Práctica 2. Introducción al diseño software

1. Introducción

Comenzamos la segunda práctica de la asignatura con el análisis, desde el punto de vista del diseño software, de un problema práctico. Este problema lo iremos resolviendo poco a poco en sucesivas prácticas.

El problema es el siguiente. Supongamos que, desde una empresa, se te solicita que desarrolles un sistema software para dar soporte a una completa infraestructura de sensores. Para ello, tendrás que diseñar el sistema que da solución al problema y, posteriormente, implementarlo siguiendo el diseño que hayas elegido y empleando las técnicas de la programación orientada a objetos (POO) en C++ que veremos durante el curso.

El desarrollo de este diseño te ayudará a acostumbrarte a los tipos de problemas que se dan en el mundo laboral, así como saber dar respuesta a ellos. Empezaremos el proceso de diseño especificando los requerimientos del sistema; esto es, cuál es el propósito general del sistema multisensorial y qué se supone que debe hacer este.

2. Especificación de requerimientos

Una empresa de Fuenlabrada, denominada *Julio Veganos e Hijos*, dedicada al desarrollo autosostenible, pretende implantar un sistema multisensorial en un invernadero, de forma que pueda tener controladas todas las variables climatológicas que se dan en el mismo, tanto

presencial como remotamente. Además, este sistema debe incorporar los mecanismos necesarios para controlar la seguridad de la empresa: alarma, detección de puertas/ventanas abiertas, etc.

Cualquier empleado de la empresa podrá acceder —mediante autenticación— al interfaz del sistema para acceder a todos los datos de los componentes descritos. Esto supone que cada empleado solo puede tener una cuenta en el sistema.

La interfaz de usuario contiene los siguientes componentes hardware:

- Una pantalla que muestre todos los datos.
- Un teclado que permite la entrada de datos al usuario (lo primero de todo, su datos de *login*).
- Un micrófono que permite dar instrucciones y que sean escuchadas desde las distintas estaciones remotas del sistema.
- Un interruptor o similar que permita activar/desactivar cómodamente la alarma de la empresa.

Esta empresa te pide que desarrolles el software que permita leer los datos de los distintos sensores y cámaras instalados en distintas ubicaciones de su plantación, y los vierta de la forma más *amigable* posible mediante un interfaz gráfico. Este software debe encapsular la funcionalidad de los distintos componentes hardware dentro de los correspondientes componentes software, sin preocuparse por cómo estos dispositivos realizan sus tareas.

Esta plataforma no está desarrollada aún, por lo que en lugar de desarrollar el software para que sea ejecutado en el sistema final multisensorial, deberás desarrollar una primera versión del mismo para que —simplemente— se ejecute en el ordenador. Piensa que la versión definitiva deberá poder ejecutarse en la placa (e.g. Arduino, Raspberry, etc.) que se use en las estaciones remotas, pero esa no va a ser nuestra tarea. Esta versión *beta* utilizará el monitor del PC para simular el interfaz del sistema, el teclado como entrada de datos, y distintos algoritmos para simular los datos que verterían los distintos dispositivos sensoriales.

Una sesión con el interfaz del sistema consiste en la autenticación de un usuario (un empleado de la empresa); esto es, proporcionar la identidad del usuario con base en un número de identificación de empleado, seguida del *dashboard* o parrilla en la que se muestran de forma gráfica los datos que vierten los sensores conectados al sistema. Para autenticar un

usuario y que este pueda acceder al sistema, el programa debe interactuar con la base de datos de información sobre los empleados registrados en la empresa.

Nota: una base de datos es una colección organizada de datos almacenados en un ordenador/servidor.

Para cada empleado, la base de datos almacena un número de usuario, un NIF y un *timestamp* que indica la fecha del último acceso de este usuario al sistema.

Nota: para simplificar, vamos a asumir que la empresa planea construir sólo un sistema multisensorial, en su invernadero de Fuenlabrada, por lo que no necesitamos preocuparnos por que varias delegaciones accedan a esta base de datos al mismo tiempo. Lo que es más, vamos a suponer que la empresa no va a realizar modificaciones en la información que hay en la base de datos mientras un usuario accede al sistema. Además, cualquier sistema comercial como este se topa con cuestiones de seguridad con una complejidad razonable, las cuales van más allá del alcance de los objetivos de esta asignatura. No obstante, para simplificar nuestro ejemplo vamos a suponer que la empresa confía en este sistema para que acceda a la información de la base de datos y la manipule sin necesidad de medidas de seguridad considerables.

Al acercarse al puesto de acceso al sistema, el usuario deberá experimentar la siguiente secuencia de eventos:

1. La pantalla muestra un mensaje de bienvenida y pide al usuario que introduzca un número de empleado.

2. El usuario introduce un número de empleado de cinco dígitos, mediante el uso del teclado numérico.

3. En la pantalla aparece un mensaje, en el que se pide al usuario que introduzca su NIF (número de identificación fiscal) asociado con el número de empleado especificado.

4. El usuario introduce un NIF de ocho dígitos mediante el teclado numérico.

5. Si el usuario introduce un número de empleado válido y el NIF correcto para ese usuario, la pantalla muestra el *dashboard*. Si el usuario introduce un número de empleado inválido o un NIF incorrecto, la pantalla muestra un mensaje apropiado y después el sistema regresa al paso 1 para reiniciar el proceso de autenticación.

Una vez que el sistema autentica al usuario, el *dashboard* debe contener información actualizada de los valores vertidos por los sensores conectados al sistema, así como las imágenes

en tiempo real vertidas por las distintas cámaras de seguridad distribuidas en la finca de la empresa. También debe haber una opción para poder salir del sistema en cualquier momento.

Las opciones que muestra la pantalla principal dependerá del número de sensores que estén conectados. Se va a considerar que debe haber un mínimo de cuatro sensores conectados y dos cámaras. Pero hay que tener en cuenta que, en cualquier momento de la vida del sistema, este número puede aumentar, y habría que dar cabida de forma dinámica a los nuevos componentes. Así, las opciones que habrá por defecto en el *dashboard* son:

1. Temperatura.
2. Humedad.
3. Calidad del aire.
4. Nivel de iluminación.
5. Cámara RGB.
6. Cámara térmica.
7. Salir.

Las opciones deben estar numeradas para permitir al usuario acceder a la información de cada componente y visualizarla de forma más detallada en pantalla completa. Si el usuario introduce una opción inválida, la pantalla muestra un mensaje de error y vuelve a mostrar el menú principal. Si introduce la opción de *Salir*, se cierra el sistema y se vuelve a la pantalla de bienvenida. Por último, si el usuario introduce cualquier opción válida, se debe cambiar a la pantalla de información detallada, donde se muestre el gráfico con más detalle así como los datos registrados por ese sensor durante la última hora; además de una opción para *Volver al menú*.

3. Análisis del sistema

En la declaración anterior se presentó un ejemplo simplificado de una especificación de requerimientos. Por lo general, dicho documento es el resultado de un proceso detallado de recopilación de requerimientos, el cual podría incluir entrevistas con usuarios potenciales del sistema y especialistas en campos relacionados con el mismo. Por ejemplo, un analista de sistemas que se contrate para preparar una especificación de requerimientos para software de sistemas multisensoriales o videovigilancia (como el sistema que describimos aquí) podría

entrevistar a expertos en seguridad, domótica o robótica para obtener una mejor comprensión de qué es lo que debe hacer el software. El analista utilizaría la información recopilada para compilar una lista de requerimientos del sistema y, con ello, guiar a los diseñadores de sistemas en el proceso de diseño del mismo.

El proceso de recopilación de requerimientos es una tarea clave de la primera etapa del ciclo de vida del software. El ciclo de vida del software especifica las etapas a través de las cuales el software evoluciona, desde el tiempo en que fue concebido hasta el tiempo en que se retira de su uso. Por lo general, estas etapas incluyen: análisis, diseño, implementación, prueba y depuración, despliegue, mantenimiento y retirada. Existen varios modelos de ciclo de vida del software, cada uno con sus propias preferencias y especificaciones respecto a cuándo y con qué frecuencia deben llevar a cabo los ingenieros de software las diversas etapas. Los modelos de cascada realizan cada etapa de forma sucesiva, mientras que los modelos iterativos pueden repetir una o más etapas varias veces a lo largo del ciclo de vida de un producto.

La etapa de análisis del ciclo de vida del software se enfoca en definir el problema a resolver. Al diseñar cualquier sistema, uno debe resolver el problema de la manera correcta, pero de igual manera uno debe resolver el problema correcto. Los analistas de sistemas recolectan los requerimientos que indican el problema específico a resolver. Nuestra especificación de requerimientos describe nuestro sistema con el suficiente detalle como para tú no necesites pasar por una etapa de análisis exhaustiva; esto ya se hizo previamente.

Para capturar lo que debe hacer un sistema propuesto, los desarrolladores emplean a menudo una técnica conocida como *modelado de casos de uso*. Este proceso identifica los casos de uso del sistema, cada uno de los cuales representa una capacidad distinta que el sistema provee a sus clientes. Por ejemplo, es común que un sistema multisensorial tenga varios casos de uso, como *Ver temperatura*, *Ver humedad*, *Ver calidad del aire*, etc. El sistema simplificado que construiremos requiere seis casos de uso (Figura 1).

Cada uno de los casos de uso describe un escenario común en el cual el usuario utiliza el sistema. Ya vimos las descripciones de los casos de uso del sistema en la especificación de requerimientos; las listas de pasos requeridos para realizar cada acción describen en realidad los seis casos de uso de nuestro sistema.

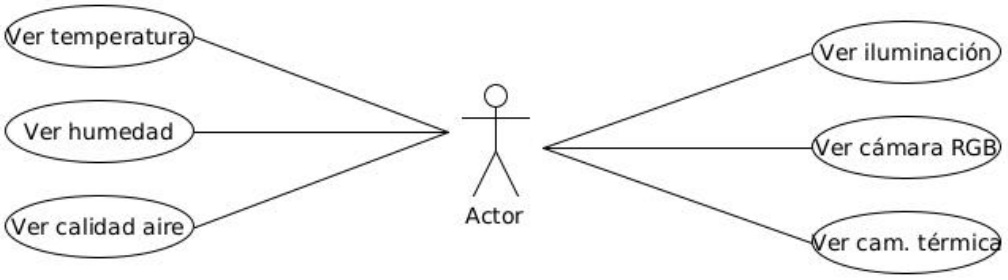

Figura 1: Diagrama de casos de uso para el sistema multisensorial, desde la perspectiva del usuario

4. Diagramas de casos de uso

Ahora presentaremos el primero de varios diagramas de UML para nuestro ejemplo práctico. Vamos a crear un diagrama de casos de uso para modelar las interacciones entre los clientes de un sistema (en este ejemplo práctico, los empleados de la empresa) y el sistema. El objetivo es mostrar los tipos de interacciones que tienen los usuarios con un sistema sin proveer los detalles; éstos se mostrarán en otros diagramas de UML (los cuales presentaremos a lo largo del ejemplo práctico). A menudo, los diagramas de casos de uso se acompañan de texto informal que describe los casos de uso con más detalle, como el texto que aparece en la especificación de requerimientos. Los diagramas de casos de uso se producen durante la etapa de análisis del ciclo de vida del software. En sistemas más grandes, los diagramas de casos de uso son herramientas simples pero indispensables que ayudan a los diseñadores de sistemas a enfocarse en satisfacer las necesidades de los usuarios.

La Figura 1 muestra el diagrama de casos de uso para nuestro sistema. La figura humana representa a un actor, el cual define los roles que desempeña una entidad externa (como una persona u otro sistema) cuando interactúa con el sistema. Para nuestro programa, el actor es un usuario que puede ver la temperatura, la humedad, la calidad del aire, etc. El usuario no es una persona real, sino que constituye los roles que puede desempeñar una persona real (al desempeñar el papel de un usuario) mientras interactúa con el sistema. Hay que tener en cuenta que un diagrama de casos de uso puede incluir varios actores. Por ejemplo, el diagrama de casos de uso para un sistema como el planteado para una empresa real podría incluir también un actor llamado *Administrador*, que pueda —entre otras cosas— modificar los datos del perfil de cada *Usuario*.

Para identificar al actor en nuestro sistema, debemos examinar la especificación de requerimientos, la cual dice: *los usuarios del sistema deben poder ver la temperatura, la humedad,*

etc.. Por lo tanto, el actor en cada uno de estos seis casos de uso es el *Usuario* que interactúa con el sistema. Una entidad externa (una persona real) desempeña el papel del usuario para realizar tales acciones. La Figura 1 muestra un actor, cuyo nombre (*Usuario*) aparece debajo del actor en el diagrama. UML modela cada caso de uso como un óvalo conectado a un actor con una línea sólida.

Los ingenieros de software (más específicamente, los diseñadores de sistemas) deben analizar la especificación de requerimientos o un conjunto de casos de uso, y diseñar el sistema antes de que los programadores lo implementen. Durante la etapa de análisis, los analistas de sistemas se enfocan en comprender la especificación de requerimientos para producir una especificación de alto nivel que describa qué es lo que el sistema debe hacer. El resultado de la etapa de diseño (una especificación de diseño) debe especificar claramente cómo debe construirse el sistema para satisfacer estos requerimientos. En las siguientes prácticas llevaremos a cabo los pasos de un proceso simple de diseño orientado a objetos (DOO) con el sistema multisensorial planteado, para producir una especificación de diseño que contenga una colección de diagramas de UML y texto de apoyo. Recuerda que UML está diseñado para utilizarse con cualquier proceso de DOO. Existen muchos de estos procesos de desarrollo software, de los cuales el más conocido es *Rational Unified Process (RUP)*, desarrollado por Rational Software Corporation (ahora una división de IBM). RUP es un proceso robusto guiado por los casos de uso para diseñar aplicaciones a nivel industrial. Para este ejemplo práctico, presentaremos nuestro propio proceso de diseño simplificado.

5. Diseño del sistema multisensorial

Ahora comenzaremos la etapa de diseño de nuestro sistema multisensorial. Un sistema es un conjunto de componentes que interactúan para resolver un problema. Por ejemplo, para realizar sus tareas designadas, nuestro sistema tiene una interfaz de usuario, contiene software para ejecutar diversas acciones e interactúa con una base de datos de información de los empleados de la empresa. La estructura del sistema describe los objetos del sistema y sus interrelaciones. El comportamiento del sistema describe la manera en que cambia el sistema a medida que sus objetos interactúan entre sí. Todo sistema tiene tanto estructura como comportamiento; los diseñadores deben especificar ambos. Existen varios tipos distintos de estructuras y comportamientos de un sistema. Por ejemplo, las interacciones entre los objetos en el sistema son distintas a las interacciones entre el usuario y el sistema, pero aun así ambas constituyen una porción del comportamiento del sistema.

El estándar UML 2 especifica 13 tipos de diagramas para documentar los modelos de

sistemas. Cada tipo de diagrama modela una característica distinta de la estructura o del comportamiento de un sistema: seis diagramas se relacionan con la estructura del sistema; los siete restantes se relacionan con su comportamiento. A continuación se describen sólo los seis tipos de diagramas que utilizaremos en nuestro ejemplo práctico, uno de los cuales (el diagrama de clases) modela la estructura del sistema, mientras que los otros cinco modelan el comportamiento:

- Los **diagramas de casos de uso**, como el de la Figura 1, modelan las interacciones entre un sistema y sus entidades externas (actores) en términos de casos de uso (capacidades del sistema, como *Ver la temperatura*, *Ver la humedad*, etc.).

- Los **diagramas de clases** modelan las clases —o bloques de construcción— que se utilizan en un sistema. Cada sustantivo u objeto que se describe en la especificación de requerimientos es candidato para ser una clase en el sistema (por ejemplo, *CuentaDeEmpleado*, *Teclado*). Los diagramas de clases nos ayudan a especificar las relaciones estructurales entre las partes del sistema. Por ejemplo, el diagrama de clases del sistema multisensorial especificará que este está compuesto físicamente de una pantalla y un teclado.

- Los **diagramas de máquina de estado**, modelan las formas en que un objeto cambia de estado. El estado de un objeto se indica mediante los valores de todos los atributos del objeto en un momento dado. Cuando un objeto cambia de estado, puede comportarse de manera distinta en el sistema. Por ejemplo, después de validar el NIF de un usuario, el sistema cambia del estado *usuario no autenticado* al estado *usuario autenticado*, punto en el cual el sistema permite al usuario realizar las operaciones de monitorización que desee.

- Los **diagramas de actividad** modelan la actividad de un objeto: el flujo de trabajo (secuencia de eventos) del objeto durante la ejecución del programa. Un diagrama de actividad modela las acciones que realiza el objeto y especifica el orden en el cual desempeña estas acciones. Por ejemplo, un diagrama de actividad muestra que el sistema debe obtener los datos del usuario (de la base de datos de información de las cuentas de los trabajadores de la empresa) antes de que la pantalla pueda mostrar las acciones disponibles de monitorización al usuario.

- Los **diagramas de comunicación** (llamados diagramas de colaboración en versiones anteriores de UML) modelan las interacciones entre los objetos en un sistema, con un énfasis acerca de qué interacciones ocurren. Por ejemplo, el sistema debe comunicarse con la base de datos de información de las cuentas de los trabajadores de la empresa para obtener los datos de estos.

- Los **diagramas de secuencia** modelan también las interacciones entre los objetos en un sistema, pero a diferencia de los diagramas de comunicación, enfatizan en cuándo ocurren las interacciones. Por ejemplo, la pantalla pide al usuario que seleccione la opción de monitorización que desea visualizar antes de efectuar tal operación.

Ejercicio

Una vez entendida la teoría de cómo funciona un desarrollo software, implementa un primer borrador del diagrama de clases que modelará la estructura del sistema de monitorización planteado. En esta primera práctica, deberás realizar este borrador a mano y, una vez lo tengas terminado, hazle una foto y súbelo al repositorio de esta práctica.