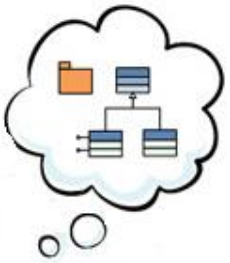
Máster Universitario de Investigación en Ingeniería

de Software y Sistemas Informáticos

TRABAJO FIN DE MASTER



Código (31105151) ARQUITECTURAS PARA SISTEMAS SOFTWARE

"ArchE: Arquitecturas y Toma de decisiones. Aplicación al Análisis de Arquitecturas Software para Sistemas E-learning"

Autor: Ángel Luis Álvarez Martín

Director: José Félix Estívariz López

Curso académico 2017/18

Convocatoria: Septiembre

Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos

Código (31105151) ARQUITECTURAS PARA SISTEMAS SOFTWARE

Título: "ArchE: Arquitecturas y Toma de decisiones. Aplicación al Análisis de Arquitecturas Software para Sistemas E-learning"

Trabajo tipo A

Estudiante: Ángel Luis Álvarez Martín

Director: José Félix Estívariz López

DECLARACIÓN JURADA DE AUTORÍA DEL TRABAJO CIENTÍFICO, PARA LA DEFENSA DEL TRABAJO FIN DE MASTER

Fecha: 7/9/2018

Quién suscribe:

Autor(a): Ángel Luis Álvarez Martín
D.N.I/N.I.E/Pasaporte.: 71027006Q

Hace constar que es la autor(a) del trabajo:

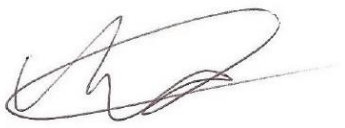
ArchE: Arquitecturas y Toma de decisiones. Aplicación al Análisis de Arquitecturas Software para Sistemas E-learning

En tal sentido, manifiesto la originalidad de la conceptualización del trabajo, interpretación de datos y la elaboración de las conclusiones, dejando establecido que aquellos aportes intelectuales de otros autores, se han referenciado debidamente en el texto de dicho trabajo.

DECLARACIÓN:

- ✓ Garantizo que el trabajo que remito es un documento original y no ha sido publicado, total ni parcialmente por otros autores, en soporte papel ni en formato digital.
- ✓ Certifico que he contribuido directamente al contenido intelectual de este manuscrito, a la génesis y análisis de sus datos, por lo cual estoy en condiciones de hacerme públicamente responsable de él.
- ✓ No he incurrido en fraude científico, plagio o vicios de autoría; en caso contrario, aceptaré las medidas disciplinarias sancionadoras que correspondan.

Fdo.



IMPRESO TFDM05_AUTORPBL
AUTORIZACIÓN DE PUBLICACIÓN
CON FINES ACADÉMICOS

Impreso TFdM05_AutorPbl. Autorización de publicación y difusión del TFM para fines académicos

Autorización

Autorizo/amos a la Universidad Nacional de Educación a Distancia a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firma del/los Autor/es

Juan del Rosal, 16
28040, Madrid
Tel: 91 398 89 10
Fax: 91 398 89 09
www.lsi.uned.es

RESUMEN

A la hora de diseñar arquitecturas software existe una serie de atributos especiales denominados *atributos de calidad*. Estos atributos son importantes para entender las características deseadas en el producto desde la definición de requisitos. Logrando que nuestro diseño sea capaz de cumplir con las especificaciones para dichos atributos obtendremos arquitecturas que se adapten realmente a las necesidades del cliente. Estos atributos en lo relacionado con la gestión del proyecto son importantes para: definir el enfoque en la planificación del proyecto, establecer los recursos que se requieren en los diferentes entornos para el desarrollo, para el control estadístico del proceso y como medida de consecución de los objetivos del proyecto.

Debido a la envergadura de la mayoría de los proyectos se han creado herramientas de ayuda a la toma de decisiones de diseño que nos aportan ideas sobre cómo enfocar mejor nuestros proyectos. Usando como medida para ello los llamados atributos de calidad. Convirtiéndose estas en grandes aliadas de los ingenieros de software.

El presente trabajo aborda el uso de la herramienta ArchE, un sistema experto en toma de decisiones de diseño. Para ello usaremos como ejemplo el diseño de una aplicación e-learning. Una vez evaluada nuestra arquitectura candidata mediante el planteamiento de diversos escenarios de calidad. Observaremos las tácticas de mejora que Arche nos propone y las implementaremos para que así el diseño satisfaga los criterios de calidad planteados. Demostrando así la utilidad de este tipo de herramientas.

PALABRAS CLAVE

Ingeniería del software, ArchE, e-learning, atributos de calidad, escenarios de calidad, toma de decisiones, diseño de arquitecturas software

CONTENIDO

1. Introducción, objetivos y estructura del trabajo 15
1.1 Introducción 15
1.2 Objetivos y estructura 16
2. Arquitecturas software para e-learning. 17
2.1 Introducción a las plataformas e-learning 17
2.2 Ejemplo de diseño de un sistema software de e-learning para desarrollo de código. 19
2.2.1 Toma de requisitos 20
2.3 Arquitectura y vistas 25
2.3.1 Marco de referencia para la descripción de la arquitectura. 25
2.3.2 Vista lógica 26
2.3.3 Vista de proceso 28
2.3.4 Vista Física 30
2.3.5 Vista desarrollo 31
2.3.6 Vista escenarios 32
3. Caso de estudio particular y análisis con Arche. 33
3.1 Definición del sistema 33
3.1.1 Identificación de actores principales y objetivos: 33
3.2 Arquitectura del sistema e-learning 47
3.2.1 Escenarios de calidad 50
3.3 Análisis con Arche 55
3.3.1 Definimos funciones y responsabilidades 55
3.3.2 Definimos las relaciones 57
3.3.3 Definición de los escenarios 59
3.3.4 Mapeo de escenarios a responsabilidades 60
3.3.5 Definimos las responsabilidades 62
3.3.5 Evaluación con Arche y aplicación de tácticas 63
3.4 Resultados 72
3.5 Valoración de los resultados 73
4. Conclusiones y trabajos futuros. 76
4.1 Conclusiones 76
4.2 Trabajos futuros 78
5. Bibliografía. 79
6. Glosario. 80

ILUSTRACIONES

Ilustración 1. Características generales e-learning 17
Ilustración 2. Arquitectura Moodle 19
Ilustración 3. Modelo conceptual de un espacio virtual de enseñanza 20
Ilustración 4. IEEE 1471 descripción 25
Ilustración 5. Modelo Krunchen 4+1 25
Ilustración 6. Diagrama de clases propuesto 26
Ilustración 7. Diagrama de secuencia para buscar código y editar. 28
Ilustración 8. Diagrama de secuencia para petición de reunión 29
Ilustración 9. División en capas de sistema e-learning 30
Ilustración 10. Diagrama de componentes 31
Ilustración 11. Diagrama de caso de uso 32
Ilustración 12. Diagrama casos de uso e-learning 34
Ilustración 13. Arquitectura por capas 47
Ilustración 14. Colaboración entre componentes MVC 48
Ilustración 15. Diagrama de paquetes E-LEARNING 48
Ilustración 16. Diagrama de despliegue 49
Ilustración 17. Tácticas mejora de rendimiento 51
Ilustración 18. Tácticas de modificabilidad 53
Ilustración 19. Nuevo proyecto ArchE 55
Ilustración 20. Introducimos funciones 55
Ilustración 21. Responsabilidades 56
Ilustración 22. Función – responsabilidad 56
Ilustración 23. Gráfico de responsabilidades 57
Ilustración 24. Cargamos relaciones en Arche 58
Ilustración 25. Cargamos escenarios 1 59
Ilustración 26. Cargamos escenarios 2 60
Ilustración 27. Mapeado escenarios responsabilidades 61
Ilustración 28. Responsabilidades definidas con costo de cambio y tiempo de ejecución 62
Ilustración 29. Evaluación 63
Ilustración 30. Tácticas y sugerencias Arche 64
Ilustración 31. Vista calificación de las tácticas 65
Ilustración 32. Usamos táctica 2 66
Ilustración 33. Recomendaciones y asignación de coste 66
lustración 34. Vista del módulo dividido 67
Ilustración 35. Redefinir costes tras la división 67
Ilustración 36. Seguimos sin cumplir el escenario 68
Ilustración 37. Aplicar táctica 1 69
Ilustración 38. Revisar responsabilidades 70
Ilustración 39. Todos los escenarios cumplidos 70
Ilustración 40. MAST 71

Ilustración 41. Diagrama original 72
Ilustración 42. Diagrama resultado 72
Ilustración 43. Arquitectura tras los cambios 75
Ilustración 44. Palladio 78

TABLAS

Tabla 1. Usuarios – intereses 22
Tabla 2. Tabla de requisitos del sistema. 22
Tabla 3. Caso de uso de edición 32
Tabla 4. Caso de uso 1 Registrar Docente 35
Tabla 5. Caso de uso 2 Registrar cursos 35
Tabla 6. Caso de uso 3 modificar 35
Tabla 7. Caso de uso 4 Consulta información académica 35
Tabla 8. Caso de uso 5 Consultar/ crear eventos 36
Tabla 9. Caso de uso 6 consultar notas 36
Tabla 10. Caso de uso 7 consultar materias 36
Tabla 11. Caso de uso 8 Generar informes 37
Tabla 12. Caso de uso 9 acreditar alumnos y profesores 37
Tabla 13. Caso de uso 10 preinscripción en cursos 37
Tabla 14. Caso de uso 11 Información académica 38
Tabla 15. Caso de uso 12 visualización de eventos 38
Tabla 16. Caso de uso 13 Consulta información académica 38
Tabla 17. Caso de uso 13 Consulta información académica 38
Tabla 18. Caso de uso 15 consultar materias 39
Tabla 19. Caso de uso 16 Visualizar cursos en los que está matriculado 39
Tabla 20. Caso de uso 17 descarga material del curso 39
Tabla 21. Caso de uso 18 Participación en Foros 40
Tabla 22. Caso de uso 19 Realizar inscripciones 40
Tabla 23. Caso de uso 20 Registrar estudiantes 40
Tabla 24. Caso de uso 21 Registrar curos 41
Tabla 25. Caso de uso 22 Crear/consultar información académica 41
Tabla 26. Caso de uso 23 Crear eventos 41
Tabla 27. Caso de uso 24 Crear/consultar notas 41
Tabla 28. Caso de uso 25 Crear/consultar materia 42
Tabla 29. Caso de uso 26 Modificar cursos 42
Tabla 30. Caso de uso 27 modificar datos personales 42
Tabla 31. Caso de uso 28 Gestionar cursos 43
Tabla 32. Caso de uso 29 Gestionar descargas 43
Tabla 33. Caso de uso 30 Participar en foros 43
Tabla 34. Caso de uso 31 Crear/consultar información académica 44

1. Introducción, objetivos y estructura del trabajo.

1.1 Introducción

A la hora de diseñar una arquitectura software partimos de una serie de requerimientos, ligados a estos requerimientos existen unos atributos denominados **atributos de calidad.** Es importante entenderlos desde la concepción de la idea a partir de las necesidades del cliente o mercado, considerarlos como parte de la solución y creación del producto para finalmente demostrar que han sido adecuadamente integrados en el producto final. Estos atributos son clave para obtener una buena arquitectura que les de soporte y esta pueda satisfacer a todos actores implicados en su ciclo de vida.

A la vista de esta realidad, en el SEI (Software Engineering Institute) surge el proyecto ArchE. Se trata de una herramienta que usa marcos de razonamiento para analizar arquitecturas software y proponer tácticas para mejorarlas.

Para la realización de este trabajo he decidió usar como arquitectura de referencia, un sistema e-learning a fin de analizarla con ArchE. Mediante el cual realizaremos una evaluación y las mejoras pertinentes.

1.2 Objetivos y estructura

Los objetivos y la estructura en que se desarrolla el trabajo es la siguiente:

1. En primer lugar realizaremos una *introducción a las plataformas e-learning*, acompañado de un ejemplo de diseño. Con el fin de estudiar algunas de las particularidades de este tipo de sistemas.

2. *Caso de estudio de un sistema e-learning y evaluación con Arche*: partiendo de cero propondré unos casos de uso de un sistema e-learning, de los cuales extraeremos los requisitos y a continuación propondré una arquitectura software que les de soporte. Proponiendo como objetivos de mejora los siguientes aspectos, atendiendo a los *atributos de calidad* ponderables con la herramienta Arche:

    - Rendimiento: rapidez con la que se ejecutan las funciones críticas de la arquitectura.
    - Modificabilidad: tiempo necesario para modificar los diferentes módulos que componen la arquitectura.

    Seguidamente evaluaremos con Arche dicha arquitectura en diferentes escenarios propuestos para los atributos de calidad anteriormente descritos. Y aplicaremos mejoras al diseño derivadas de las tácticas propuestas por Arche. Con el objetivo de obtener mejoras en nuestro diseño inicial.

3. Conclusiones extraidas de la realización del presente trabajo y exposición de resultados. Futuras líneas de investigación.

2. Arquitecturas software para e-learning.

Comenzaremos con una pequeña introducción y seguidamente con un ejemplo de una plataforma e-learning de desarrollo de código fuente.

2.1 Introducción a las plataformas e-learning

[5] En 1962, R. Buckminster Fuller publica su visión de la enseñanza y el aprendizaje en su artículo que título "Educación Automática". En el cual, este conjetura con que el futuro de la educación estará fuertemente condicionada por la tecnología y se caracterizará por no tener límites geográficos o temporales. Podríamos decir que es la primera aproximación a las características de los sistemas e-learning de la que se tiene constancia.

Estas plataformas e-learning constituyen actualmente una realidad tecnológica y que da soporte a la enseñanza tradicional e incluso la sustituye.

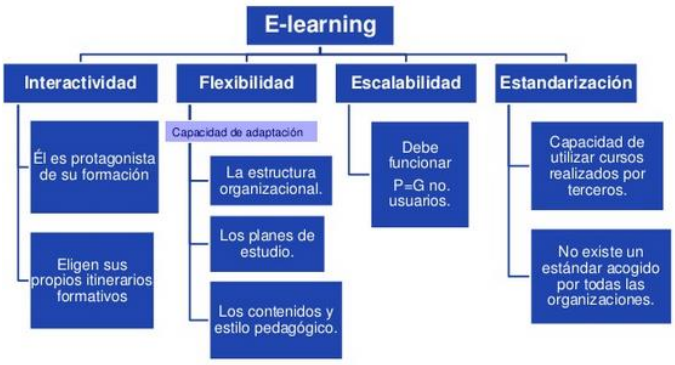


Ilustración 1. Características generales e-learning

Una plataforma e-elarning es una aplicación software que integra diferentes herramientas de gestión, comunicación, evaluación, etc. Con el fin de proveer soporte tecnológico a los profesores y alumnos en la educación online.

Dependiendo de las funcionalidades y objetivos de estas plataformas las podemos clasificar en:

- **CMS** (Content Management System): Es el más básico y sirve para gestionar contenidos a través del sistema.

- **LMS** (Learning Management System): Centrado especialmente en el área de la educación. Permite controlar los contenidos y como los usuarios interactúan con ellos.

- **LCMS** (Learning Content Management System): Integra los beneficios de los dos sistemas anteriores.

Las herramientas incorporadas en los sistemas e-learning se dividen en dos subgrupos:

- La parte pública cursos y utilidades para el estudiante.
- La parte privada herramientas administrativas.

Estas herramientas podrán ser accedidas dependiendo del rol "privilegios" que cada usuario tenga otorgados. Los elementos con los que usualmente cuentan estas plataformas son:

1) Servicios y Áreas de Interacción educativa: Son los espacios virtuales donde los usuarios interactúan con el sistema de gestión del aprendizaje (LMS): contenidos, herramientas de comunicación, tutores y compañeros.

2) Herramientas de Comunicación: Las herramientas de comunicación son aquellas aplicaciones que permiten a los distintos usuarios opinar, preguntar, hablar, etc., en definitiva, interactuar o relacionarse con otros, ya sea a través de texto, imágenes o sonidos.
   Existen 2 tipos de herramientas de comunicación:
   - *Asíncronas*
     -Foro y correo webmail.
   - *Síncronas*
     -Chat, pizarra electrónica.

3) Gestión y Distribución de Contenidos: Herramientas de gestión y activación de estos materiales para que los tutores u organizadores incorporen los contenidos. Por otro lado, debe contar con un entorno claro y amigable para que el alumno descargue y visualice el material didáctico.

4) Herramientas de trabajo en Grupo: Esta área cuenta con herramientas de comunicación (chat, foro, correo) y con un repositorio de contenidos propios diferenciados para cada grupo de trabajo.

5) Gestión de Usuarios: Como requisito general, cada usuario debe disponer de medidas de seguridad que limiten la entrada a usuarios no autorizados. Un nombre de usuario y contraseña por ejemplo.

6) Gestión de Cursos: Para poder gestionar la información relativa a cada curso.

7) Herramientas de Seguimiento y Participación: Además de registrar debe facilitar al tutor o coordinador la realización de informes de esta participación y asistencia.

8) Sistema de Evaluación: Debe ser un sistema que gestione tanto los resultados obtenidos en: ejercicios, test de evaluación, trabajos extra, participación en foros, chats, etc.

9) Auditoría del Sistema: Los informes de auditorías tienen como principal finalidad proporcionarnos toda la información requerida para poder mejorar: Entrada al sistema, errores de acceso, ficheros subidos, utilización de recursos, posibles caídas del servidor, usuarios totales, cursos impartidos, etc.

10) Personalización: Debe posibilitar la personalización tanto funcional como estética.

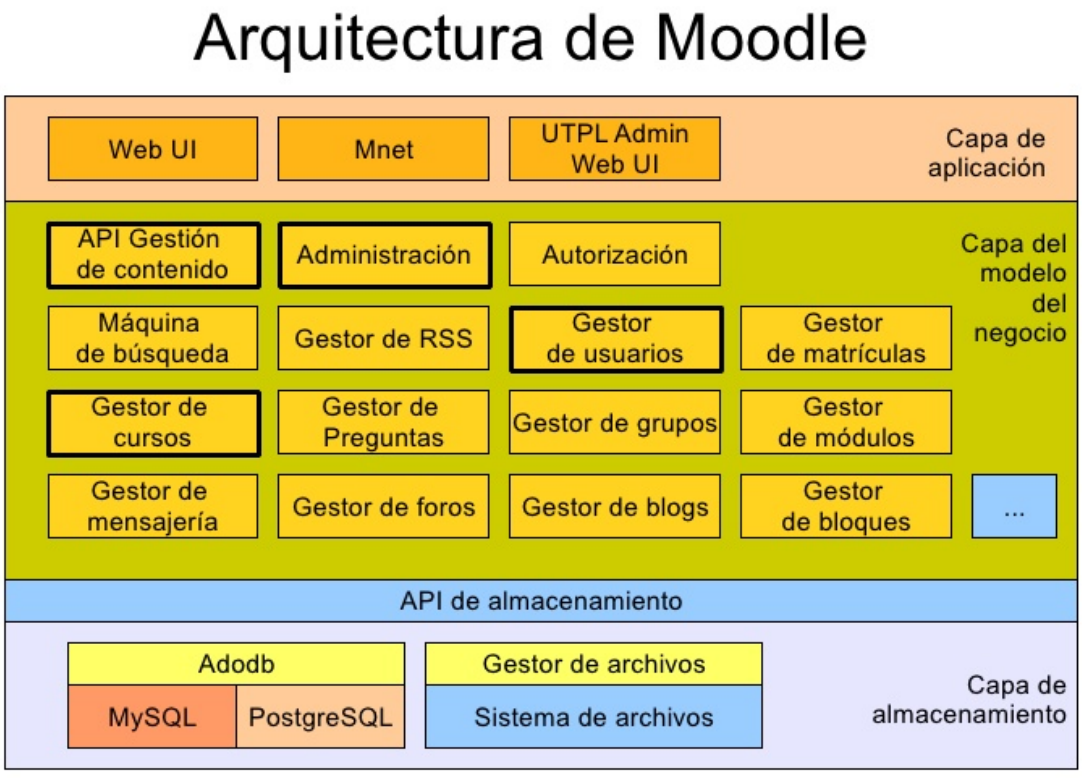


Ilustración 2. Arquitectura Moodle

2.2 Ejemplo de diseño de un sistema software de e-learning para desarrollo de código.

En este apartado se describe una arquitectura de un sistema e-learning a modo de ejemplo para así adentrarnos en la materia.

Desde el punto de vista software en el diseño de una plataforma e-learning se plantean distintas funcionalidades para la plataforma que hay que tener en cuenta a la hora del diseño.

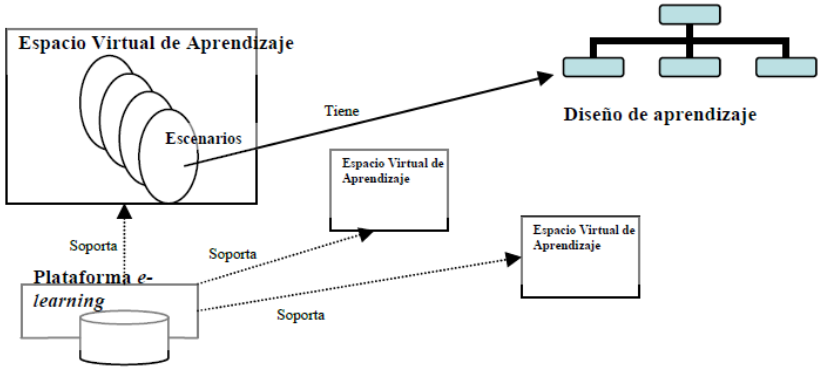


Ilustración 3. Modelo conceptual de un espacio virtual de enseñanza

En estas plataformas los usuarios son capaces de crear, cambiar, borrar y compartir información. Estas acciones están soportadas por herramientas implementadas en el sistema formadas por varias piezas complejas de software. Precisamente por esto es importantísimo desarrollar una buena arquitectura que las permita trabajar conjuntamente de manera eficaz.

A continuación describiré una arquitectura software que da soporte a una plataforma e-learning, razonando las decisiones tomadas y el esfuerzo estimado de desarrollo de la misma. El objetivo de esta plataforma es usarla para asignaturas de programación en la que los alumnos puedan colaborar entre sí para desarrollar diferentes proyectos de manera colaborativa. Dividiendo los usuarios en diferentes "roles": programadores, revisores de código, diseñadores, administradores e investigadores.

2.2.1 Toma de requisitos

[7]Para comenzar realizaremos un estudio de los requisitos funcionales basándonos en las necesidades de los usuarios "actores" que intervienen en la plataforma. Cada actor

representa uno o varios usuarios o grupos de usuarios que tienen un cierto interés o cometido "rol" con el sistema.

2.2.1.1 Requisitos de los Usuarios "stakeholders" y del sistema.

Este entorno e-learning estará centrado en un sistema de desarrollo colaborativo de proyectos de programación y constara de diferentes herramientas para ayudar a: los programadores, revisores de código, diseñadores, administradores e investigadores.

En la Tabla 7 se hace un estudio los requerimientos de uso de cada usuario atendiendo a diferentes criterios de calidad: funcionalidad, rendimiento, usabilidad, eficiencia, mantenibilidad y portabilidad.

A continuación los describo en la siguiente tabla:

CRITERIO	USO/INTERES	USUARIO
FUNCIONALIDAD	Documentación de código y otros artefactos necesarios para un programa como varios pedazos de código organizados.	PROGRAMADORES REVISORES DE CÓDIGO DISEÑADORES ADMINISTRADORES INVESTIGADORES
	Analizar las propiedades de los artefactos	PROGRAMADORES REVISORES DE CÓDIGO
	Permitir la inspección de los artefactos del programa.	REVISORES DE CÓDIGO ADMINISTRADORES INVESTIGADORES
	Los artefactos tienen que ser fácilmente accesibles	PROGRAMADORES
CONFIABILIADAD	Entorno distribuido que permita participar en los proyectos independientemente de la ubicación geográfica.	PROGRAMADORES REVISORES DE CÓDIGO DISEÑADORES ADMINISTRADORES INVESTIGADORES
	El sistema debe de tener tolerancia a fallos.	PROGRAMADORES REVISORES DE CÓDIGO DISEÑADORES ADMINISTRADORES INVESTIGADORES
USABILIDAD	Función para exportar archivos	PROGRAMADORES
MANTENIBILIDAD	Se permite a los usuarios introducir reseñas en relación a los problemas que encuentren.	PROGRAMADORES REVISORES DE CÓDIGO DISEÑADORES ADMINISTRADORES INVESTIGADORES

	Los artefactos deben de documentarse	PROGRAMADORES REVISORES DE CÓDIGO ADMINISTRADORES INVESTIGADORES
	Modificación de cualquier artefacto del proyecto, aprovechando su tipo	PROGRAMADORES
	Dar nuevas funcionalidades al programa	ADMINISTRADORES
PORTABILIDAD	Soportar la resolución de problemas relacionados con el proyecto.	PROGRAMADORES REVISORES DE CÓDIGO DISEÑADORES ADMINISTRADORES INVESTIGADORES

Tabla 1. Usuarios – intereses

En la siguiente tabla tomamos los resultados obtenidos anteriormente y los transformamos en una aproximación a los **requisitos del sistema**.

NÚMERO	DESCRIPCIÓN
1	Entorno distribuido que permita participar en los proyectos independientemente de la ubicación geográfica.
2	Documentación de código y otros artefactos necesarios para un programa como varios pedazos de código organizados.
3	Permitir la inspección de los artefactos del programa.
4	Modificación de cualquier artefacto del proyecto, aprovechando su tipo
5	Analizar las propiedades de los artefactos
6	Los artefactos deben de documentarse Y fácilmente buscables
7	Se permite a los usuarios introducir reseñas en relación a los problemas que encuentren.
8	Implementar nuevas funcionalidades
9	Soportar la resolución de problemas con el proyecto
10	Exportar ficheros

Tabla 2. Tabla de requisitos del sistema.

2.2.2 Requisitos Funcionales.

Describen las funcionalidades que el sistema debe poseer. Estos requisitos funcionales se asociaran a ciertas herramientas que deben de ser soportadas por el sistema final.

Requisitos funcionales del sistema:

1. Los usuarios deben de tener acceso a las diferentes herramientas relacionadas con su rol.
2. Los pedazos de código deben de estar debidamente indexados.
3. Los archivos deben de ser indexables por su contenido o por categorías.
4. Dentro del editor de código debe de estar integrado un buscador de código.
5. Deben de estar explicitas las propiedades básicas ligadas a los artefactos: número de líneas, referencias de identificadores... etc.
6. Todos los artefactos deben de estar explicados de manera que su reutilización sea simple.
7. Debe existir un seguimiento de la resolución de los problemas.
8. Comentar problemas encontrados y proponer nuevas funcionalidades.
9. Soportar mecanismos para compartir ideas: chat, pizarras compartidas, etc.
10. Distribuir el código fuente de manera que sea fácilmente reutilizable con su documentación asociada.

2.2.3 Requisitos No funcionales.

No se incluyen en las características del sistema, pero se toman en cuenta para el comportamiento que este debe de tener.

- **Mantenibilidad** – Manejo: el sistema debe de garantizar consistencia entre las diferentes copias de los artefactos que se guardan en el.
- **Portabilidad** – conformidad: el sistema debe de ser independiente de la plataforma.
- **Eficiencia** – Tiempos de respuesta: tener unos tiempos de procesamiento y acceso que permitan trabajar de manera cómoda.
  – comportamiento de recursos: el sistema debe de responder de manera eficiente a las actividades típicas de edición.

2.2.4 Restricciones

Se trata de los requerimientos que deben de existir en el sistema final para satisfacer las funcionalidades requeridas por los usuarios.

- **Disponibilidad**: Debe de existir un repositorio SVN (repositorio de subversión).
- **Confiabilidad:** el sistema debe garantizar la consistencia entre todas las parte y en caso de verse comprometida comunicarlo.
- **Tolerancia a fallos**: debe de existir una redundancia de datos.

2.3 Arquitectura y vistas.

2.3.1 Marco de referencia para la descripción de la arquitectura.

En este caso la arquitectura se estructurara conforme a ANSI/IEEE 1471-2000, para sistemas software-intensivos.

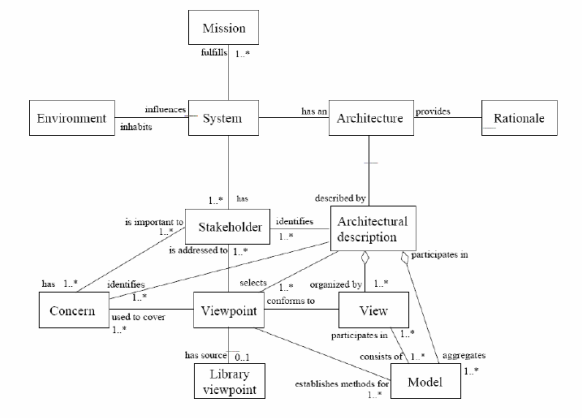


Ilustración 4. IEEE 1471 descripción

Para crear la descripción de la arquitectura se usara el modelo Krunchen 4+1 y la notación UML 2.

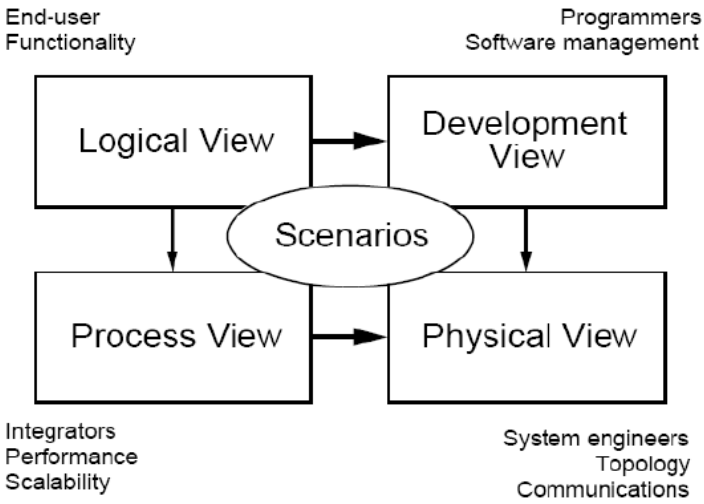


Ilustración 5. Modelo Krunchen 4+1

2.3.2 Vista lógica

Esta soporta los requerimientos funcionales, se divide el sistema en una serie de abstracciones clave. Estas abstracciones son los objetos o clases.

La lista de herramientas que contendrá la aplicación, es la siguiente:

- Herramientas de codificación:
    1. Buscador de código.
    2. Editor de código.
    3. Analizador de código.
    4. Documentador de código.
- Herramienta de seguimiento de errores.
- Herramienta de chat/discusión.
- Herramienta para añadir funcionalidades.
- Herramienta de exportación.

El razonamiento aplicado es dividir las funcionalidades en diferentes paquetes "clases", de esta manera se realiza una separación lógica más efectiva. En la ilustración 32 observamos el diagrama de clases con sus: atributos, asociaciones y métodos. En él se satisfacen todos los requerimientos del sistema.

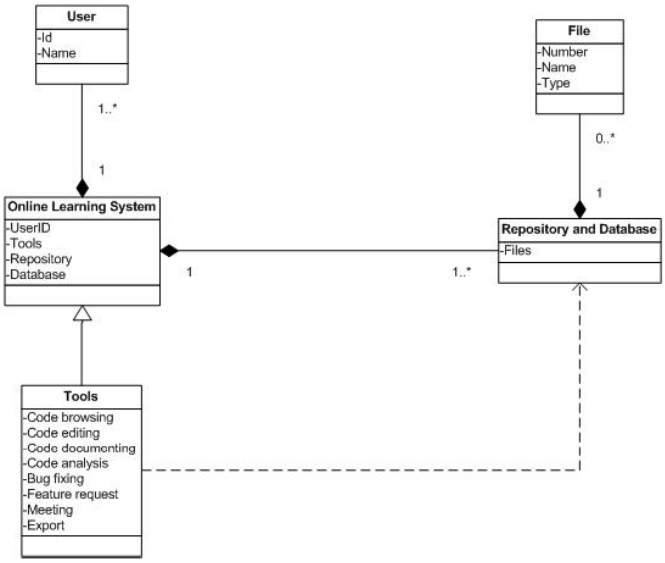


Ilustración 6. Diagrama de clases propuesto

Breve descripción de las clases:

- Online learning system: la clase principal del programa que se asocia con el resto.
- User: clase necesaria para la identificación de los usuarios, asi como sus roles y privilegios.
- File: la clase responsable de indexar los pedazos de código de cada proyecto.
- Tools: en esta clase se recogen todas las herramientas para el desarrollo del trabajo.
- Repository and data base: es responsable del almacenamiento y distribución de los ficheros.

2.3.3 Vista de proceso

Este punto de vista captura los aspectos de concurrencia, sincronización y distribución del sistema. Explica de que manera los objetos interactúan para poder lograr sus especificaciones.

En la siguiente ilustración se muestran la secuencia de procesos que tienen lugar cuando un usuario usa el buscador de código y el editor.

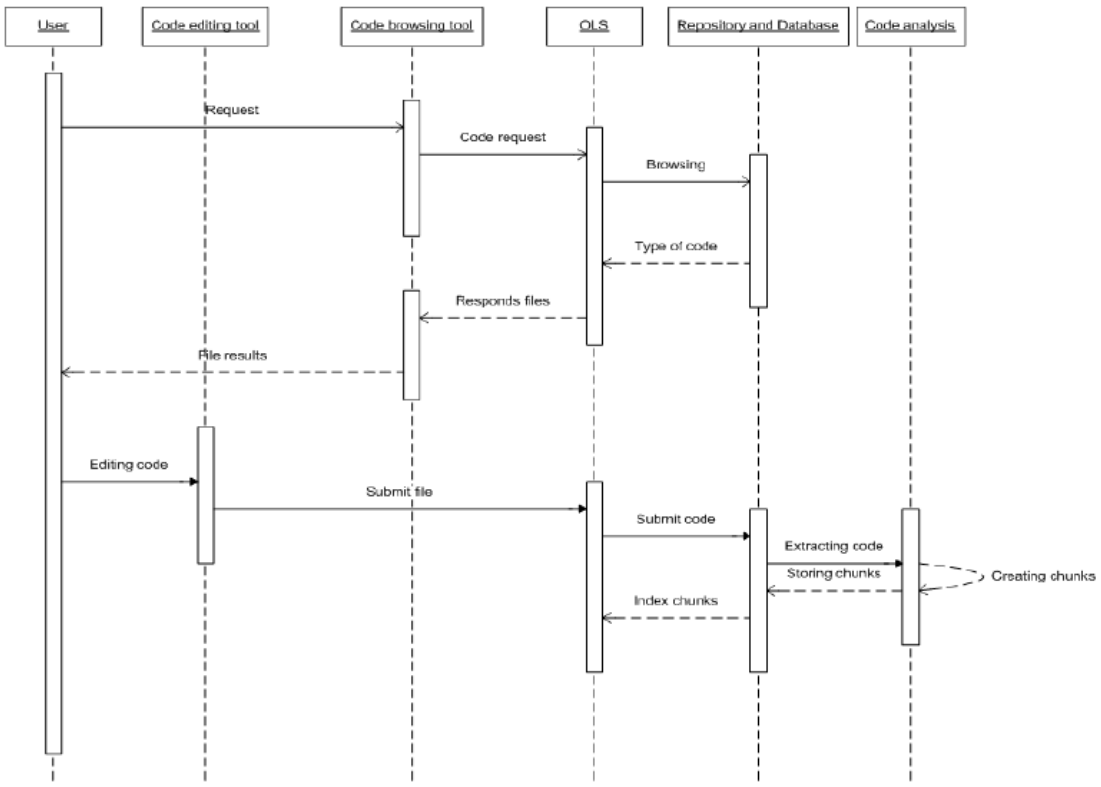


Ilustración 7. Diagrama de secuencia para buscar código y editar.

Para definir las responsabilidades de cada objeto usamos la técnica que se basa en el modelo-vista-controlador (MVC) que define cada objeto como: *frontera, control o entidad*. Siendo objetos frontera los que interactúan directamente con el actor mostrando información o recibiendo peticiones. Los de control los que reciben mensajes de los objetos frontera y deciden las acciones que tomar. Y los objetos de entidad son los encargados de realizar los procesos.

En esta arquitectura el interface se crearía por el módulo de presentación web el cual sería realmente el objeto frontera, mediante el cual se interactúa con el actor. A fin de reflejar con más claridad la arquitectura en los diagramas de secuencia hare que este paso sea "Transparente". Con esto quiero decir, que para el desarrollo de estos diagramas entenderé que el actor interactúa directamente con los objetos para así evitar la redundancia de pasar por el interface.

Atendiendo a esta clasificación las responsabilidades de los objetos del diagrama son:

- Frontera: módulo de presentación Web.
- Control: “Code editing tool” y “Code Browsing tool” ya que reciben los mensajes del módulo frontera y los gestionan.
- Objetos entidad: “OLS” gestiona las operaciones con la base de datos.

Como ejemplo, los pasos de la búsqueda de código son:

1. El usuario contacta con la herramienta de búsqueda con una petición para buscar “algo”, en el código fuente.
2. La herramienta de búsqueda hace una consulta al sistema con los parámetros y argumentos que correspondan.
3. El sistema busca en el repositorio con los criterios aportados.
4. La base de datos y repositorio devuelven la información pedida.
5. El sistema muestra la información mediante la herramienta de búsqueda.
6. La herramienta muestra la lista en el formato adecuado para el usuario.

A continuación se describe la secuencia de procesos para solicitar una reunión, información o aclaraciones. Secuencia que caracteriza esta arquitectura e-learning.

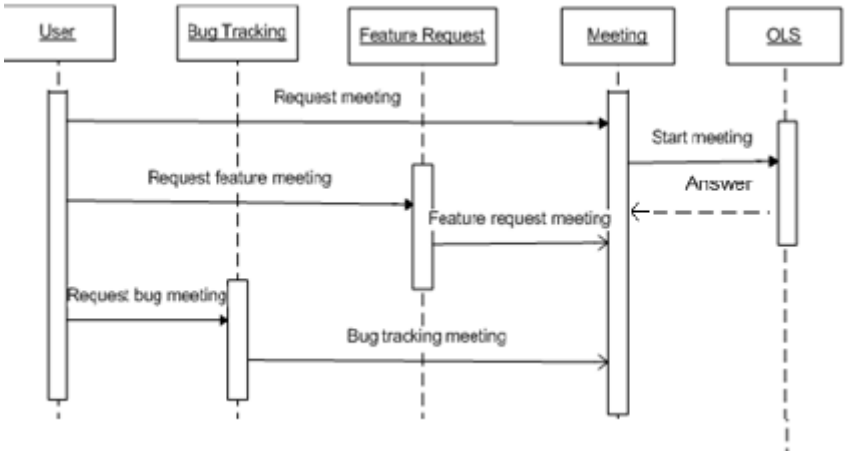


Ilustración 8. Diagrama de secuencia para petición de reunión

En este caso, ya bien sea para: solicitar información, solicitar nuevas funcionalidades o para una reunión para buscar errores. Ciñéndonos a la anterior clasificación las responsabilidades aquí se reparten de la siguiente manera:

- Frontera: módulo de presentación Web.

- Control: “Bug Tacking”, “feature request” y “Meeting”. Ya que reciben peticiones del módulo de presentación Web.

-Objetos entidad: “OLS”, que es el encargado de aceptar o denegar las peticiones del usuario y ”Meeting” que está encargado de realizar las comunicaciones.

Para realizar una petición se sigue la siguiente secuencia:

1. El actor mediante su aplicación cliente en el navegador web puede solicitar empezar una sesión de encuentro de tres formas:
   a. Solicitando una reunión simplemente al módulo de reuniones, por ejemplo con objeto de tratar algún tema con un compañero/s o profesor/es.
   b. Reunión mediante la solicitud de nuevas funcionalidades con su equipo.
   c. Reunión mediante el seguidor de fallos para contactar con los miembros de desarrollo y así discutir como corregirlos.
2. Todas estas peticiones pasan por “OLS” encargado de: controlarlas, aceptarlas o denegarlas.

2.3.4 Vista Física

Toma los atributos no funcionales como: tolerancia a fallos, disponibilidad, escalabilidad y funcionamiento. El sistema se estructurara en un patrón de tres capas como se muestra en la siguiente ilustración.

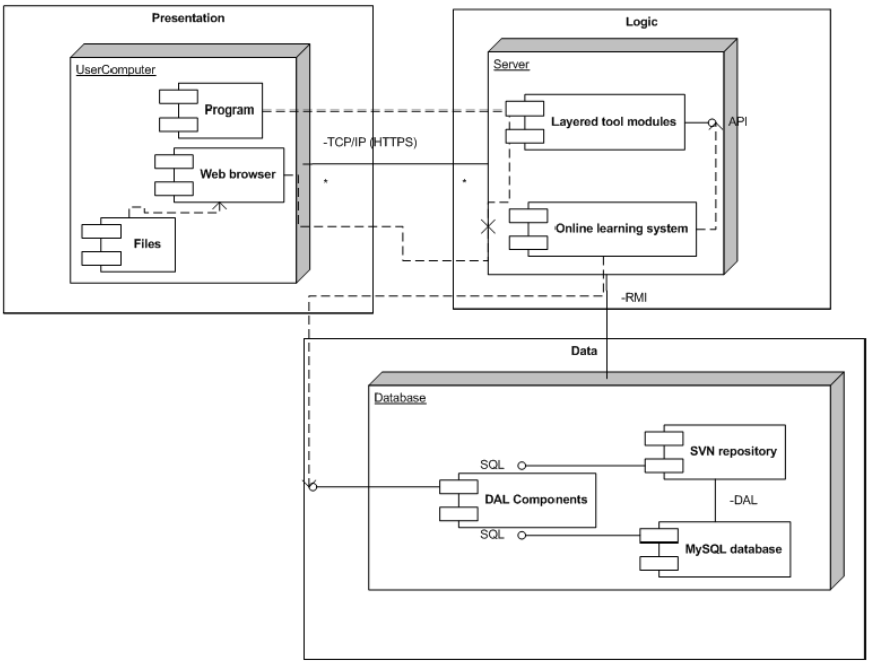


Ilustración 9. División en capas de sistema e-learning

- **Capa de presentación**: En esta capa esta el lado que el usuario ve e interactúa con el sistema.
- **Capa lógica**: es la capa donde se realizan las operaciones en base a las peticiones del cliente.
- **Capa datos**: es la encargada de mediar con los sistemas de almacenamiento.

2.3.5 Vista desarrollo

Se centra en la organización del desarrollo del programa, y lo visualiza como un conjunto de módulos que interactúan entre ellos pero que pueden ser desarrollados de manera individual, a fin de mejorar el trabajo en equipo para el desarrollo del software. Se representa mediante un diagrama de componentes.

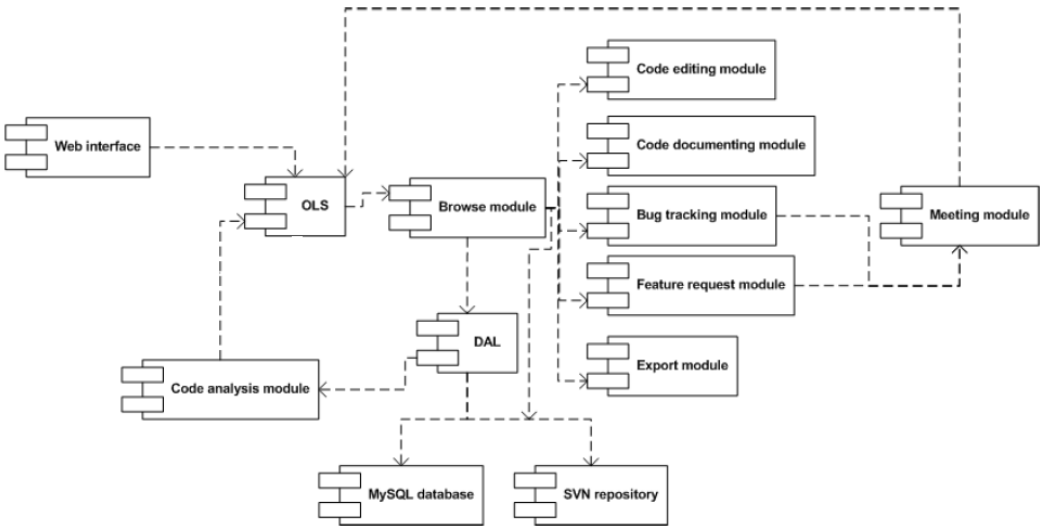


Ilustración 10. Diagrama de componentes

Descripción de los módulos:

- *Web interface*: nos proporciona la vista en nuestro navegador web.
- *OLS*: es el que controla y administra la interacción entre las funcionalidades.
- *Code analysis module:* se encarga de analizar los códigos.
- *Browse module*: Es el buscador de códigos.
- *DAL:* (Database Abstraction Layer) gestiona las bases de datos.
- *Code editing module:* proporciona todas las funcionalidades de edición.
- *Code documenting module:* nos permite documentar el código.
- *Bug tracking module:* nos previene sobre fallos.
- *Feature request module:* sirve para solicitar nuevas funcionalidades.
- *Export module:* permite exportar en diferentes formatos.
- *Meeting module:* es el que le da el carácter de e-learning a la aplicación ya que es el que permite interactuar entre los usuarios. Trabaja en combinación las peticiones de nuevas funcionalidades y la detención de errores.

2.3.6 Vista escenarios

Conecta todas las visiones anteriores en esta visión se adjuntan los casos de uso que explican la manera como el usuario y el sistema interactúan. Como se trata de un actor no humano tendrá que ser manejado mediante una interfaz o un adaptador.

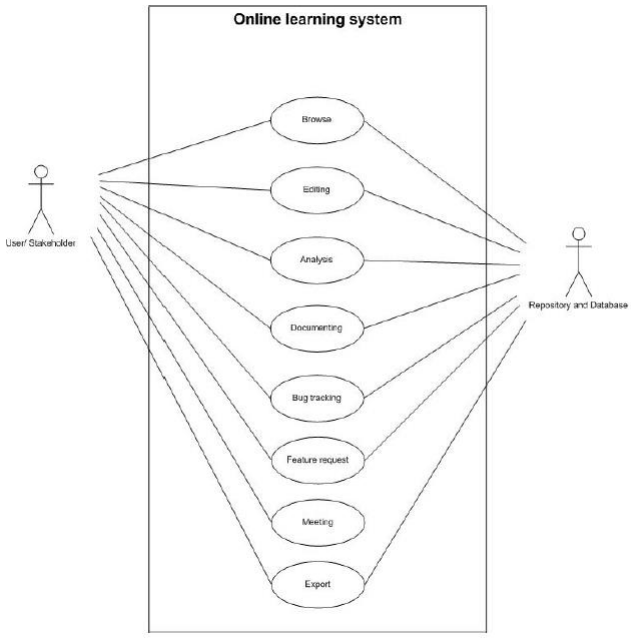


Ilustración 11. Diagrama de caso de uso

2.3.6.1 Casos de uso

Describiremos a modo de ejemplo el caso de uso de "edición". Se entiende que se modifica la información existente en la base de datos.

CASO DE USO	EDITAR
ACTOR	Programadores, revisores de código, investigadores, administradores y diseñadores.
DESCRIPCIÓN	Modificar cualquier artefacto del proyecto
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Se editan los ficheros deseados quedando guardados en la BD
SECUENCIA DE EJUCIÓN	1- Entrada en el sistema. 2- El sistema conecta con la base de datos. 3- Se editan los ficheros deseados.

Tabla 3. Caso de uso de edición

3. Caso de estudio particular y análisis con Arche.

Una vez estudiado el ejemplo de sistema de desarrollo de código e-learning a través de sus etapas. En este punto plantearemos una arquitectura particular, la cual analizaremos con la herramienta ArchE para encontrar las **deficiencias** que nos plantea nuestro diseño a la hora de satisfacer unos escenarios de calidad propuestos. Seguidamente con la ayuda de las sugerencias "Tácticas" que la herramienta nos aporte realizaremos las **mejoras** de dicha arquitectura.

3.1 Definición del sistema

Construiremos una arquitectura software que de soporte a un sistema web e-learning para impartir cursos online. Cuyos requisitos exploraremos a través de los casos de uso. Una vez construida dicha arquitectura la analizaremos con Arche en base a unos escenarios de calidad propuestos. Realizaremos los cambios pertinentes usando las tácticas propuestas por el sistema experto.

3.1.1 Identificación de actores principales y objetivos:

[7]En el cual existirán cuatro tipos de actores cada cual con unos objetivos concretos:

- **Administrador:** Podrá generar informes de uso del sistema, acreditar profesores y realizar consultas.

- **Profesor:** Registrar alumnos y gestionarán las aulas virtuales. Tendrá capacidad para modificar datos.

- **Estudiante:** Participaran en los cursos. Pueden consultar su expediente.

- **Visitante:** Puede ver la información sobre los cursos y preinscribirse en estos.

Diagrama de casos de uso:

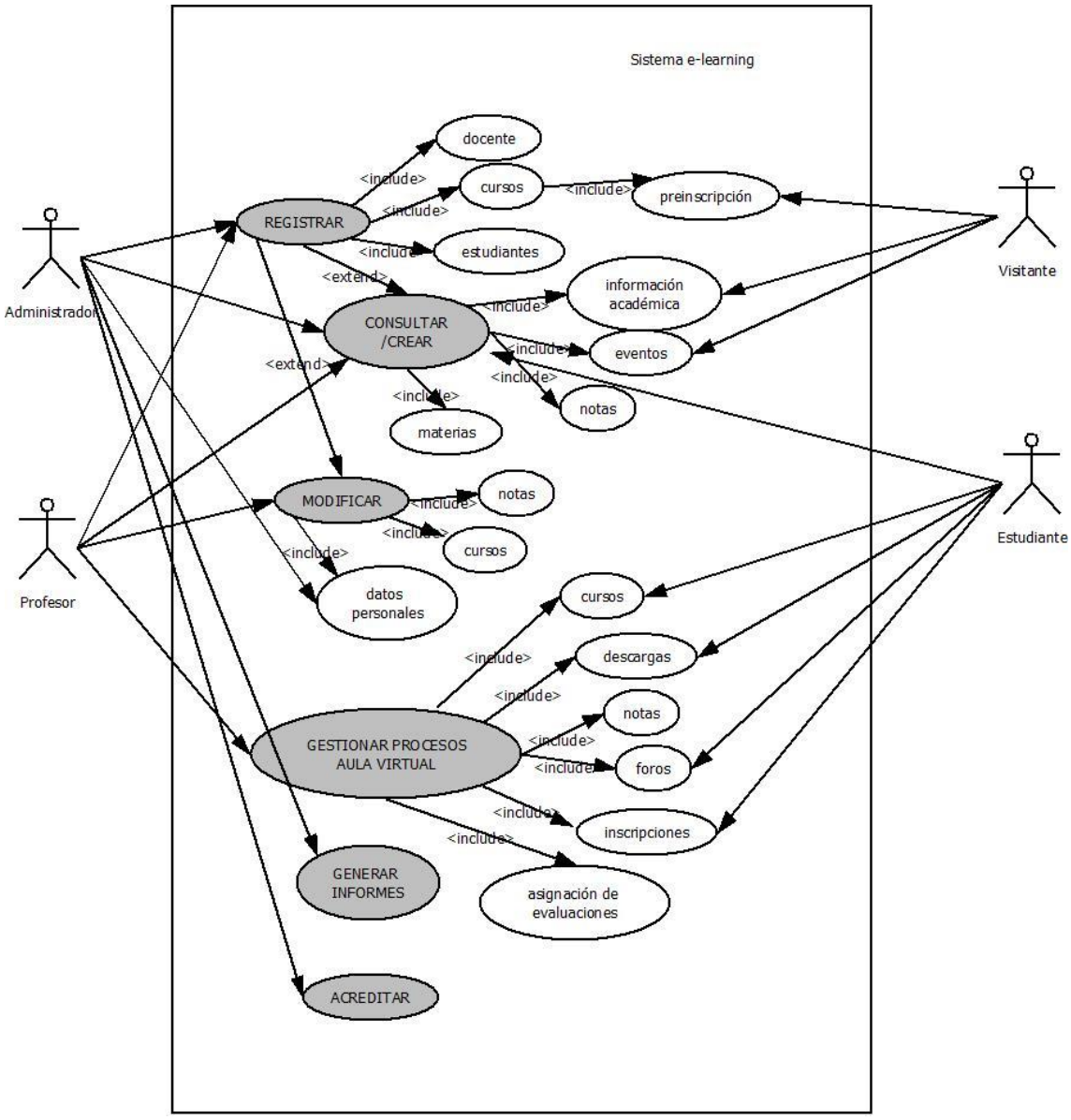


Ilustración 12. Diagrama casos de uso e-learning

Describimos los casos de uso que se derivan del anterior diagrama y en los cuales quedan implícitos los requisitos funcionales:

CASO DE USO - 1	Registrar Docente
ACTOR	Administrador
DESCRIPCIÓN	Dar de alta nuevos profesores
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Nuevos profesores registrados
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Rellenar datos y privilegios. 3- Dar de alta al docente.

Tabla 4. Caso de uso 1 Registrar Docente

CASO DE USO -2	Registrar Cursos
ACTOR	Administrador
DESCRIPCIÓN	Registra un nuevo curso en la plataforma
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Nuevo curso añadido
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar características del nuevo curso 3- Dar de alta el curso y asignar Docente/s

Tabla 5. Caso de uso 2 Registrar cursos

CASO DE USO - 3	Registrar Estudiantes
ACTOR	Administrador
DESCRIPCIÓN	Registrar estudiantes y asignarles los cursos en los que estén matriculados.
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Nuevo estudiante Registrado
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Validar datos estudiante. 3- Verificar cursos a los que está inscrito

Tabla 6. Caso de uso 3 modificar

CASO DE USO - 4	Consulta información académica
ACTOR	Administrador
DESCRIPCIÓN	Consulta de los datos académicos de los estudiantes
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Consulta/ exportación de datos
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Alumno 3- Consultar o exportar datos académicos

Tabla 7. Caso de uso 4 Consulta información académica

CASO DE USO - 5	**Consultar/ crear eventos**
ACTOR	Administrador
DESCRIPCIÓN	Se pueden consultar los eventos creador por otros usuarios o crear uno nuevo.
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Evento creado o consultado.
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar para su lectura un evento o crear uno nuevo

Tabla 8. Caso de uso 5 Consultar/ crear eventos

CASO DE USO - 6	**Consultar notas**
ACTOR	Administrador
DESCRIPCIÓN	Seleccionando al alumno consultar su expediente
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Datos de expediente consultados
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar consulta de expediente 3- Buscar alumno

Tabla 9. Caso de uso 6 consultar notas

CASO DE USO - 7	**Consultar materias**
ACTOR	Administrador
DESCRIPCIÓN	Consultar las materias disponibles
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Materias disponibles
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar consulta de materias 3- Obtener listado de materias y atributos

Tabla 10. Caso de uso 7 consultar materias

CASO DE USO – 8	**Generar informes**
ACTOR	Administrador
DESCRIPCIÓN	Generar informes de uso de la aplicación
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Informes en formato exportable
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar realizar informes. 3- Elegir el tipo de informe y cualidades 4- Generar informe en el formato deseado

Tabla 11. Caso de uso 8 Generar informes

CASO DE USO – 9	**Acreditar alumnos y profesores**
ACTOR	Administrador
DESCRIPCIÓN	Habilitar la acreditación para los alumnos y los profesores. A fin de que cada uno acceda a los cursos que realizan o se encargan.
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Profesor o alumno acreditado
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar acreditar 3- Buscar usuario 4- Modificar sus acreditaciones.

Tabla 12. Caso de uso 9 acreditar alumnos y profesores

CASO DE USO – 10	**Preinscripción en cursos**
ACTOR	Visitante
DESCRIPCIÓN	Realizar una pre matricula en un curso
CONDICIONES INICIALES	Entrada en la plataforma
CONDICIONES FINALES	Plaza reservada en un curso
SECUENCIA DE EJECUCIÓN	1- Seleccionar curso 2- Rellenar formulario pre-matrícula

Tabla 13. Caso de uso 10 preinscripción en cursos

CASO DE USO – 11	**Información académica**
ACTOR	Visitante
DESCRIPCIÓN	Acceder a la información de los cursos disponibles en la plataforma.
CONDICIONES INICIALES	Entrada en la plataforma
CONDICIONES FINALES	Descarga documentación académica
SECUENCIA DE EJECUCIÓN	1- Seleccionar curso. 2- Consulta o descarga de la información académica de ese curso

Tabla 14. Caso de uso 11 Información académica

CASO DE USO – 12	**Visualización de eventos**
ACTOR	Visitante
DESCRIPCIÓN	Visualización de eventos
CONDICIONES INICIALES	Entrada en la plataforma
CONDICIONES FINALES	Visualización de eventos activos
SECUENCIA DE EJECUCIÓN	1- Seleccionar curso 2- Visualización de eventos.

Tabla 15. Caso de uso 12 visualización de eventos

CASO DE USO – 13	**Consulta información académica**
ACTOR	Estudiante
DESCRIPCIÓN	Consulta de la información académica de un curso
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Información académica consultada
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Consulta información académica

Tabla 16. Caso de uso 13 Consulta información académica

CASO DE USO – 14	**Consultar eventos**
ACTOR	Estudiante
DESCRIPCIÓN	Consulta de los eventos generales o relativos a un curso
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Información del evento visualizada
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Visualizar evento

Tabla 17. Caso de uso 13 Consulta información académica

CASO DE USO – 15	**Consultar materias**
ACTOR	Estudiante
DESCRIPCIÓN	Consulta de las materias en las que está matriculado
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Visualización de los datos
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar consulta 3- Seleccionar materia y curso

Tabla 18. Caso de uso 15 consultar materias

CASO DE USO – 16	**Visualizar cursos en los que está matriculado**
ACTOR	Estudiante
DESCRIPCIÓN	El alumno se le listarán los cursos en los que está matriculado
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Listado de cursos en los que está matriculado el alumno
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Ver los cursos en los que está matriculado en su página de inicio

Tabla 19. Caso de uso 16 Visualizar cursos en los que está matriculado

CASO DE USO – 17	**Descarga material del curso**
ACTOR	Estudiante
DESCRIPCIÓN	El alumno podrá descargar ficheros asociados a los cursos en los que este matriculado
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Descarga de recursos
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Selección de un curso 3- Descarga de archivos asociados

Tabla 20. Caso de uso 17 descarga material del curso

CASO DE USO - 18	**Participación en Foros**
ACTOR	Estudiante
DESCRIPCIÓN	El alumno puede leer y participar en los foros de sus cursos
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Lectura o participación en los foros
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar Curso 3- Participación o lectura del foro

Tabla 21. Caso de uso 18 Participación en Foros

CASO DE USO – 19	**Realizar inscripciones**
ACTOR	Estudiante
DESCRIPCIÓN	El alumnos puede inscribirse en cursos
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Inscripción en curso
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Ver listado de cursos disponibles 3- Inscribirse en estos

Tabla 22. Caso de uso 19 Realizar inscripciones

CASO DE USO – 20	**Registrar estudiantes**
ACTOR	Profesor
DESCRIPCIÓN	El profesor pude inscribir a los matriculados en sus cursos
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Alumnos registrados en cursos
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Registro de participantes

Tabla 23. Caso de uso 20 Registrar estudiantes

CASO DE USO – 21	**Registrar cursos**
ACTOR	Profesor
DESCRIPCIÓN	Crear nuevos cursos
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Nuevo curso disponible
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Crear nuevo curso 3- Determinar sus atributos

Tabla 24. Caso de uso 21 Registrar curos

CASO DE USO – 22	**Crear/consultar información académica**
ACTOR	Profesor
DESCRIPCIÓN	Se puede crear o consultar la información académica relativa a un curso
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Información académica consultada o creada
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Consultar/crear información académica

Tabla 25. Caso de uso 22 Crear/consultar información académica

CASO DE USO - 23	**Crear eventos**
ACTOR	Profesor
DESCRIPCIÓN	Crear eventos relativos a los cursos
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Crear evento en un curso
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Crear evento

Tabla 26. Caso de uso 23 Crear eventos

CASO DE USO – 24	**Crear/consultar notas**
ACTOR	Profesor
DESCRIPCIÓN	Consultar notas de alumnos
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Nota consultada o creada
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Seleccionar alumno 4- Crear o consultar notas

Tabla 27. Caso de uso 24 Crear/consultar notas

CASO DE USO – 25	**Crear/consultar materia**
ACTOR	Profesor
DESCRIPCIÓN	Modificar la materia asociada al curso
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Materia modificada
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Consultar/crear materia

Tabla 28. Caso de uso 25 Crear/consultar materia

CASO DE USO – 26	**Modificar Cursos**
ACTOR	Profesor
DESCRIPCIÓN	Se puede modificar los atributos de los cursos
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Atributos de los cursos modificados
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Modificar los atributos de los cursos

Tabla 29. Caso de uso 26 Modificar cursos

CASO DE USO – 27	**Modificar datos personales**
ACTOR	Profesor
DESCRIPCIÓN	Modificar los datos personales relativos a los alumnos
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Datos personales modificados
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Seleccionar alumno 4- Modificar sus datos personales

Tabla 30. Caso de uso 27 modificar datos personales

CASO DE USO – 28	**Gestionar cursos**
ACTOR	Profesor
DESCRIPCIÓN	Cambiar parámetros del curso
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Parámetros del curso modificados
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Seleccionar gestión del curso 4- Modificar parámetros

Tabla 31. Caso de uso 28 Gestionar cursos

CASO DE USO – 29	**Gestionar descargas**
ACTOR	Profesor
DESCRIPCIÓN	Seleccionar el material de descarga, hacerlo visible u ocultarlo
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Descargas gestionadas
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Gestionar las descargas

Tabla 32. Caso de uso 29 Gestionar descargas

CASO DE USO – 30	**Participar en foros**
ACTOR	Profesor
DESCRIPCIÓN	Participar en foros.
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Participación en foros
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Participar en los foros

Tabla 33. Caso de uso 30 Participar en foros

CASO DE USO – 31	**Asignar evaluación**
ACTOR	Profesor
DESCRIPCIÓN	El profesor elige cuando evaluar a los alumnos y realizar exámenes online
CONDICIONES INICIALES	Login correcto en la plataforma
CONDICIONES FINALES	Asignación de curso y alumnos para evaluación
SECUENCIA DE EJECUCIÓN	1- Entrada en el sistema. 2- Seleccionar curso 3- Seleccionar alumnos 4- Crear evaluación y atributos.

Tabla 34. Caso de uso 31 Crear/consultar información académica

Relación entre actores y casos de uso:

ACTOR	FUNCIÓN PRINCIPAL/ OBJETIVO	CASOS DE USO
PROFESOR	Registrar cursos y estudiantes, Gestionar los cursos. Modificar datos alumnos.	Registrar estudiantes Registrar cursos Crear/consultar inflo. Académica Crear/consultar eventos Crear/consultar notas Crear/consultar materias Modificar notas Modificar cursos Modificar datos personales Gestionar cursos Gestionar descargas Gestionar notas Participar en foros Asignar evaluación
ADMINISTRADOR	Acceder a todo el sistema excepto a la gestión de los cursos. Acreditar nuevos administradores o profesores.	Registrar Docente Registrar cursos Registrar estudiantes Consultar información académica Crear eventos Consultar Notas Consultar Materias Generar informes de uso Acreditar alumnos y profesores
VISITANTE	Realizar preinscripciones y visualizar información sobre cursos	Preinscribirse en cursos Información académica Visualización de eventos
ALUMNO	Realizar cursos y acceder a su información personal.	Consultar información académica Consultar eventos Consultar notas Consultar materias Consulta cursos Descargar material del curso Participación en foros Realizar inscripciones

Tabla 35. Actores e-learning

Funcionalidades del sistema:

Se extraen de los casos de uso:

FUNCIONALIDADES	DESCRIPCIÓN
REGISTRAR DOCENTE	Permite añadir un nuevo profesor
REGISTRAR CURSO	Permite añadir nuevos cursos
REGISTRAR ESTUDIANTES	Añadir nuevos estudiantes
CONSULTAR INFORMACIÓN ACADEMICA	Para observar la información sobre los cursos
MODIFICAR DATOS PERSONALES	Realizar cambios en los datos personales
CREAR EVENTOS	Crear eventos generales o en los cursos
CONSULTA DE NOTAS	Consulta de calificaciones
CONSULTAR MATERIAS	Para tener acceso a la información de las materias en las que esta cada curso
GENERAR INFORMES	Realizar informes de uso de la plataforma
ACREDITAR	Para validar nuevos alumnos o profesores
INSCRIPCIÓN YPREINSCRIPCIÓN EN CURSOS	Permite preinscribir a los visitantes e inscribirse a los alumnos
VISUALIZAR EVENTOS	Permite visualizar los eventos que estén creados
PARTICIPAR FOROS	Permite visualizar y participar en foros de debate
GESTIONAR CURSOS	Gestionar cursos y crearlos
GESTIONAR DESCARGAS	Insertar, eliminar y modificar privilegios
GESTIONAR NOTAS	Modificar y crear notas
ASIGNAR EVALUACIÓN	Crear y asignar exámenes a los alumnos

Tabla36. Tabla de funcionalidades

3.2 Arquitectura del sistema e-learning

En este apartado hablaremos de la arquitectura que usaremos para la implementación de la plataforma.

Ya que se trata de una aplicación Web usaremos archiconocido patrón de diseño MVC ilustración 13.

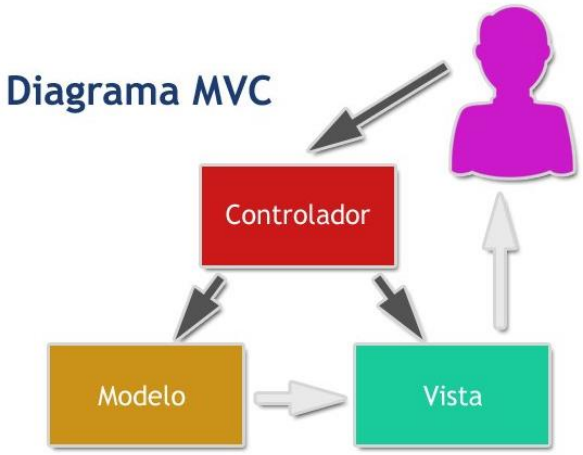


Ilustración 13. Arquitectura por capas

Este patrón nos introduce diversas partes o conceptos en los que debemos separar el código de nuestra arquitectura.

- **Modelo:** Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

- **Controlador:** Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

- **Vista:** Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

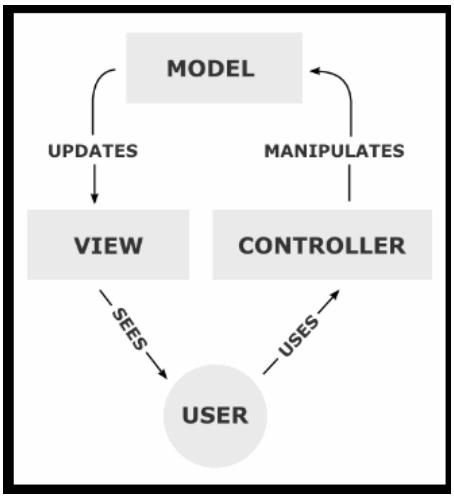


Ilustración 14. Colaboración entre componentes MVC

Agruparemos las funcionalidades en paquetes de la siguiente manera:

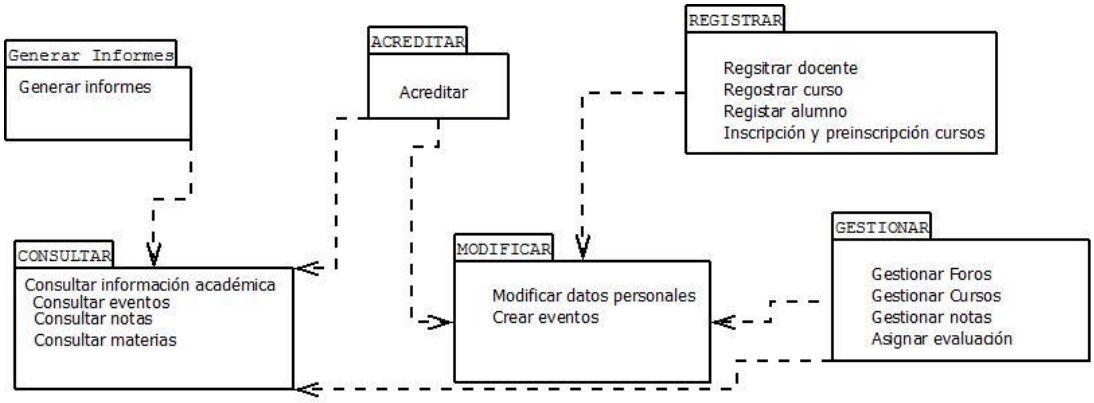


Ilustración 15. Diagrama de paquetes E-LEARNING

Las dependencias que existen entre paquetes son:

- Generar informes depende de consultar para extraer la información.
- Acreditar depende de realizar primero una consulta y luego modificar.
- Registrar depende de modificar, para poder realizar cambios.
- Gestionar necesita realizar operaciones de consulta y modificaciones.

Mostramos el diagrama de despliegue:

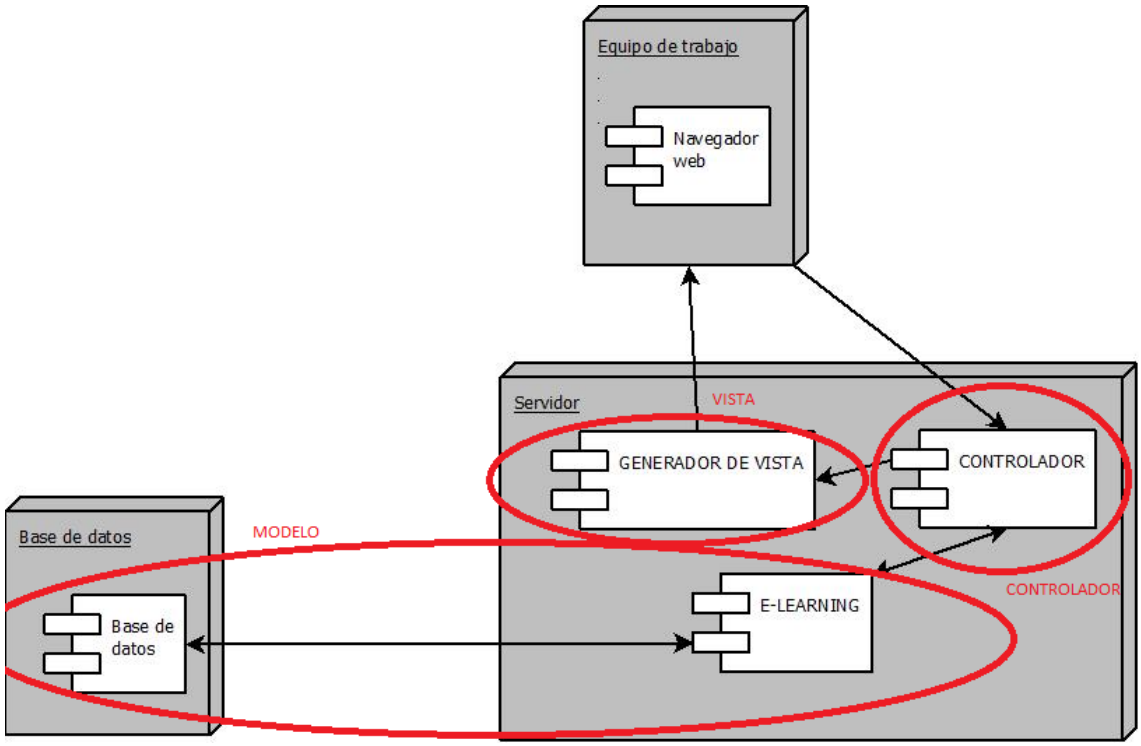


Ilustración 16. Diagrama de despliegue

3.2.1 Escenarios de calidad

Proponemos varios escenarios de rendimiento y modificabilidad. Ya que Arche solo puede evaluar estos dos atributos de calidad.

3.2.1.1 Escenarios de rendimiento

En estos escenarios se definen requisitos de calidad relativos al comportamiento de la aplicación en términos de rendimiento. Al tratarse de una aplicación web dependemos de los siguientes factores:

- El equipo cliente: El tiempo que tarda el equipo del cliente en enviar, recibir y transformar la información.
- La red: tiempo de transito de la información.
- Servidor: Tiempo de ejecución de los diferentes procesos.
- Base de datos: Tiempos de acceso o modificación de la información en la base de datos.

[8] Para medir esta dimensión de la calidad se usan pruebas de rendimiento para descubrir problemas que pueden ser resultado de: falta de recursos en el lado servidor, red con ancho de banda inadecuada, capacidades de base de datos inadecuadas, capacidades de sistema operativo deficientes o débiles, funcionalidad de webapp pobremente diseñada y otros conflictos de hardware o software que pueden conducir a rendimiento cliente-servidor degradado.

En nuestro diseño al tratarse de una propuesta de arquitectura se hace difícil la realización de pruebas de rendimiento reales. Fuertemente dependientes del hardware. Simplemente lo que se quiere evaluar usando Arche es que las hipótesis de tiempos para la ejecución de los diferentes módulos es correcta. Es decir, que si a la hora de implementar nuestra aplicación logramos mantener dichos tiempos nuestro sistema será planificable y cumplirá los requisitos de rendimiento establecidos en los escenarios correspondientes.

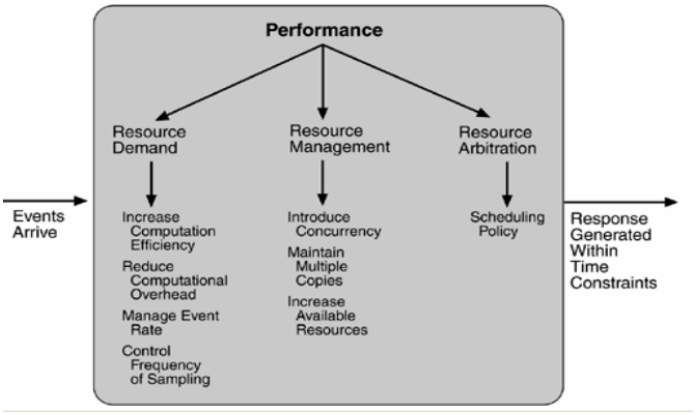


Ilustración 17. Tácticas mejora de rendimiento

- **Escenario 1:** El módulo Acreditar necesita recibir la información procedente de los módulos consultar y modificar. Para saber que se ha acreditado un usuario correctamente. El tiempo máximo ha de ser de 100 ms. El módulo acreditar debe de ser capaz de realizar su función en 90 ms. Los módulos afectados son Acreditar, Consultar y Modificar.

Elemento	Atributo/valor	Tipo	Unidad	Valor
Estímulo	Petición de acreditar	Periodo	ms	100
Fuente del estimulo	Consultar, modificar, controlador	Sistema	-	
Entorno	Condiciones normales	Condiciones normales	-	
Artefacto	Sistema	Sistema	-	
Respuesta	Se acredita usuario	Latencia de tarea	-	
Medida de la respuesta	Antes de 90 ms	Caso peor	ms	90

Tabla 37. Escenario Rendimiento P1

- **Escenario 2:** El módulo Gestionar A.V. necesita recibir la información procedente del módulo consultar. Para poder realizar los test de evaluación. El tiempo máximo de respuesta ha de ser de 100 ms. El módulo Gestionar A.V. debe de ser capaz de realizar su función en 80 ms. Los módulos afectados son Gestionar A.V. y consultar.

Elemento	Atributo/valor	Tipo	Unidad	Valor
Estímulo	Realizar test modulo	Periodo	ms	100

	Gestionar A.V.			
Fuente del estimulo	Consultar	Sistema	-	
Entorno	Condiciones normales	Condiciones normales	-	
Artefacto	Sistema	Sistema	-	
Respuesta	Se carga información de test	Latencia de tarea	-	
Medida de la respuesta	Antes de 80 ms	Caso peor	ms	80

Tabla 38. Escenario Rendimiento P2

3.2.1.2 Escenarios de modificabilidad

Estudian los costes de realizar cambios en el sistema, como afectan al sistema y otros módulos. Valoran la habilidad del sistema para ser flexible frente a cambios.

Las mediciones se realizan realizando cambios en los diferentes módulos y viendo cuanto costo tienen (entendiendo como costo el tiempo que conllevan realizar dichos cambios). Para que estos datos sean positivos deben de entrar dentro de los márgenes marcados por el cliente en los escenarios de calidad propuestos a este respecto.

En la arquitectura que planteamos para medir esto estableceremos unos valores de modificabilidad asociados cada módulo (en días de trabajo). Usaremos Arche para verificar si se cumplen los escenarios propuestos. Y de no ser así, usaremos las tácticas que este nos proponga para conseguir el objetivo marcado en los escenarios de modificabilidad.

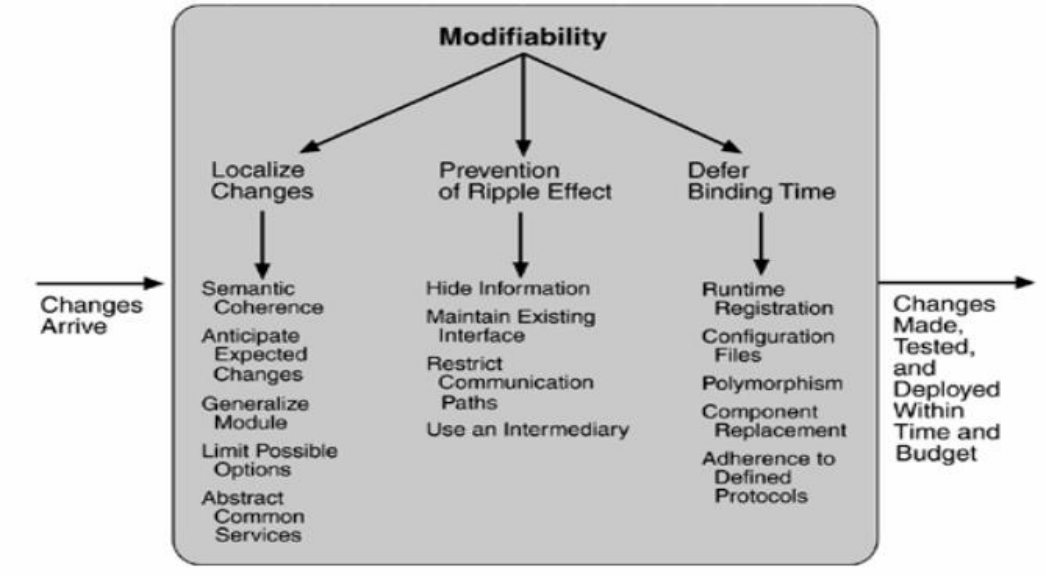


Ilustración 18. Tácticas de modificabilidad

- **Escenario 3:** Modificar el módulo Consultar, para agregar mejoras o nuevas funcionalidades afecta a Generar informes, acreditar, y gestionar A.V/. El coste de estas modificaciones seria de 15 días.

ELEMENTO	ATRIBUTO/VALOR
Estímulo	Modificación de módulo Consultar
Fuente del estimulo	Inge. Softw.
Entorno	Tiempo de mantenimiento o diseño.
Artefacto	Sistema
Respuesta	Modificación de consultar
Medida de la respuesta	15 días

Tabla 39. Escenario de modificabilidad M1

- **Escenario 4:** Modificar los campos en los datos de los alumnos que se almacenan afectara a los módulos modificar y consultar. El coste de estas modificaciones seria de 30 días.

ELEMENTO	ATRIBUTO/VALOR
Estímulo	Modificar los campos de datos de los alumnos
Fuente del estimulo	Inge. Softw.
Entorno	Tiempo de mantenimiento o diseño
Artefacto	Sistema
Respuesta	Modificación de los campos de datos de los alumnos
Medida de la respuesta	30 días

Tabla 40. Escenario de modificabilidad M2

3.3 Análisis con Arche

Creamos un nuevo proyecto ArchE

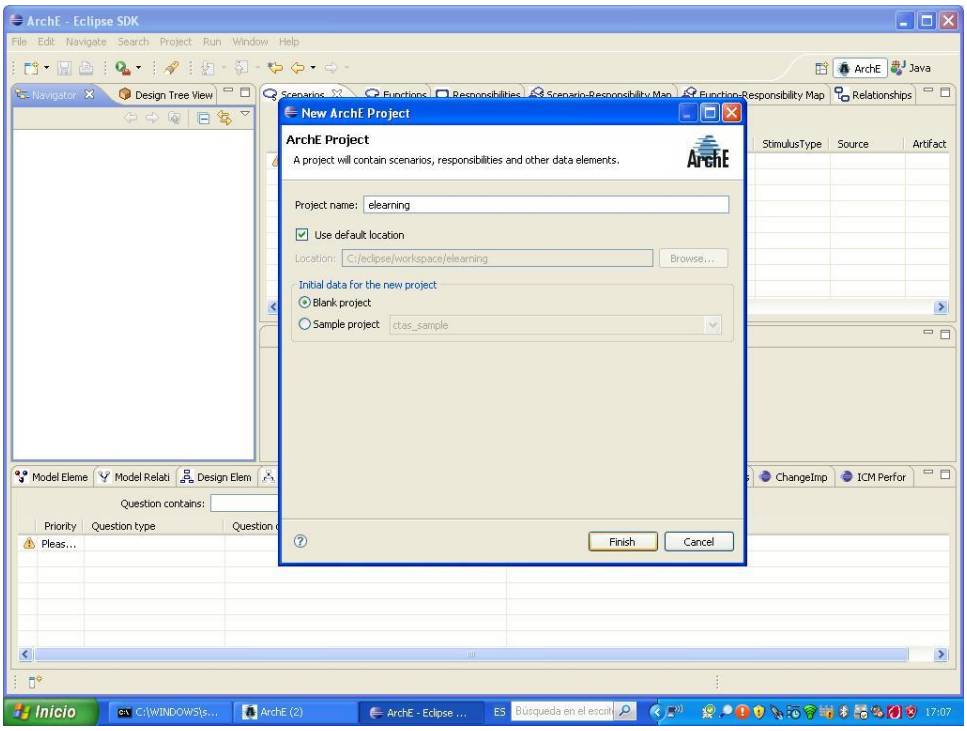


Ilustración 19. Nuevo proyecto ArchE

3.3.1 Definimos funciones y responsabilidades

Empezamos a introducir las funciones. Se les asigna un identificador y un nombre.

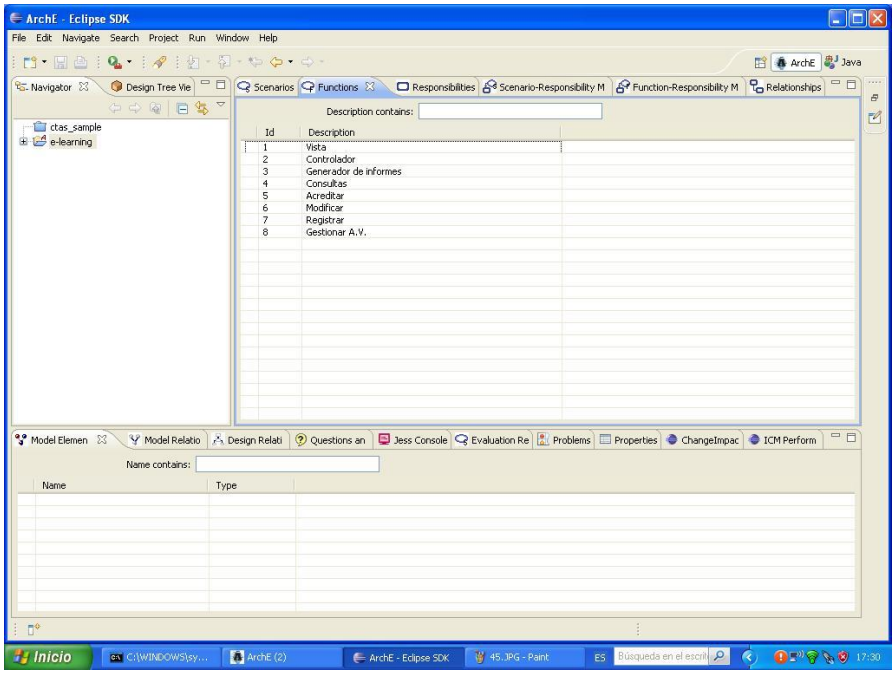


Ilustración 20. Introducimos funciones

Las responsabilidades automáticamente aparecen con el nombre idéntico al de las funciones.

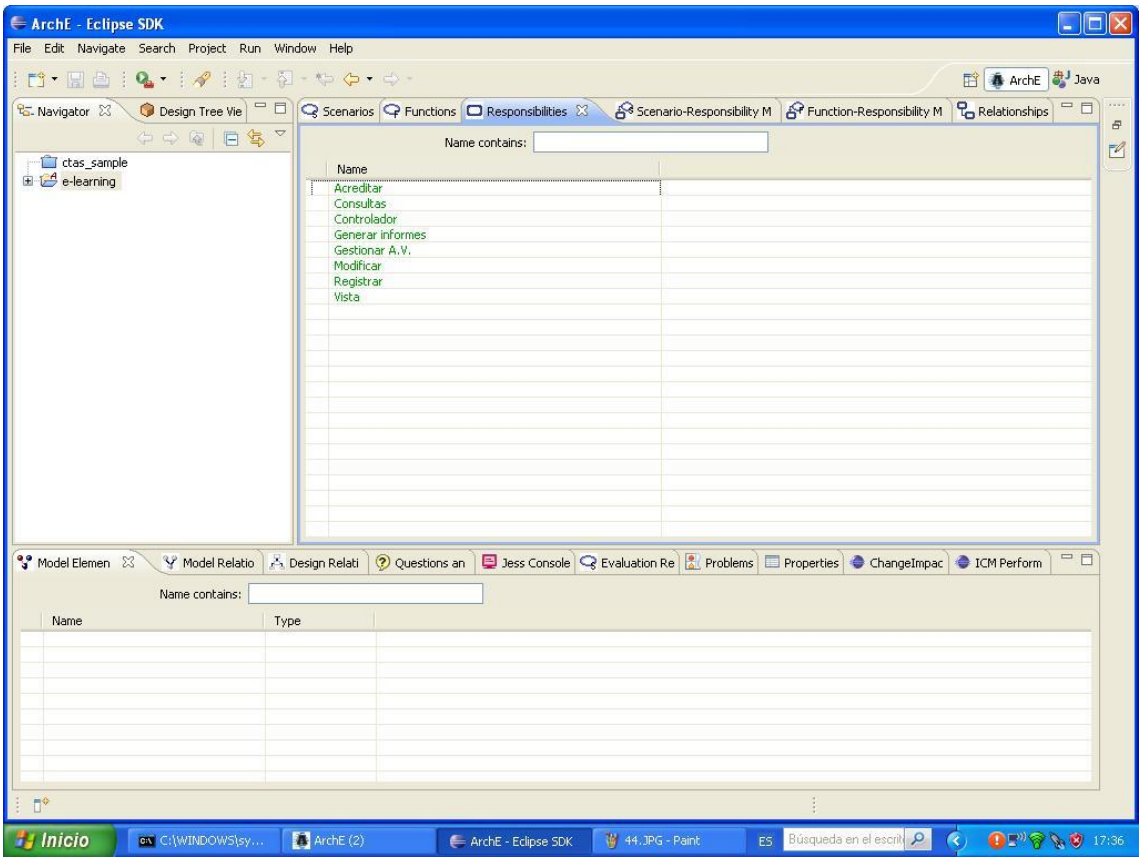


Ilustración 21. Responsabilidades

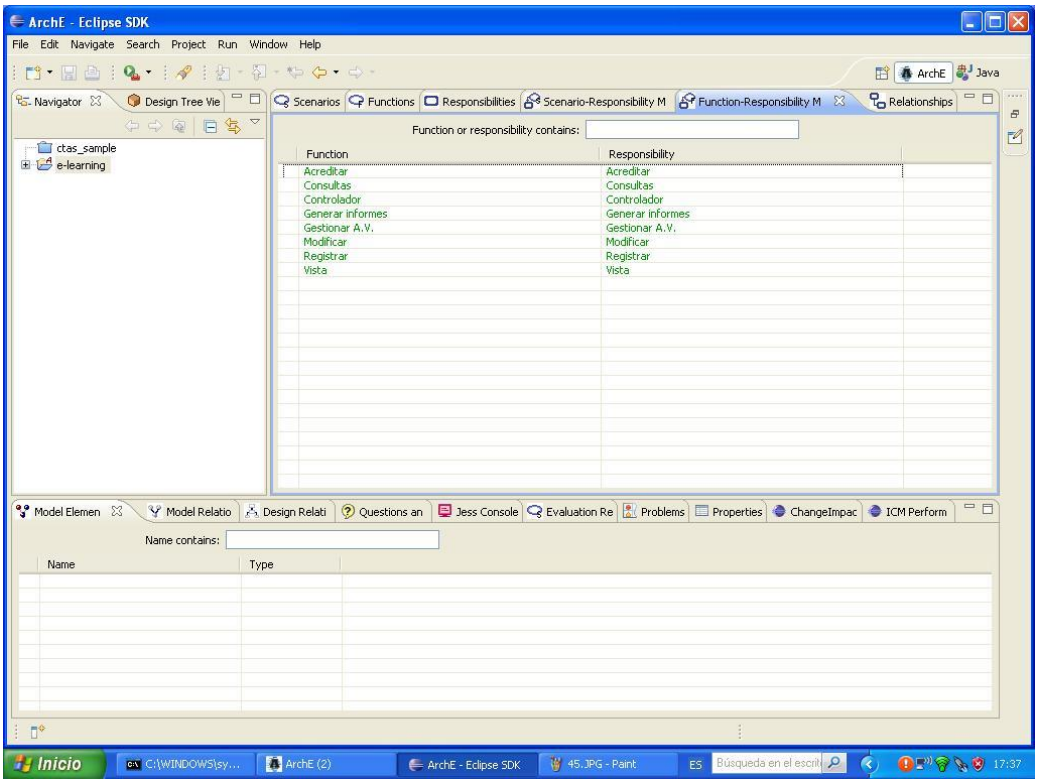


Ilustración 22. Función – responsabilidad

3.3.2 Definimos las relaciones

Esta etapa es de suma importancia, ya que le indicara a Arche como las funciones se interrelacionan entre sí.

Para definir las relaciones nos servimos del siguiente gráfico con las responsabilidades:

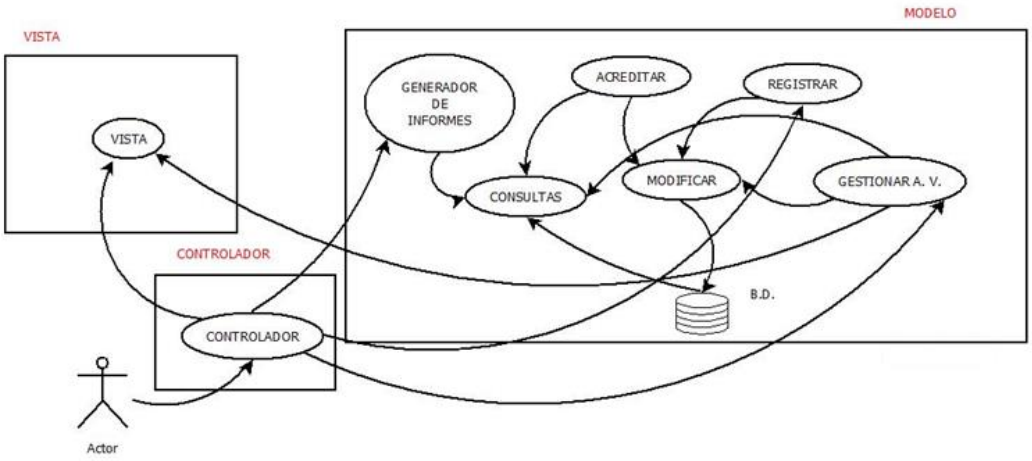


Ilustración 23. Gráfico de responsabilidades

3.3.2.1 Relaciones de modificabilidad

Las relaciones del marco de modificabilidad tienen que ver con cómo afecta la modificación de una responsabilidad a otras con las que guarda algún tipo de relación. En Arche se denomina relación de Dependencia.

CHILD RESPOSABILITY / PARENT RESPONSABILITY	VISTA	CONTROLADOR	GENERAR INFORMES	CONSULTAS	ACREDITAR	MODIFICAR	REGISTRAR	GESTIONAR A.V.
VISTA		D						D
CONTROLADOR			D		D		D	D
GENERAR INFORMES				D				
CONSULTAS								
ACREDITAR				D		D		
MODIFICAR							D	D
REGISTRAR								
GESTIONAR A.V.				D		D		

3.3.2.2 Relaciones de reacción

Relaciones de reacción que se utilizan para el marco de razonamiento de rendimiento y representan que una responsabilidad se ejecuta antes que otra.

CHILD RESPOSABILITY / PARENT RESPONSABILITY	VISTA	CONTROLADOR	GENERAR INFORMES	CONSULTAS	ACREDITAR	MODIFICAR	REGISTRAR	GESTIONAR A.V.
VISTA		R						R
CONTROLADOR								
GENERAR INFORMES		R		R				
CONSULTAS								
ACREDITAR		R		R		R		
MODIFICAR								
REGISTRAR		R				R		
GESTIONAR A.V.		R		R		R		

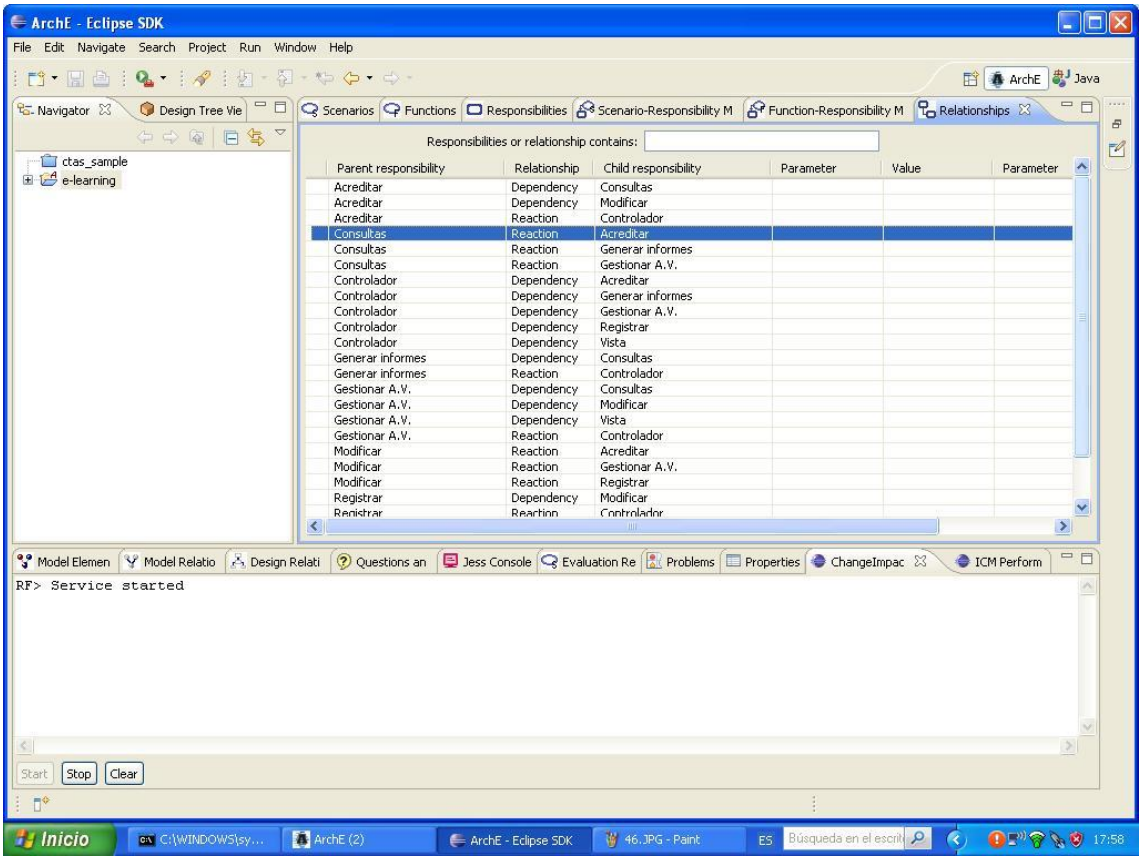


Ilustración 24. Cargamos relaciones en Arche

3.3.3 Definición de los escenarios

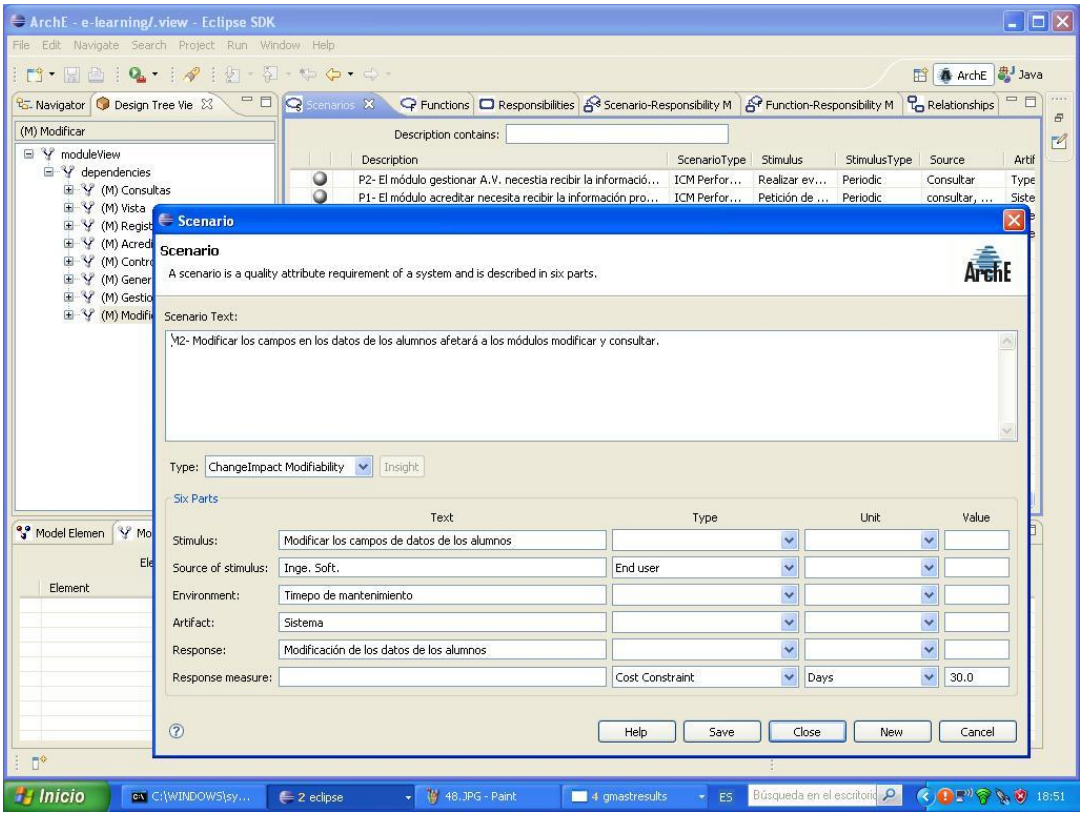


Ilustración 25. Cargamos escenarios 1

En la ilustración 25, se muestra el formulario con los diferentes campos para rellenar. Estos son:

- **Estimulo:** que hace que ocurra el cambio.
- **Fuente del estimulo:** quien lo provoca.
- **Condiciones del sistema**: como se encuentra el sistema en el momento del cambio. En este ejemplo el cambio se realiza en tiempo de mantenimiento.
- **Artefacto:** Al cual afecta
- **Respuesta:** Que finalidad realiza dicho cambio.
- **Tiempo de respuesta:** medida del tiempo que tarda en realizarse el cambio.

Este en concreto es un escenario de modificabilidad y el campo más importante es “Tiempo de respuesta”. Este marcara el objetivo de dicho escenario.

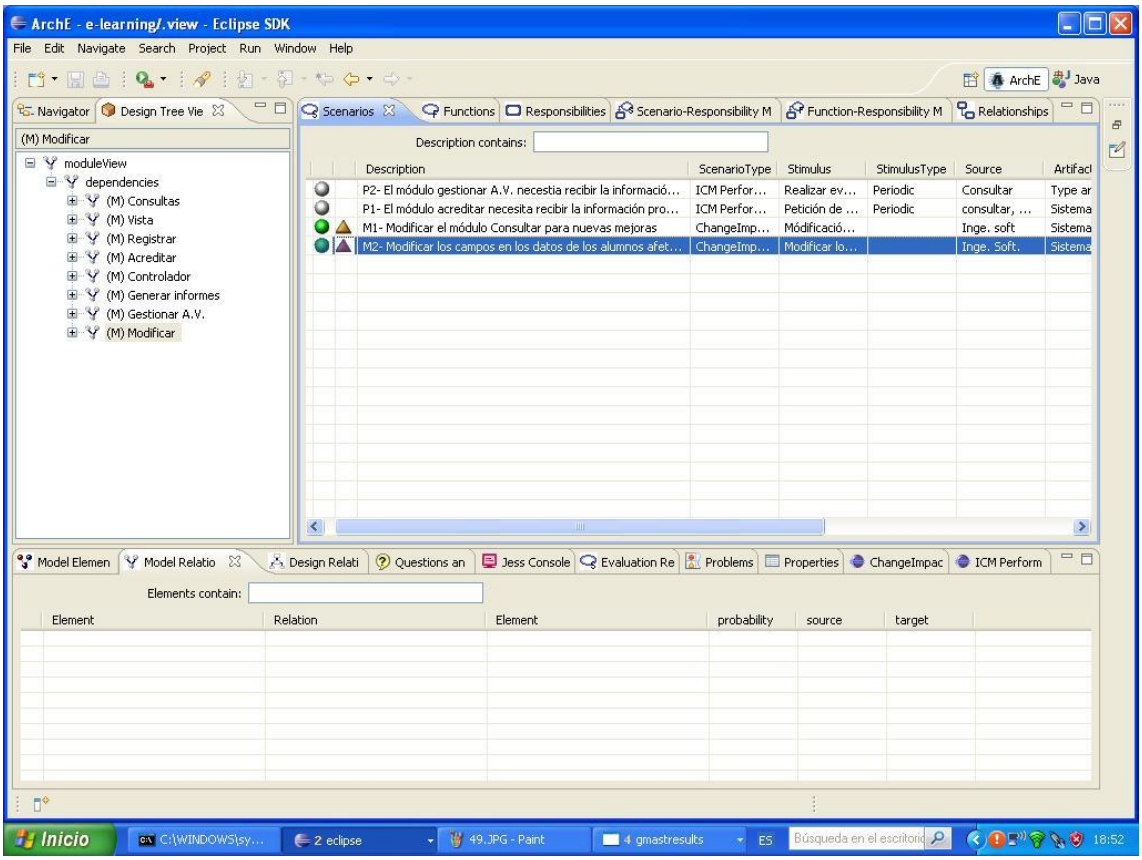


Ilustración 26. Cargamos escenarios 2

3.3.4 Mapeo de escenarios a responsabilidades

Esta etapa es crucial, ya que en ella se definen las funcionalidades a las que afecta cada escenario. Arche las usa para calcular si los escenarios se satisfacen o no. Y así puede inferir que el sistema sea panificable o no. En la tabla 41 observamos las relaciones entre módulos en cada escenario de calidad propuesto.

	M1	M2	P1	P2
Vista				
Controlador				
Consultas		X	X	X
Modificar		X	X	
Gestionar A.V.	X			X
Generar informes	X			
Acreditar	X		X	
Registrar				

Tabla 41. Mapeado escenarios responsabilidades

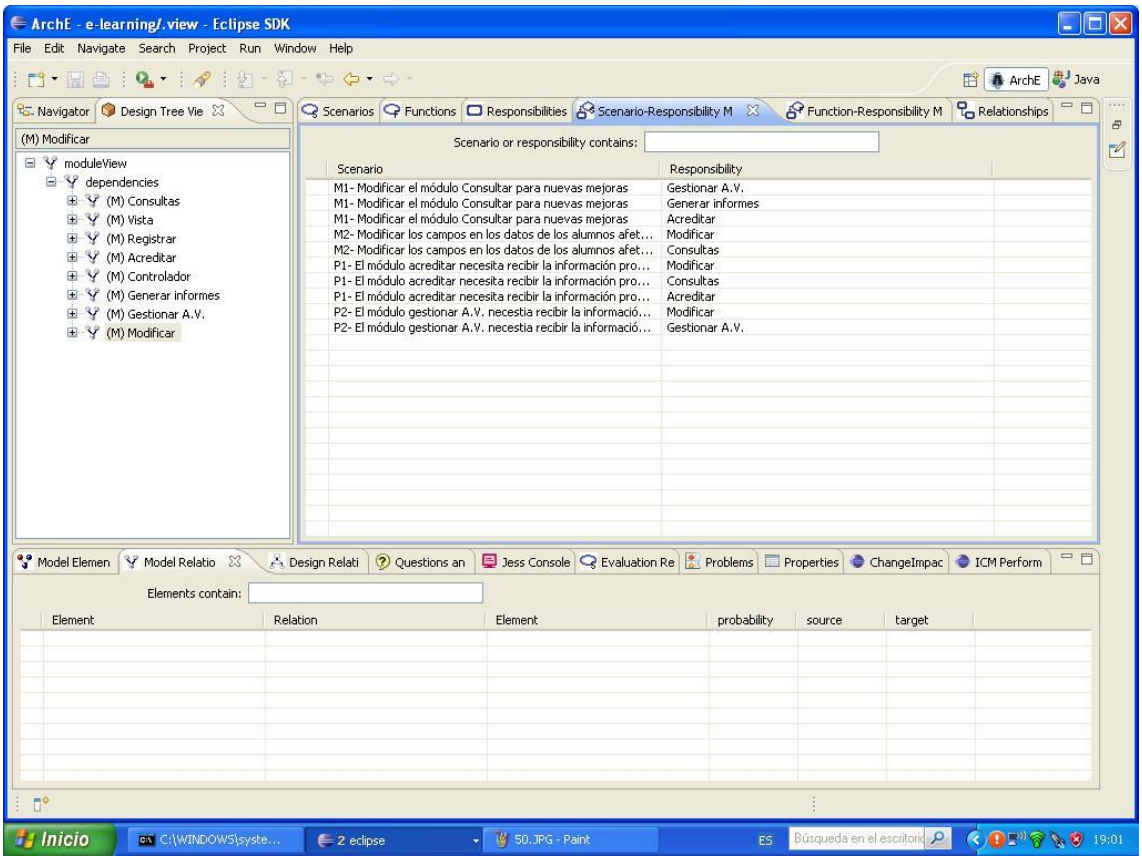


Ilustración 27. Mapeado escenarios responsabilidades

En la ilustración 27, observamos cada escenario, tanto los de modificabilidad como los de rendimiento, y con las funcionalidades a las que afecta.

3.3.5 Definimos las responsabilidades

Definimos los atributos de cada responsabilidad. Con esto indicamos al sistema el tiempo de coste indicado en días y el tiempo de ejecución indicado en milisegundos.

Nombre	Coste cambio (días)	Tiempo ejecución(msecs)
Acreditar	2	1
Consultas	5	5
Generar informes	2	5
Gestionar A.V.	15	10
Modificar	5	5
Registrar	2	5
Vista	10	5
Controlador	15	10

Aunque estos valores son arbitrarios para el ejemplo, hemos intentado ajustarlos a la realidad asignando costos de cambio más altos a las funcionalidades más complejas.

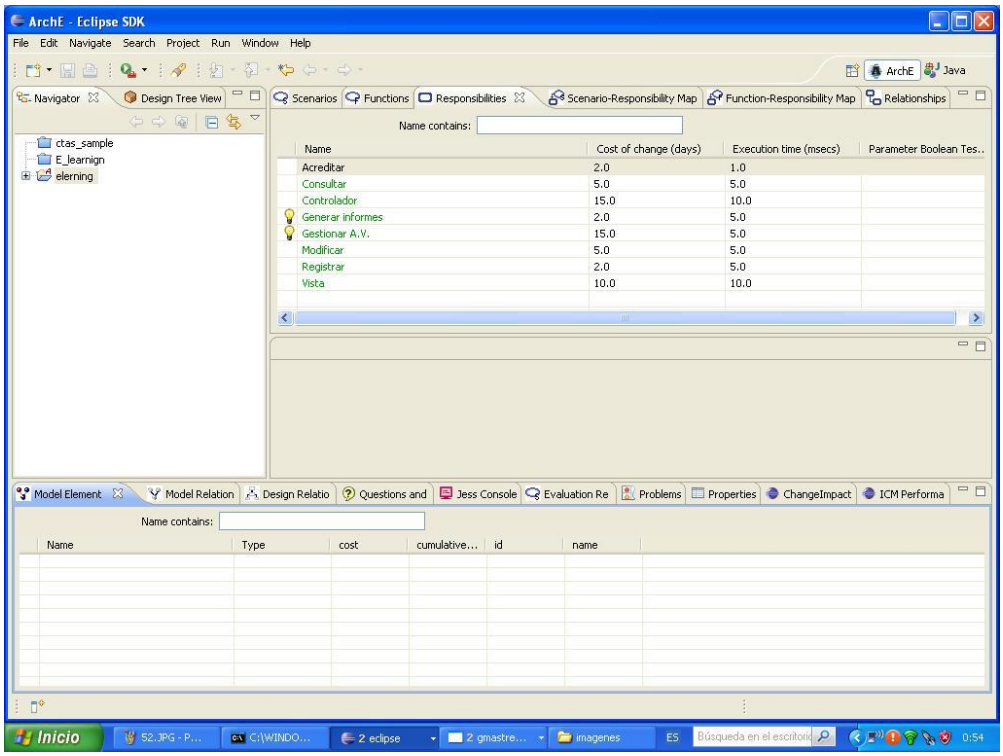


Ilustración 28. Responsabilidades definidas con costo de cambio y tiempo de ejecución

3.3.5 Evaluación con Arche y aplicación de tácticas

Comenzamos con la evaluación de los diferentes escenarios propuestos. Para ofrecernos una información visual de la valoración Arche usa un sistema de colores asociado a una forma geométrica. De la siguiente manera:

- Para indicar si el escenario se cumple o no:
  - Un circulo que puede ser:
    - Rojo: no se cumple.
    - Verde: Si se cumple
- Para indicar el estado del escenario después de la última modificación:
  - Un triangulo:
    - Verde: si ha mejorado las condiciones.
    - Amarillo: si no ha afectado.
    - Rojo: so ha empeorado las condiciones.

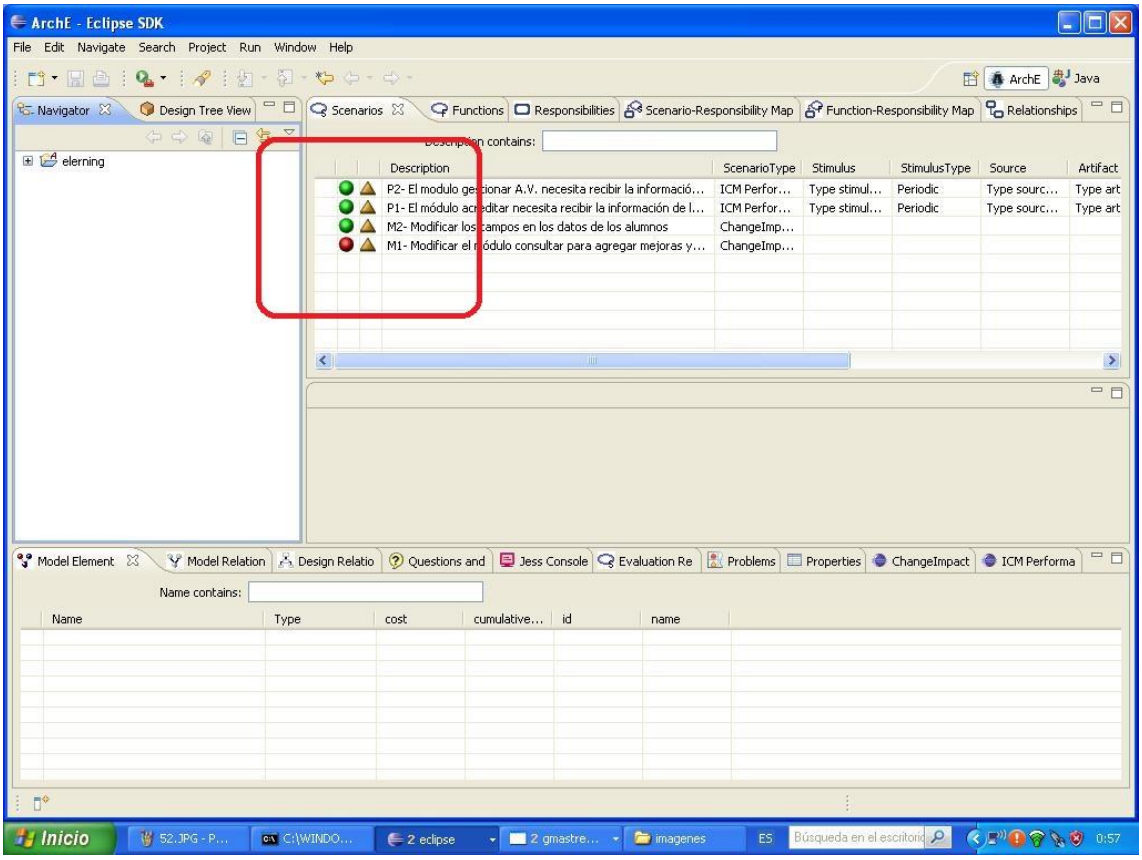


Ilustración 29. Evaluación

Como podemos observar Se cumplen todos los escenarios menos el M1. Ya que indicamos 15 días para el cambio. Pero la suma de los cambios de los módulos que afecta arche nos indica que es 19.

Veamos que tácticas y sugerencias nos propone Arche

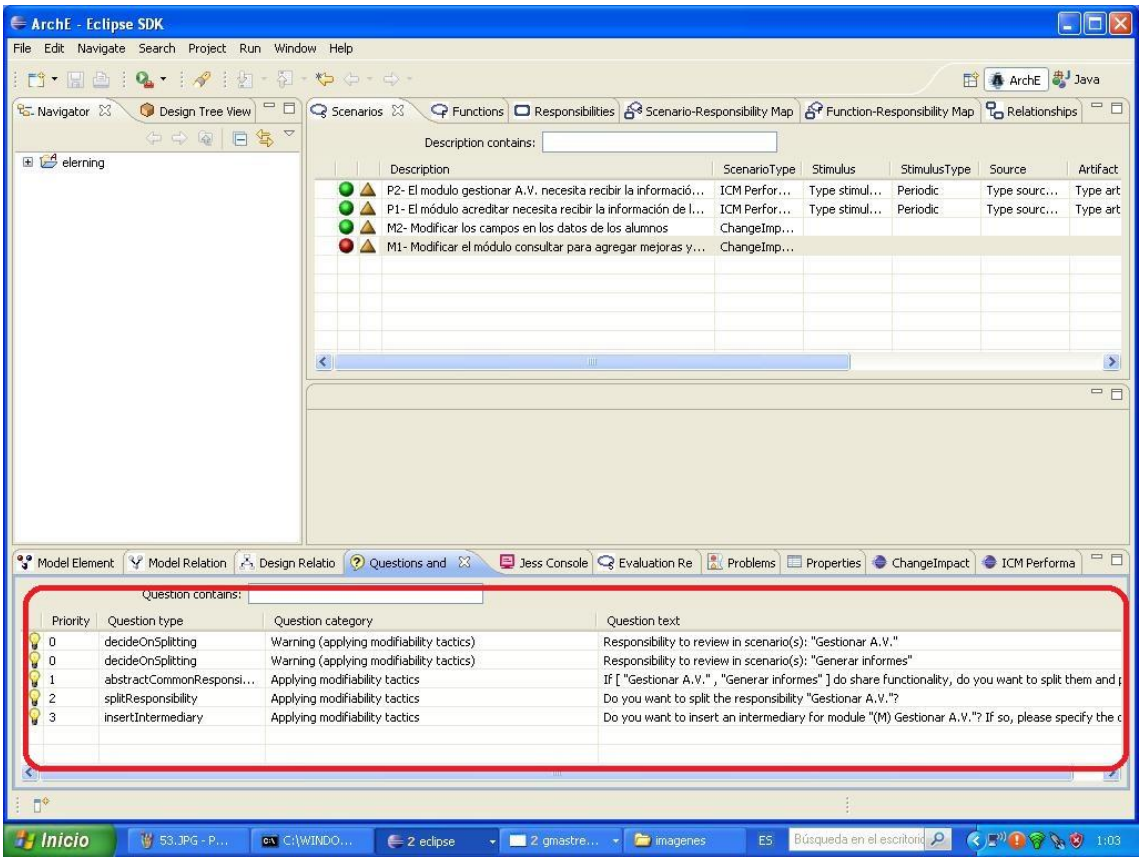


Ilustración 30. Tácticas y sugerencias Arche

En el área de notificaciones en la pestaña "*Questions and tactics*" se nos indican:

- Alertas: son sugerencias generales para el sistema, con prioridad 0.
- Tácticas: Son propuestas a fin de mejorar las condiciones para que se cumplan los escenarios.

Como podemos observar nos aparecen 2 warnings y 3 tácticas:

Warnings

- "decideOnSplitting" Responsability to review in "Gestionar A.V."
  - Nos indica que beneficiaría el sistema el pensar en dividir las responsabilidades de la función "Gestionar A.V.".

- "decideOnSplitting" Responsability to review in "Generar informes"
  - Nos indica que beneficiaría el sistema el pensar en dividir las responsabilidades de la función "Gestionar A.V.".

Tácticas

Táctica 1- Nos indica que si "Gestionar A.V." y "Generar informes" comparten funcionalidades por qué no sacar los factores comunes a otra función..

Táctica 2- Dividir la responsabilidad de "Gestionar A.V."

Táctica 3- Nos indica que si queremos añadir un módulo intermediario para "Gestionar A.V."

En la siguiente figura Arche nos hace una predicción de cómo afectaran las diferentes tácticas a los diferentes escenarios. Utilizando de nuevo la simbología por colores: verde para mejorar, amarillo no afecta y rojo para empeora.

Se Observa que es recomendables las táctica 2 ya que esta en verde para todos los escenarios, la 3 no es muy beneficiosa para el diseño ya que esta de color rojo.

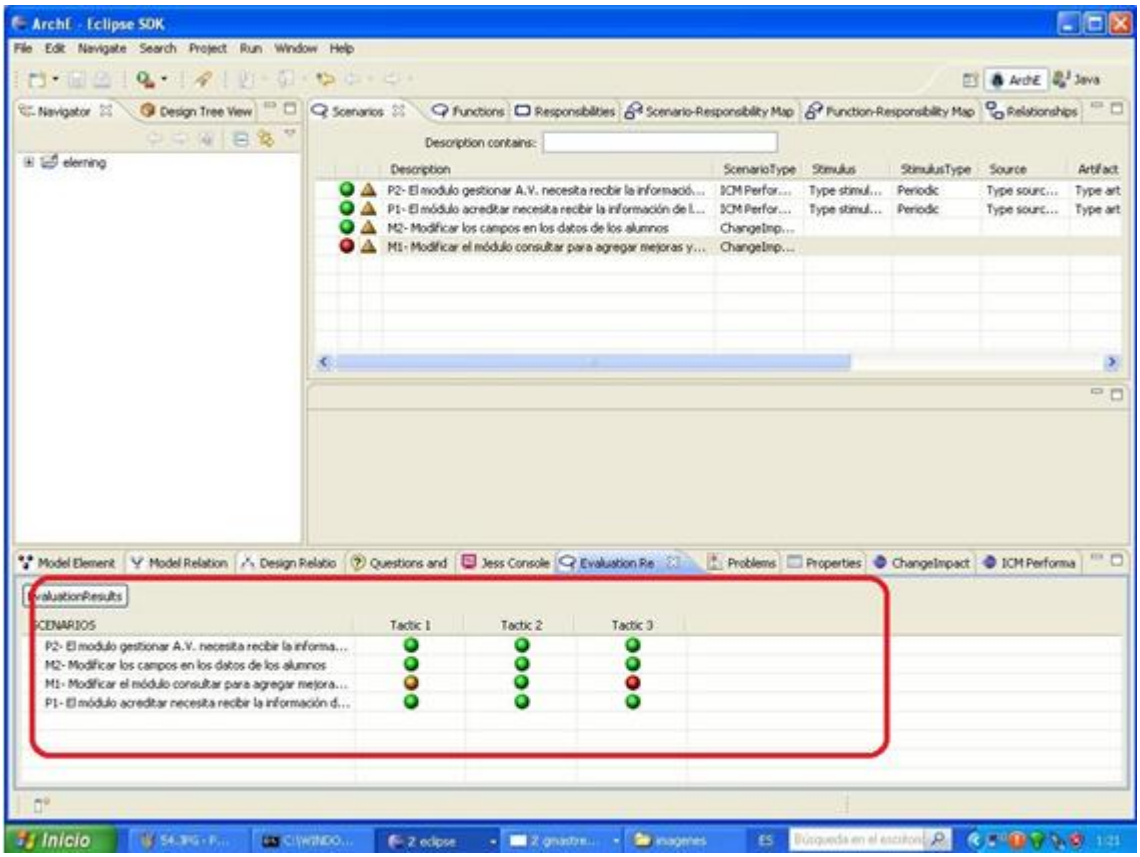


Ilustración 31. Vista calificación de las tácticas

Usamos la táctica 2

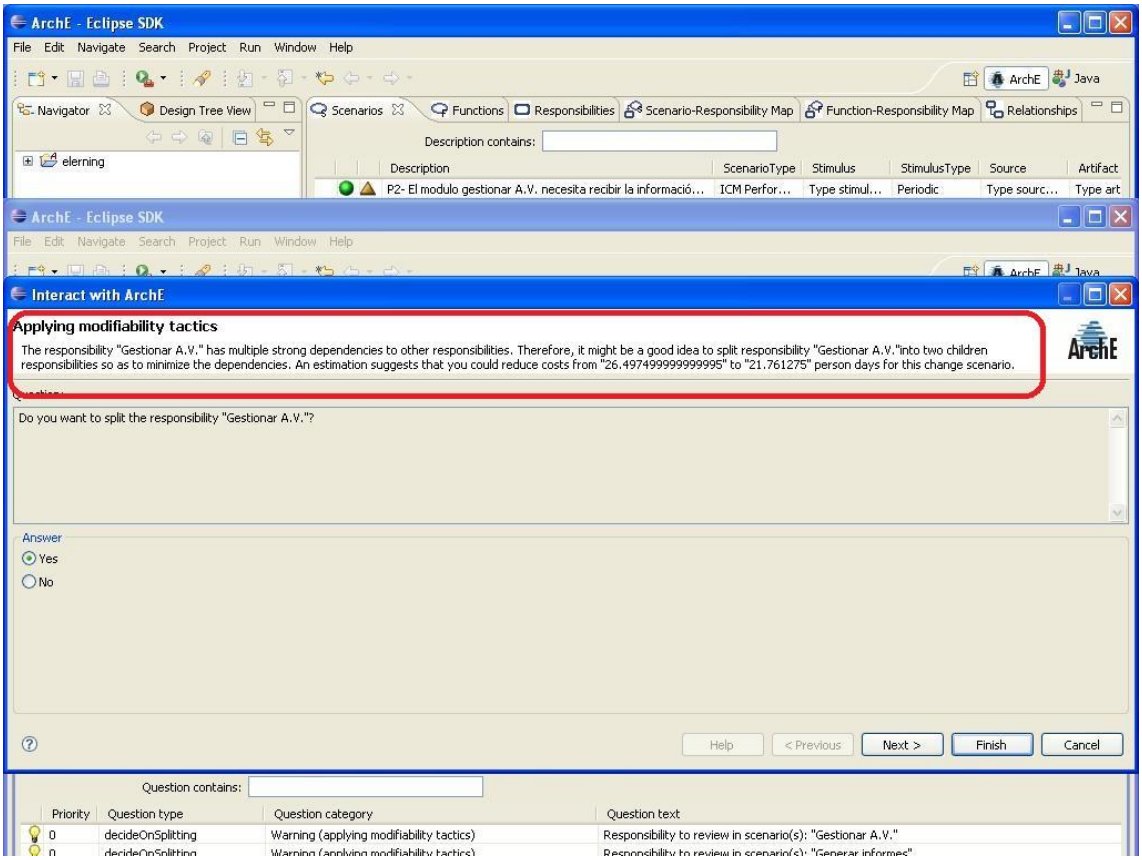


Ilustración 32. Usamos táctica 2

Se nos indica que "Gestionar A.V." tiene muchas dependencias fuertes con otras responsabilidades. Y que sería buena idea dividir la responsabilidad de esta en 2 módulos 'hijos'.

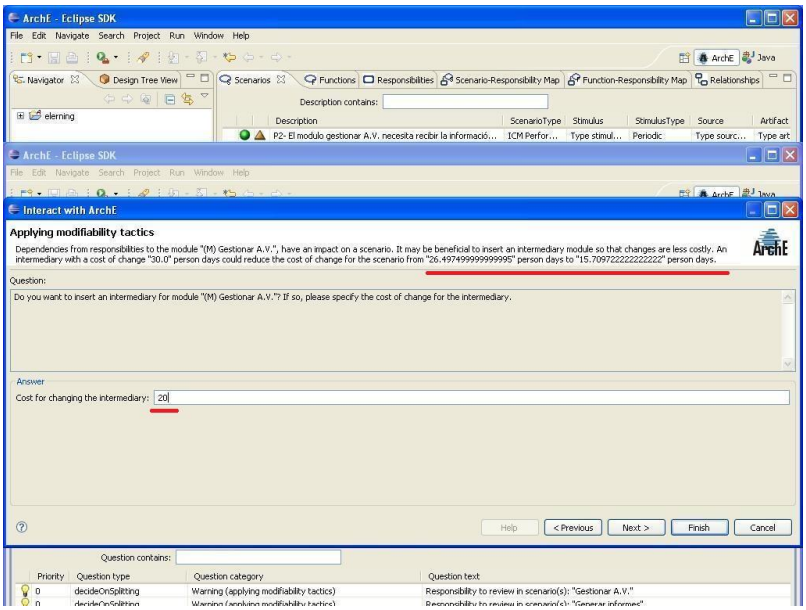
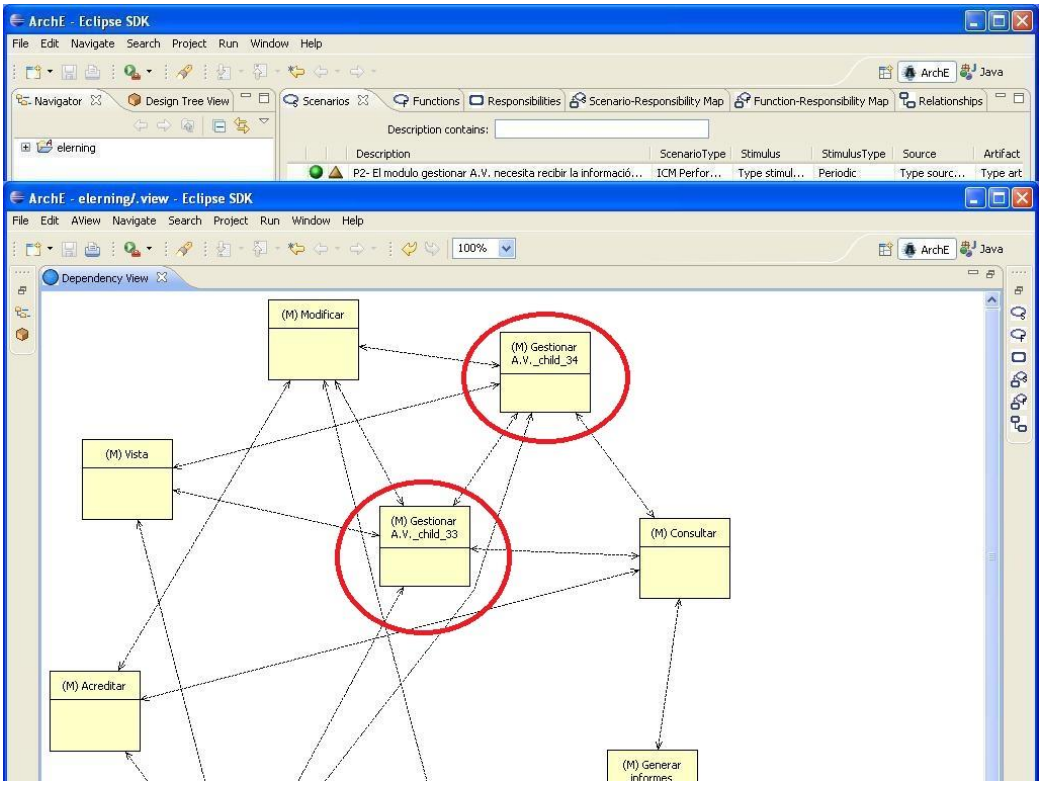


Ilustración 33. Recomendaciones y asignación de coste



lustración 34. Vista del módulo dividido

Se han introducido los dos módulos hijos en los que se ha dividido la responsabilidad de “Gestionar A.V.”

Como es de esperar la aplicación nos insta a redefinir los costes, de las funciones ya que han variado respecto del planteamiento original.

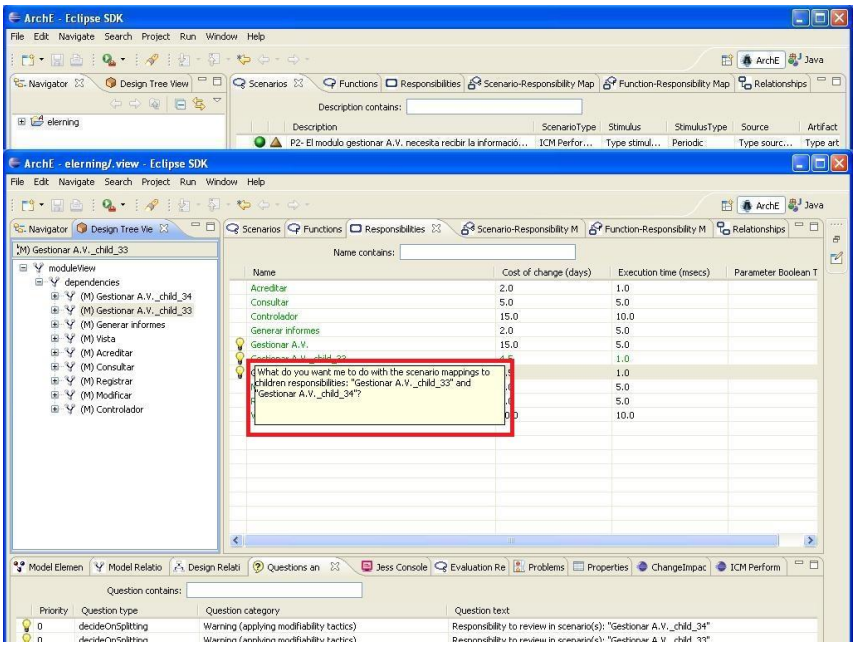


Ilustración 35. Redefinir costes tras la división

Redefiniremos los costes dividendo en partes iguales entre los dos 'hijos' de "Gestionar A.V.", por que lógicamente si las funcionalidades se dividen los costes de modificarlas también.

Una vez hecho esto, se vuelven a evaluar los escenarios automáticamente. Y vemos que seguimos sin cumplir el escenario M1.

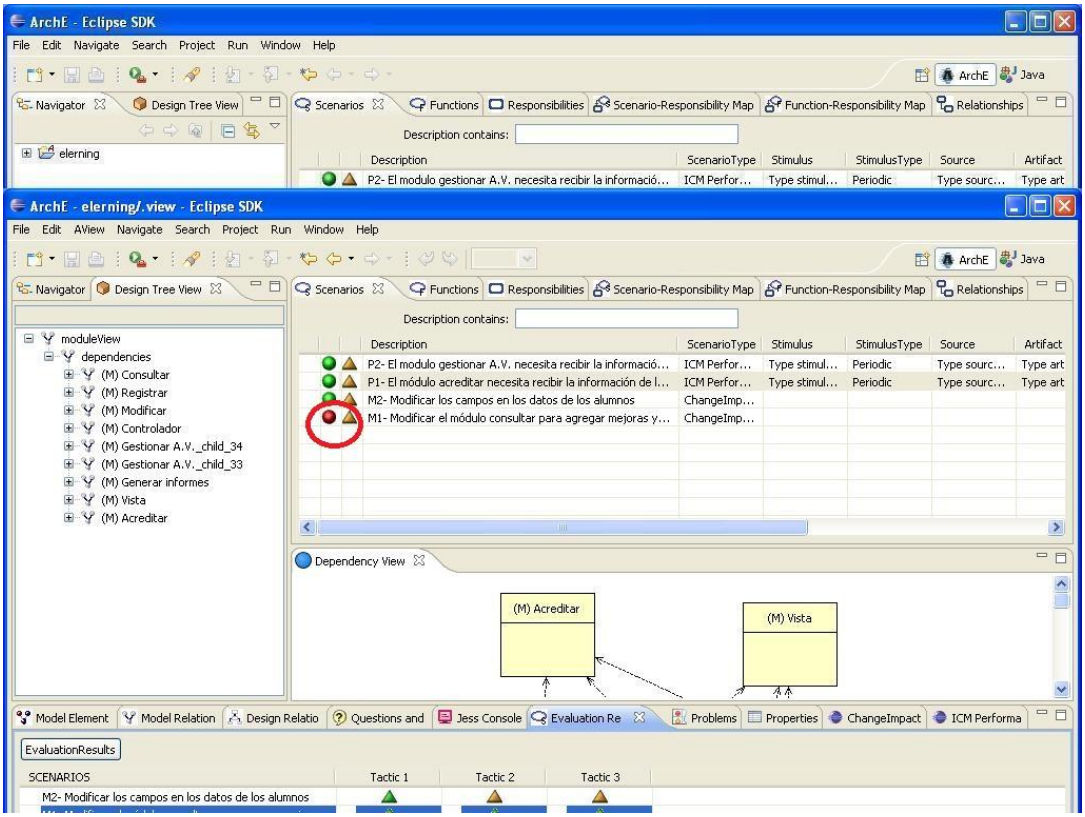


Ilustración 36. Seguimos sin cumplir el escenario

Aun con los cambios efectuados seguimos sin satisfacer el escenario M1.

De nuevo observamos las tácticas propuestas y elegimos usar la táctica 1:

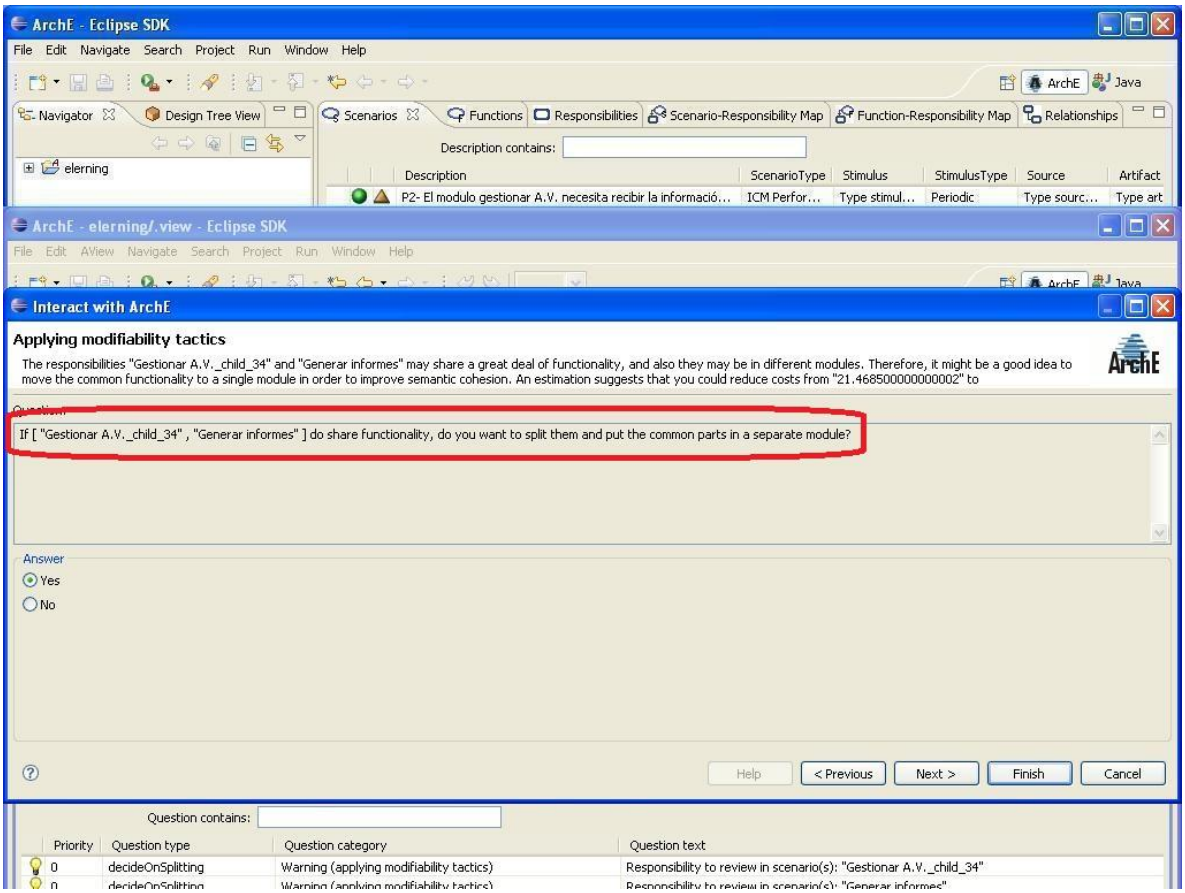


Ilustración 37. Aplicar táctica 1

En esta ocasión la táctica nos sugiere que “Gestiona A.V._child_34” y “Generar informes” comparten funcionalidades. Deberíamos generar un nuevo módulo que de soporte a estas funcionalidades. Tiene lógica ya que así vamos obteniendo módulos más específicos cuyo coste de modificabilidad será menor.

A continuación se nos indica que tenernos que revisar la responsabilidad de “Generar informes” y “Gestionar A.V._child34”

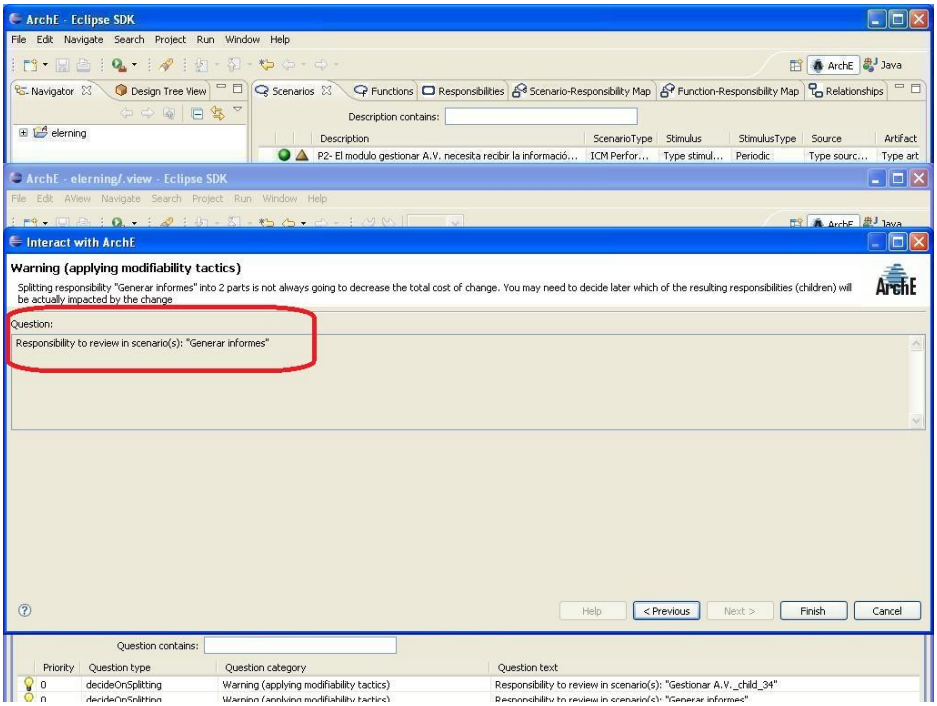


Ilustración 38. Revisar responsabilidades

La revisamos dividiendo los costes a partes iguales. Lo que indica que las funcionalidades también se dividen. Una vez hecho esto ya se cumplen todos los escenarios incluido M1:

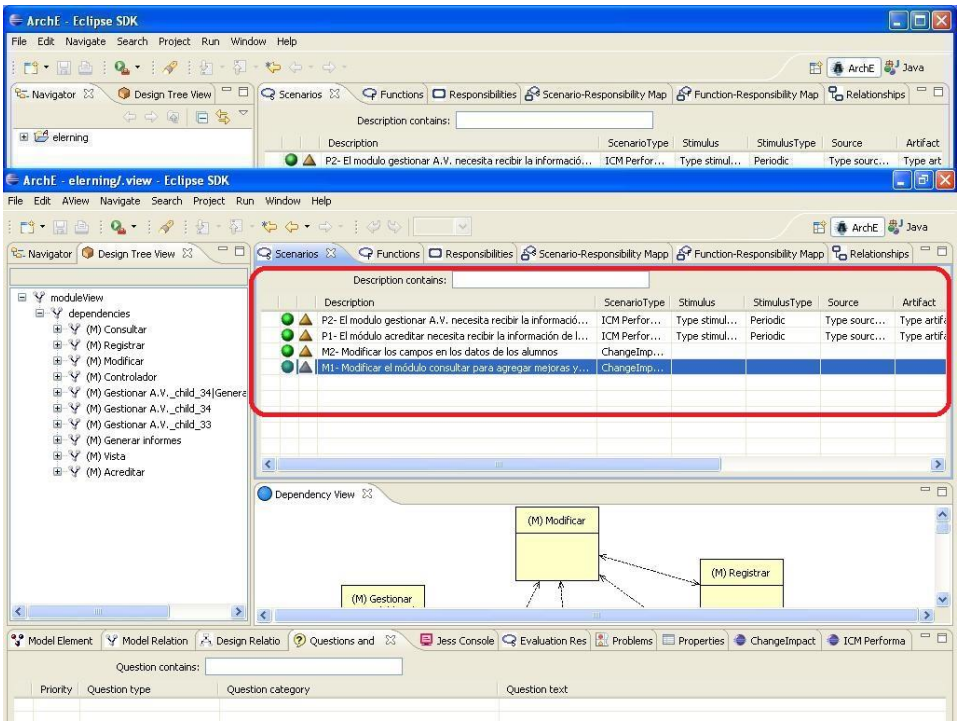


Ilustración 39. Todos los escenarios cumplidos

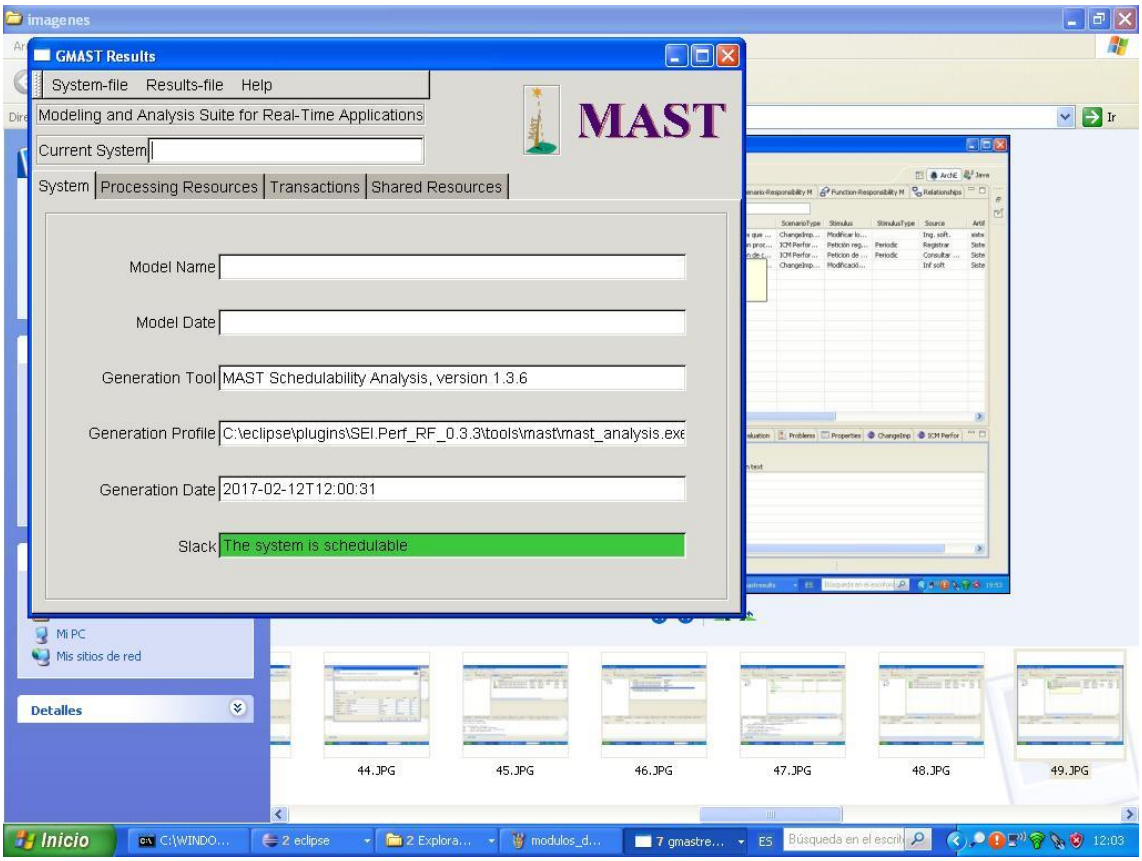


Ilustración 40. MAST

Ahora el sistema es plenamente panificable, gracias a las tácticas propuestas por ArchE.

3.4 Resultados

-

Para observar los resultados obtenidos, utilizaremos el diagrama de módulos original y el que ha sido resultado después de usar Arche.

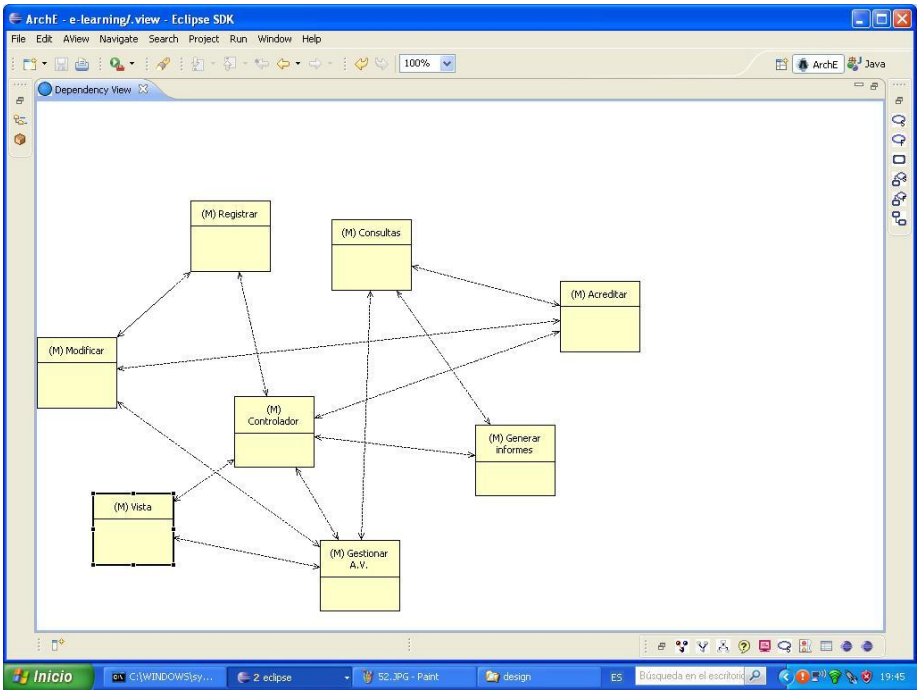


Ilustración 41. Diagrama original

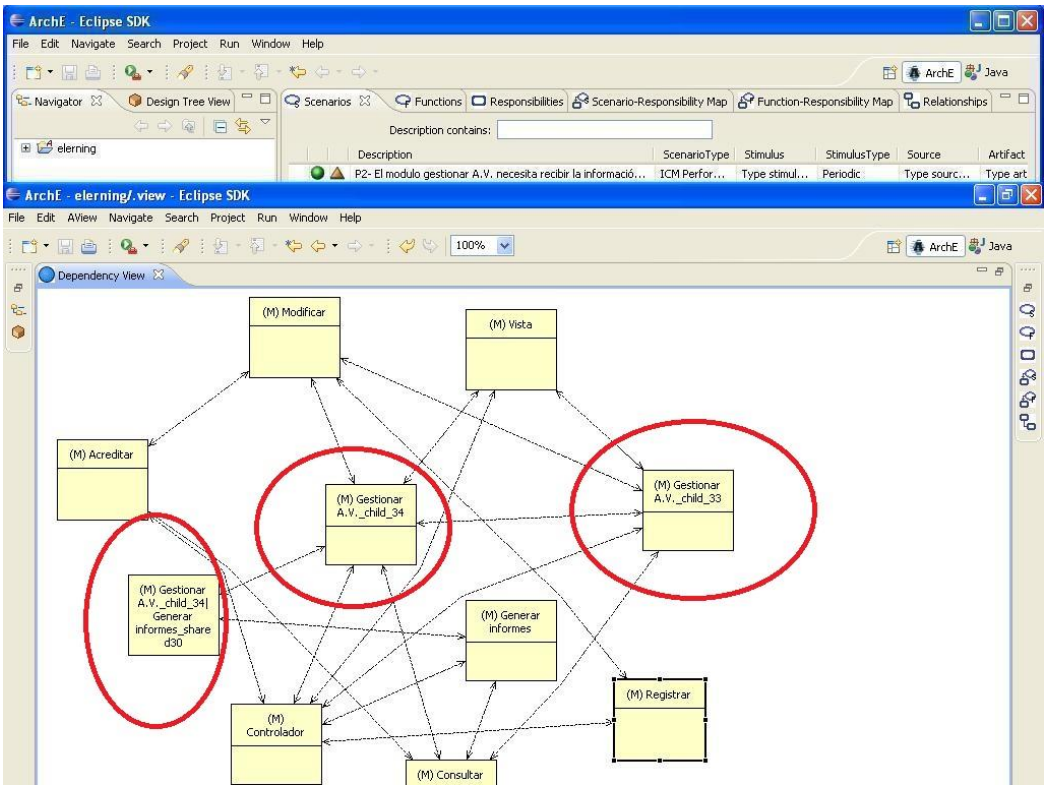


Ilustración 42. Diagrama resultado

Las recomendaciones que obtenemos de Arche para mejorar nuestro diseño y que cumpla todos los escenarios de calidad propuestos son:

- Dividir las responsabilidades de "Gestionar A.V.". Para así poder mejorar los tiempos de respuesta y los costos de modificabilidad.

- Extraer las características compartidas de "Generar informes" y "Gestionar A.V." a un nuevo modulo para hacer una mejor distribución de las responsabilidades.

3.5 Valoración de los resultados

Llegados a este punto me dispongo a realizar una comparación del sistema antes de implementar las mejoras para cumplir los requisitos de calidad descritos en los escenarios y después de su mejora y análisis con Arche.

Los atributos de cada responsabilidad antes de las mejoras eran:

Nombre	Coste cambio (días)	Tiempo ejecución(msecs)
Acreditar	**2**	**1**
Consultas	**5**	**5**
Generar informes	**2**	**5**
Gestionar A.V	**15**	**10**
Modificar	**5**	**5**
Registrar	**2**	**5**
Vista	**10**	**5**
Controlador	**15**	**10**

Después de implementar las mejoras propuestas por las tácticas de Arche. Que en nuestro caso nos divide "Gestionar A.V." en dos responsabilidades diferentes. Crea una nueva con las características compartidas de "Gestionar A.V" y "Generar informes".

Que en base a la ilustración 15 los nuevos módulos, tendrían las siguientes funciones repartidas:

- Gestionar A.V._child34:
  - Gestionar foros.
  - Gestionar cursos.
  - Gestionar notas.
- Gestionar A.V._child33:
  - Asignar evaluación.
- Gestionar A.V._child34_Generar informes Share
  - Generar informes académicos.

Gestionar A.V._child34_Generar informes Share

Nuestra nueva tabla de atributos de cada responsabilidad después de las modificaciones realizadas en el diseño, quedaría:

Nombre	Coste cambio (días)	Tiempo ejecución(msecs)
Acreditar	**2**	**1**
Consultas	**5**	**5**
Generar informes	**1**	**3**
Modificar	**5**	**5**
Registrar	**2**	**5**
Vista	**10**	**5**
Controlador	**15**	**10**
Gestionar A.V._child34	**9**	**4**
Gestionar A.V._child33	**5**	**5**
GestionarA.V._child34_Generar informes Share	**2**	**3**

Y con estas nuevas responsabilidades y módulos derivados de las tácticas propuestas por Arche. Obtendremos las mejoras en el diseño para que se cumplan los escenarios de calidad propuestos.

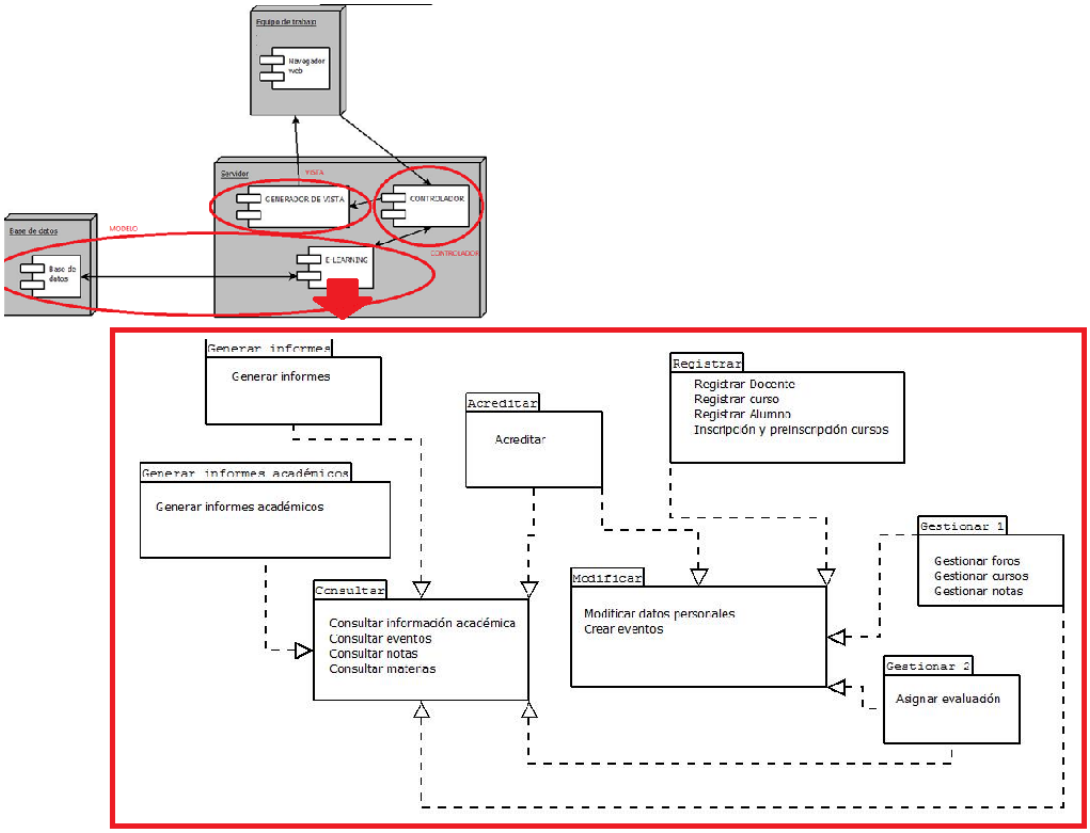


Ilustración 43. Arquitectura tras los cambios

Como se puede observar en la ilustración 43, así quedaría la nueva arquitectura que da soporte al sistema. Cumpliría los requisitos de calidad especificados en los escenarios propuestos de modificabilidad. El módulo “Gestionar A.V.” se subdivide en dos (Gestionar 1 y Gestionar 2) repartiendo sus funcionalidades entre estos. Y aparece otro nuevo módulo “Generar informes académicos” derivado de las funcionalidades compartidas entre “Generar informes” y “Gestionar 1”.

4. Conclusiones y trabajos futuros.

4.1 Conclusiones

Después de realizar la evaluación con Arche de la arquitectura propuesta y aplicar las mejoras propuestas al diseño, puedo concluir que hemos mejorado sustancialmente el atributo de **modificabilidad** del diseño. Ya que hemos pasado de no alcanzar los requisitos impuestos por el escenario e modificabilidad M1 a cumplirlo. A la vista de las tablas de costos de antes y después de las mejoras introducidas por Arche:

Nombre	Coste cambio (días)	Tiempo ejecución(msecs)
Acreditar	2	1
Consultas	5	5
Generar informes	2	5
Gestionar A.V	15	10
Modificar	5	5
Registrar	2	5
Vista	10	5
Controlador	15	10

Antes de la evaluación con Arche

Nombre	Coste cambio (días)	Tiempo ejecución(msecs)
Acreditar	2	1
Consultas	5	5
Generar informes	1	3
Modificar	5	5
Registrar	2	5
Vista	10	5
Controlador	15	10
Gestionar A.V._child34	9	4
Gestionar A.V._child33	5	5
GestionarA.V._child34_Generar informes Share	2	3

Después de las mejoras introducidas por Arche

Hemos pasado de un costo total de modificación del módulo Gestionar A.V. de 15 días a subdividirlo en dos cuyos costos son 9 y 5 días. Dependiendo de qué módulo necesitemos modificar una mejora del 40% menos de tiempo de modificación y 66,66% respectivamente. Además una mejora del 50% en Generar informes.

Nombre	Coste cambio antes(días)	Coste cambio después (días)	Porcentaje de mejora
Generar informes	**2**	**1**	**50%**
Gestionar A.V._child34	**15**	**9**	**40%**
Gestionar A.V._child33	**15**	**5**	**66,6%**

Cuantificación de las mejoras de modificabilidad introducidas

Al subdivir las funcionales de algunos módulos también experimentamos una mejora en el **rendimiento** del sistema ya que los tiempos de ejecución descienden al reducirse la complejidad de ejecución de estos.

Nombre	Tiempo ejecución antes (msecs)	Tiempo ejecución después(msecs)	Porcentaje de mejora
Generar informes	**5**	**3**	**40%**
Gestionar A.V._child34	**10**	**4**	**60%**
Gestionar A.V._child33	**10**	**5**	**50%**

A parte de las cuestiones relativas a la arquitectura evaluada en el presente me gustaría resaltar como también como conclusión los siguientes aspectos que he observado durante la realización del mismo:

- Cuando los diseños adquieren cierta complejidad, el uso de sistemas expertos de evaluación como *ArchE* lejos de ser un elemento recomendable, se convierten en imprescindibles. Ya que el manejo de gran cantidad de requisitos y la adaptación/conversión de estos a los atributos de calidad suponen una tarea que precisa de una meticulosidad extrema y este tipo de herramientas correctamente alimentadas suponen una gran ayuda para el ingeniero del software para poder realizar estudios de sus diseños con los diferentes escenarios de calidad de prueba que se propongan.

- En cuanto a la herramienta ArchE en sí, una vez que te familiarizas con su uso se hace relativamente sencillo. Es una lástima que solo disponga de marcos de razonamiento para escenarios de modificabilidad y de rendimiento. La mayor utilidad que observo, es que en proyectos complejos te obliga a hacer un exhaustivo repaso a tu diseño a la vez que vas incorporando los datos al sistema y que al tener en cuenta las dependencias entre módulos evalúa el impacto de las diferentes tácticas en resto de escenarios. Siendo esto último de gran valor a la hora de elegir soluciones concretas. Lo que confiere una gran utilidad de este tipo de herramientas para el ingeniero de software.

4.2 Trabajos futuros

En trabajos futuros creo que sería interesante el estudio y comparativa de otras herramientas que ayuden al diseño de arquitecturas software. Sobre todo herramientas en las que se puedan evaluar otros atributos de calidad que ArchE no permite. También cabe destacar que existen diferentes iniciativas colaborativas en el mundo de la ingeniería software que tratan profusamente esta temática. Una de ellas es [10] *Symposium on Software Performance* (SSP) una reunión de expertos en la materia centrada principalmente en lo referente al rendimiento.

Otros ejemplos de herramientas de uso actual similares a ArchE son:

- **Palladio** [11]: Herramienta de diseño que nos da soporte en lo referente a los atributos de: rendimiento, confiabilidad, mantenimiento y costes.

Ilustración 44. Palladio

- **AADL** (Architecture Analysis and Design Language) [12]: especializado en desarrollo en sistemas de tiempo real. Usado en mayor medida en sistemas de aviónica y en automóviles.

Realizando un estudio comparativo de varias herramientas para un mismo diseño se podría obtener una información muy valiosa para los ingenieros de software, ya que quedaría plasmado como trabaja cada herramienta y ayudaría a seleccionar la que más se adapte a las necesidades de cada proyecto.

5. Bibliografía.

[1] "Software Architecture in Practice, 2nd Edition". Len Bass, Paul Clements y Rick Kazman. SEI Series in Software Engineering, Ed. Addison-Wesley 2003.

[2] "An Introduction to Software Architecture". David Garlan and Mary Shaw. School of Computer Science, Carnegie Mellon University. January 1994 (CMU-CS-94-166).

[3] "ArchE Version 2.1 User's Guide Version 1.0". Software Engineering Institute, Carnegie Mellon University, 2007.

[4] http://resources.sei.cmu.edu/asset_files/Presentation/2007_017_001_22589.pdf

[5] ANDERSON, T. y ELLOUMI, F. (2004). Theory and Practice of Online Learning. Athabasca University.

[6] V. Vonk, M. Wasmann, D. Sidharta "Architecture for an Online Learning System" Utrecht University

[7] Craig Larman "UML y patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado", Segunda edición Pearson Education 2003.

[8] Roger S. Pressman "Ingeniería del software – un enfoque práctico", séptima edición McGraw Hill 2010.

[9] Henry H. Liu "Software performance and scalability. A quantitative approach", Wiley 2009.

[10] www.performance-symposium.org

[11] www.palladio-simulator.com

[12] www.aadl.info

6. Glosario.

JRE- Java runtime environment

JDK- Java development kit

Xml- extensible markup language

BD- Base de datos