la perspectiva de los usuarios o de los bloques estructurales que las utilizan
- La necesidad de tratar las interfaces internas y externas de manera diferente
- Diferentes enfoques para implementar las interfaces (R3):
  - Enfoque orientado a los recursos (REST, REpresentational State Transfer)
  - Enfoque orientado a los servicios (véase servicios web basados en WS-\*/SOAP)

### **Referencias**

[\[Bass+2012\]](#), [\[Fowler 2003\]](#), [\[Gharbi+2017\]](#), [\[Gamma+94\]](#), [\[Martin 2003\]](#), [\[Buschmann+1996\]](#) y [\[Buschmann+2007\]](#), [\[Starke 2017\]](#), [\[Lilienthal 2017\]](#)

### 3. Especificación y comunicación de arquitecturas de software

Duración: 180 min.	Tiempo propuesto para ejercicios: 60 min.
--------------------	---

#### Términos fundamentales

Vista (arquitectónica); estructura; concepto (técnico); documentación; comunicación; descripción; orientado a los implicados; meta estructuras y plantillas para la descripción y la comunicación; contexto del sistema; bloque estructural; vista de bloque estructural; vista en tiempo de ejecución; vista de despliegue; nodo; canal; artefacto de despliegue; mapear bloques estructurales en los artefactos de despliegue; descripción de las interfaces y decisiones de diseño; UML; herramienta para la documentación

#### Objetivos

##### OA 3-1: Explicar y considerar la calidad de la documentación técnica (R1)

Los arquitectos de software conocen los requisitos de calidad de la documentación técnica y pueden tenerlos en cuenta y cumplirlos al documentar los sistemas:

- Capacidad de ser comprendido, corrección, eficiencia, adecuación, mantenibilidad
- Forma, contenido y nivel de detalle se adaptan a los implicados

Saben que sólo la audiencia objetivo puede evaluar la capacidad de ser comprendida de la documentación técnica.

##### OA 3-2: Describir y comunicar arquitecturas de software (R1)

Los arquitectos de software están capacitados para:

- Documentar y comunicar las arquitecturas a los implicados correspondientes, dirigiéndose así a los diferentes grupos objetivo, por ejemplo, la dirección, los equipos de desarrollo, el control de calidad, otros arquitectos de software y posiblemente implicados adicionales
- Consolidar y armonizar el estilo y el contenido de las contribuciones de los diferentes grupos de autores
- Conocer los beneficios de la documentación basada en plantillas

##### OA 3-3: Explicar y aplicar notaciones/modelos para describir una arquitectura de software (R2)

Los arquitectos de software conocen al menos los siguientes diagramas UML para describir las vistas arquitectónicas:

- Diagramas de clase, paquete, componente y composición-estructura
- Diagramas de despliegue
- Diagramas de secuencia y actividad
- Diagramas de estado

Los arquitectos de software conocen notaciones alternativas a los diagramas UML, especialmente para la vista de tiempo de ejecución, por ejemplo, diagramas de flujo, listas numeradas, BPMN.

**OA 3-4: Explicar y utilizar vistas arquitectónicas (R1)**

Los arquitectos de software son capaces de utilizar las siguientes vistas arquitectónicas:

- Vista de contexto
- Vista de bloque estructural o componente (composición de bloques estructurales de software)
- Vista en tiempo de ejecución (vista dinámica, interacción entre los bloques estructurales en tiempo de ejecución, máquinas de estado)
- Vista de despliegue (infraestructura de hardware y técnica, así como el mapeo de los bloques estructurales en la infraestructura)

**OA 3-5: Explicar y aplicar la vista de contexto de los sistemas (R1)**

Los arquitectos de software pueden:

- Describir el contexto de los sistemas, por ejemplo, en forma de diagramas de contexto con explicaciones
- Representar las interfaces externas de los sistemas en la vista de contexto
- Diferenciar el contexto funcional y técnico

**OA 3-6: Documentar y comunicar asuntos transversales (R1)**

Los arquitectos de software son capaces de documentar y comunicar adecuadamente los asuntos de interés transversal típicos, por ejemplo, la persistencia, la gestión del flujo de trabajo, la interfaz de usuario, el despliegue/integración, el registro.

**OA 3-7: Describir interfaces (R1)**

Los arquitectos de software son capaces de describir y especificar tanto las interfaces internas como las externas.

**OA 3-8: Explicar y documentar las decisiones arquitectónicas (R2)**

Los arquitectos de software son capaces de:

- Tomar, justificar, comunicar y documentar de forma sistemática decisiones arquitectónicas
- Identificar, comunicar y documentar las interdependencias entre las decisiones de diseño

**OA 3-9: Utilizar la documentación como comunicación escrita (R2)**

Los arquitectos de software utilizan la documentación para dar soporte al diseño, la implementación y el desarrollo ulterior (también llamado mantenimiento o evolución) de los sistemas.

**OA 3-10: Conocer recursos y herramientas adicionales para la documentación (R3)**

Los arquitectos de software conocen:

- Fundamentos de diferentes marcos de trabajo públicos para la descripción de las arquitecturas de software, por ejemplo:
  - ISO/IEEE-42010 (antes 1471)

- arc42
- C4
- RM-ODP
- FMC
- Ideas y ejemplos de listas de control para la creación, documentación y prueba de arquitecturas de software
- Posibles herramientas para crear y mantener la documentación arquitectónica

## Referencias

[\[Bass+2012\]](#), [\[Clements+2010\]](#), [\[Gharbi+2017\]](#), [\[Starke 2017\]](#), [\[Zörner 2015\]](#)

## 4. Arquitectura de software y calidad

Duración: 60 min.	Tiempo propuesto para ejercicios: 60 min.
-------------------	---

### Términos fundamentales

Calidad; características de calidad (también llamados atributos de calidad); DIN/ISO 25010; escenarios de calidad; árbol de calidad; compromisos entre las características de calidad; valoración cualitativa de la arquitectura; métricas y valoración cuantitativa

### Objetivos

#### OA 4-1: Debatar sobre modelos de calidad y características de calidad (R1)

Los arquitectos de software pueden:

- Explicar el concepto de calidad (basado en la norma DIN/ISO 25010, anteriormente 9126) y las características de calidad
- Explicar los modelos de calidad genéricos (como DIN/ISO 25010)
- Explicar las correlaciones y compromisos de las características de calidad, por ejemplo:
  - Configurabilidad frente a fiabilidad
  - Requisitos de memoria frente a la eficiencia de desempeño
  - Seguridad frente a usabilidad
  - Flexibilidad en tiempo de ejecución frente a mantenibilidad

#### OA 4-2: Precisar los requisitos de calidad de las arquitecturas de software (R1)

Los arquitectos de software pueden:

- Aclarar y formular requisitos de calidad específicos para el software que se va a desarrollar y sus arquitecturas, por ejemplo, en forma de escenarios y árboles de calidad
- Explicar y aplicar escenarios y árboles de calidad

#### OA 4-3: Llevar a cabo un análisis cualitativo y una evaluación de las arquitecturas de software (R2-R3)

Los arquitectos de software:

- Conocen los enfoques metódicos para el análisis cualitativo y la evaluación de las arquitecturas de software (R2), por ejemplo, como se especifica en ATAM (R3)
- Pueden analizar y evaluar cualitativamente sistemas más pequeños (R2)
- Saben que las siguientes fuentes de información pueden ayudar en el análisis cualitativo y la evaluación de las arquitecturas (R2):
  - Requisitos de calidad, por ejemplo, en forma de árboles y escenarios de calidad
  - Documentación de la arquitectura
  - Arquitectura y modelos de diseño

- Código fuente
- Métricas
- Otra documentación del sistema, como los requisitos, la documentación operativa o de prueba

#### **OA 4-4: Llevar a cabo una evaluación cuantitativa de las arquitecturas de software (R2)**

Los arquitectos de software conocen los enfoques para el análisis cuantitativo y la evaluación (medición) del software.

Ellos saben que:

- La evaluación cuantitativa puede contribuir a identificar las partes críticas dentro de los sistemas
- Información adicional puede ser útil para la evaluación de las arquitecturas, por ejemplo:
  - Documentación de requisitos y arquitectura
  - Código fuente y métricas relacionadas como líneas de código, complejidad (ciclométrica), dependencias de entrada y salida
  - Defectos conocidos en el código fuente, especialmente en las agrupaciones de defectos
  - Casos de prueba y resultados de prueba

#### **Referencias**

[\[Bass2003\]](#), [\[Clements+2002\]](#), [\[Gharbi+2017\]](#), [\[Martin 2003\]](#), [\[Starke 2017\]](#)

## 5. Ejemplos de arquitecturas de software

Duración: 90 min.	Tiempo propuesto para ejercicios: sin tiempo
-------------------	--

Esta sección no es relevante para el examen.

### Objetivos

#### **OA 5-1: Conocer la relación entre las necesidades, restricciones y soluciones (R3)**

Los arquitectos de software han utilizado por lo menos un ejemplo para identificar y comprender la relación entre los requisitos y las restricciones, y las decisiones de solución.

#### **OA 5-2: Conocer los fundamentos de la implementación técnica de una solución (R3)**

Los arquitectos de software entienden la realización técnica (implementación, conceptos técnicos, productos utilizados, decisiones arquitectónicas, estrategias de solución) de al menos una solución.

## Referencias

- [Bass+2012] Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice. 3<sup>rd</sup> Edition, Addison Wesley 2012.
- [Buschmann+1996] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal: Pattern-Oriented Software Architecture (POSA): A System of Patterns. Wiley, 1996.
- [Buschmann+2007] Frank Buschmann, Kevlin Henney, Douglas C. Schmidt: Pattern-Oriented Software Architecture (POSA): A Pattern Language for Distributed Computing, Wiley, 2007.
- [Clements+2002] Paul Clements, Rick Kazman, Mark Klein: Evaluating Software Architectures. Methods and Case Studies. Addison Wesley, 2002.
- [Clements+2010] Paul Clements, Felix Bachmann, Len Bass, David Garlan, David, James Ivers, Reed Little, Paulo Merson and Robert Nord. Documenting Software Architectures: Views and Beyond, 2<sup>nd</sup> edition, Addison Wesley, 2010.
- [Fowler 2003] Martin Fowler: Patterns of Enterprise Application Architecture. (PoEAA) Addison- Wesley, 2003.
- [Gamma+94] Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1994.
- [Gharbi+2017] Mahboub Gharbi, Arne Koschel, Andreas Rausch, Gernot Starke: Basiswissen Softwarearchitektur. [Conocimientos básicos de la arquitectura de software.] 3.<sup>a</sup> edición, dpunkt Verlag, Heidelberg 2017.
- [Goll 2014] Joachim Goll: Architektur- und Entwurfsmuster der Softwaretechnik: Mit lauffähigen Beispielen in Java. [Patrones arquitectónicos y de diseño en la ingeniería de software. Con ejemplos ejecutables en Java.] Springer-Vieweg Verlag, 2.<sup>a</sup> edición 2014.
- [iSAQB Working Groups] Gernot Starke et. al. Annotated collection of Software Architecture References, for Foundation and Advanced Level Curricula. Freely available <https://leanpub.com/isaqbreferences>.
- [Lilienthal 2017] Carola Lilienthal: Langlebige Softwarearchitekturen. [Arquitectura de software duradera.] 2.<sup>a</sup> edición, dpunkt Verlag 2018.
- [Martin 2003] Robert Martin: Agile Software Development. Principles, Patterns, and Practices. Prentice Hall, 2003.
- [Starke 2017] Gernot Starke: Effektive Softwarearchitekturen - Ein praktischer Leitfaden. [Eficiencia en la arquitectura de software. Una guía práctica.] 8.<sup>a</sup> edición, Carl Hanser Verlag 2017. Página web: <https://esabuch.de>
- [Zörner 2015] Stefan Zörner: Softwarearchitekturen dokumentieren und kommunizieren. [Documentar y comunicar la arquitectura de software.] 2.<sup>a</sup> edición, Carl Hanser Verlag 2015.