

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

Analysis Report



ACME SOFTWARE FACTORY

OUR FIRST PROJECT IN D&T

Grado en Ingeniería Informática – Ingeniería del Software


Diseño y Pruebas 2

Curso 2023 – 2024

Grupo de prácticas: C1-009


Autores por orden alfabético:

García Galocha, Rafael David

	Diseño y Pruebas II Acme-Software-Factory
	Analysis Report


Índice de contenido

1. Resumen ejecutivo.....	3
2. Tabla de revisiones	4
3. Introducción	5
4. Contenido	6
5. Conclusiones	8
6. Bibliografía.....	9

	Diseño y Pruebas II Acme-Software-Factory
	Analysis Report


1. Resumen ejecutivo

En este informe detallaré mi análisis para los requisitos de los entregables correspondientes al Student 3 que requieran de interpretaciones adicionales.

	Diseño y Pruebas II Acme-Software-Factory
	Analysis Report

2. Tabla de revisiones

Fecha	Versión	Descripción
03/04/2024	1.0	Primera versión del documento.
17/04/2024	2.0	Añadir contenido con nuevo análisis

	Diseño y Pruebas II Acme-Software-Factory
	Analysis Report

3. Introducción

A continuación, detallaré la lista de requisitos abordados junto con las diversas opciones consideradas y la solución seleccionada. También, en ciertos casos, incluiré una imagen sobre el hilo del foro que detalla la decisión adoptada.



4. Contenido

- Requisito 002

No queda claro lo que se pide exactamente en “total time”. Se dio la siguiente respuesta:

Cadena: [Análisis] D02-Student#3 - 002

Publicación: RE: [Análisis] D02-Student#3 - 002

Autor:  RAFAEL CORCHUELO GIL 

Fecha de publicación: 1 de marzo de 2024 11:20

Fecha de edición: 1 de marzo de 2024 11:26

Estado: Publicado

Estimado Raúl:

Gracias por seguir nuestros consejos e intentar realizar un análisis de alternativas antes de plantear su pregunta.

Comenta Ud. en relación al requisito que define los cursos de formación ("training modules"). Por favor, no pierda de vista que los requisitos se enuncian de manera individual para que queden lo más claros y estructurados que sea posible, pero no se pueden entender de forma individual, sino en el contexto del correspondiente documento de elicitación de requisitos. En este caso concreto, estamos hablando de un objeto que parece un conglomerado, por lo que no podemos analizar el requisito que describe el todo ("whole") sin tener en cuenta el requisito que describe las partes ("parts", "components"). En este caso es el siguiente requisito.

[D02-S03-03] Each training module is made up of training sessions. The system must store the following data about them: a code (pattern "TS-[A-Z]{1,3}-[0-9]{3}", not blank, unique), a time period (at least one week ahead the training module creation moment, at least one week long), a location (not blank, shorter than 76 characters), an instructor (not blank, shorter than 76 characters), a mandatory contact email, and an optional link with further information.

Por otra parte, hemos ya revisado otros requisitos en los que nuestro cliente parece computar tiempos y costes como un número de horas entero. Así que ésta no será la excepción.

En relación con sus alternativas:

A1: habitualmente la medida de esfuerzo o tiempo son las horas. Los días no se usan porque un día tiene un número demasiado grande de horas y habitualmente una persona se dedica a varias tareas cada día. La cuestión es si lo implementa como "Integer" (en caso de que pueda tomar el valor "null") o como "int" (en caso de que sea obligatorio).

A2: "Duration" es una clase que no le hemos recomendado y por lo tanto, yo no la consideraría en ningún momento. Nuestra recomendación es usar siempre el tipo "java.util.Date" para representar momentos y en caso de necesitar una duración usar un número entero o doble según el caso. Por otra parte, todas las operaciones que involucren momentos de tiempo deben realizarse a través de un helper denominado "MomentHelper". El motivo es que vamos a trabajar tanto en el diseño (Lecciones L02 y L03) como en el testing (Lección L04). Para poder realizar testing necesitamos garantizar la reproducibilidad de las pruebas y eso requiere utilizar un reloj simulado durante desarrollo y pruebas y el reloj real en explotación. Java ofrece multitud de clases para representar conceptos relacionados con el tiempo * usando el reloj real * de la máquina. Eso dificultará hará las pruebas irreproducibles. Piense por ejemplo en una típica restricción del tipo "el período de ejecución de algo debe ser al menos una semana posterior al de creación de ese algo". Si trabajamos con un reloj real, llegará un momento en que los datos de prueba incumplan esa restricción porque el reloj real avanza inexorablemente a una velocidad de una hora por hora. La única solución es usar un reloj simulado que nos permita volver cada vez que lo necesitemos a un determinado momento en el pasado. Por este motivo, *les recomendamos muy, muy encarecidamente* que "jamás" usen clases relacionadas con el tiempo que no son las recomendadas. Si ignoran este consejo, seguramente se encontrarán problemas en la lección L04 y será demasiado tarde para hacer arreglos.

A3: Hemos comentado en otros mensajes las diversas alternativas cuando se implementa como double.

Comenta Ud. que debemos consultar al cliente, pero el cliente ya ha sido consultado en relación a requisitos similares y nos ha confirmado que mide esfuerzos y tiempos en horas enteras (salvo, evidentemente, que explícitamente indique lo contrario).



Saludos. RC

- Requisito 003

Sobre cómo gestionar el “time period” del que se nos habla se dio la siguiente respuesta:

Cadena: Ambigüedad en un requisito de información

Publicación: RE: Ambigüedad en un requisito de información

Autor:  RAFAEL CORCHUELO GIL 

Fecha de publicación: 22 de febrero de 2024 11:40

Estado: Publicado

Estimado Alfonso Luis:

Gracias por seguir los consejos que les proporcionamos a la hora de plantear sus cuestiones. Tan sólo le sugiero que en próximas comunicaciones identifique mejor el tema y el objeto de su consulta dado que eso ayudará más tarde a localizar los mensajes del foro. Por ejemplo, en el caso de las consultas sobre análisis, un buen título puede ser "[Análisis] D02-S01-005" o "[Análisis] D03-G-003"; son títulos compactos que hacen referencia directa al tema del mensaje (análisis de los requisitos) y el objeto (requisito entrega - grupo/individual - número).

Entrando en harina, en efecto, la palabra "duration" es confusa en el contexto de ese requisito. Por sí misma, la palabra "duration" hace referencia a una cantidad de tiempo; por ejemplo, 15 minutos, 4 horas, 5 días y 6 horas. Pero el requisito indica que la dicha "duration" tiene un momento de inicio. Realmente, el cliente se refiere a que los "objectives" tienen un período de tiempo durante el que deben ser conseguidos. Por lo tanto, el requisito en cuestión quedaría redactado de la siguiente forma:

"An objective allows an authenticated principal to define a goal or end towards which the actions or operations of a specific project are directed. The system must store the following data about them: an instantiation moment (in the past), a title (not blank, shorter than 76 characters), a description (not blank, shorter than 101 characters), a priority ("Low", "Medium", "High"), a status to indicate whether it is critical or not, "an execution period (must start at any moment after the instantiation moment and last for at least one hour)", and an optional link with further information."

En relación con sus alternativas, le comento:

A1: como Ud. mismo indica, no es una buena alternativa dado que un objeto de tipo "Date" permite almacenar un momento (timestamp con fecha y/u hora), no una duración.

A2: efectivamente, un número puede recoger perfectamente una duración de tiempo, pero no el momento de inicio al que hace referencia el requisito. Además, sería necesario aclarar si ese número hace referencia a minutos, a horas y minutos, hora y fracción, o días, por poner unos ejemplos.


A3: como Ud. también indica, sería muy buena alternativa dado que nos vamos a encontrar con períodos de tiempo en muchos requisitos y así tendríamos un datatype reutilizable para modelar el concepto de "período". Pero siga leyendo, por favor.

Aunque la A3 es globalmente la mejor alternativa, "NO PUEDO RECOMENDARSELA" en nuestro contexto. Le explico el porqué. Como hemos comentado en la sesión S01 de la L02, implementar entidades o forms es algo bastante sencillo y directo con la ayuda del framework; por desgracia, implementar un datatype no es tan simple. (Explicaremos como hacerlo en una sesión posterior de temas avanzados, pero aún así no va a ser sencillo) Eso significa que si abordamos ahora la implementación de ese datatype, nuestro proyecto podría sufrir retrasos debido a los problemas que encontremos durante el desarrollo. Y dichos retrasos no serán fáciles de explicar al cliente dado que el proyecto que ha contratado no tiene ninguna componente de I+D o de innovación que pueda justificarlo. Es mucho mejor acumular un poco de experiencia y abordar la implementación de dicho datatype cuando terminemos con el proyecto y podamos dedicar un tiempo a actividades de I+D o innovación, pero sin la presión de tener que cumplir con los plazos de entrega de un cliente que nada tiene que ver con la mejora de nuestras herramientas de desarrollo.

Por lo tanto, mi recomendación es que implemente Ud. el "execution period" al que hace referencia la nueva redacción del requisito como dos atributos de tipo "Date" más un conjunto de restricciones apropiadas: ninguno de los dos puede ser nulo, el momento de inicio debe ser posterior al de creación, el de inicio debe ser anterior al de finalización y, una vez consultado con el cliente, nos comenta que no tiene mucho sentido una duración inferior a una hora.

Es una solución sencilla y directa que tiene como único contra que no es reutilizable, como sí lo sería un datatype. Pero no es el momento adecuado ahora para implementarlo. Lo tenemos que apuntar en el roadmap y dejarlo para más adelante.

Saludos. RC

	Diseño y Pruebas II Acme-Software-Factory
	Analysis Report

- Rangos de atributos no especificados

Esto se ha corregido en diferentes entidades gracias al feedback que recibió un compañero

RAFAEL CORCHUELO GIL 🟢

hace 1 mes

RE: [Análisis] Rangos de atributos no especificados

Estimado Rafael:

Gracias por seguir los consejos que les proporcionamos a la hora de plantear sus preguntas.

Comenta Ud. sobre un problema que es tremendamente habitual en los documentos de elicitación de requisitos: los límites de los tipos de datos.

Es muy habitual que cuando algún atributo de una entidad tiene un límite claro dependiente del negocio de su cliente dicho límite se haga constar explícitamente. Por ejemplo: "el período de ejecución de algo debe comenzar al menos una semana después de la creación o la mínima duración de dicho período debe ser de un día" o "no se permiten más cinco quejas por cliente en un mismo día".

Pero existen otros muchos límites en los que habitualmente nadie piensa hasta que nos enfrentamos (de forma metódica) a crear datos de prueba. Por ejemplo: no es habitual encontrar escritos los límites para las cantidades de dinero (salvo quizá el inferior), para los momentos (salvo las restricciones habituales con respecto a los momentos de creación), o longitudes de emails o URLs.

Tampoco es habitual que se hagan explícitos los rangos de algunos tipos de datos que se entienden son lo suficientemente conocidos como para no generar dudas. Por ejemplo, si nos comentan que algo es un porcentaje queda claro que es un número en el intervalo $[0.00 - 1.00]$ (quizá $[0.00 - 100.00]$; o si nos comentan que algo es un ratio entonces queda totalmente claro que es un número en el intervalo $[0.00 - 1.00]$; la única duda podría ser en relación al número de decimales que debemos admitir.

En estos casos, lo mejor es hacer una propuesta razonable al cliente, que habitualmente no tendrá mucho inconveniente dado que esto es un detalle de implementación que poco tiene que ver con su negocio. Simplemente tenemos que confirmar que los rangos elegidos por el equipo de desarrollo son suficientemente grandes, nada más.

Por lo tanto, la alternativa más adecuada es la A1. La alternativa A2 no es muy buena porque al dejar los intervalos abiertos no podemos hacer pruebas sistemáticas. De forma general, le propongo lo siguiente:


- C cantidades de dinero: de -1.000.000,00 XXX a +1.000.000,00 XXX, donde XXX es una unidad monetaria aceptada por el sistema.
- Momentos: de 01/01/2000 00:00 a 31/12/2200 23:59.
- Emails o URLs: de 0 a 255 caracteres.

Por desgracia, no tenemos ninguna forma sencilla de fijar estos rangos mediante anotaciones estándar, por lo que tenemos que implementarlos como restricciones custom en los correspondientes servicios.

Como ingenieros de software debemos estar siempre atentos a las ideas que nos permitan mejorar nuestras herramientas de trabajo; cada mejora en dichas herramientas debe ir orientada a reducir los costes de desarrollo, decrementar las posibilidades de cometer errores y, en definitiva, mejorar la calidad de nuestros productos.


En este caso particular, está claro que sería interesante investigar de qué forma podríamos introducir esas restricciones implícitas dentro del framework. Pero no durante el desarrollo de un proyecto. Durante el desarrollo de un proyecto tenemos que darnos cuentas de los problemas; es al final de los mismos cuando debemos recapitular y tomar decisiones sobre cómo mejorar nuestro framework de desarrollo para el siguiente proyecto.

Saludos. RC

	Diseño y Pruebas II Acme-Software-Factory
	Analysis Report

5. Conclusiones

Para los requisitos que se han desarrollado, tras hacer un correcto análisis, se ha tomado generalmente la decisión correcta gracias a las respuestas del cliente.

	Diseño y Pruebas II Acme-Software-Factory
	Analysis Report

6. Bibliografía

Intencionadamente en blanco