**UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE**

**Resumen Pitfalls**

**Integrantes**:

Achig Toapanta Steven Jossue

Asmal Haro, Kevin David

Cadena Diaz, Jeremy Joel

Caiza Tacuri, Alisson Lisbeth

Cajas Recalde, Karla Lizbeth

Campoverde Encarnacion, Carlos Danny

**NRC:** 3682

**Fecha de entrega:** 06/07/21

Los pitfalls son errores que pueden ser implementados en el modelado del diagrama de casos de uso, o en la programación en sí, estas “trampas” pueden darse por descuido, desconocimiento, o simplemente pereza del programador.

Si hablamos de pitfalls en el diagrama de casos de uso podemos enlistar los siguientes:

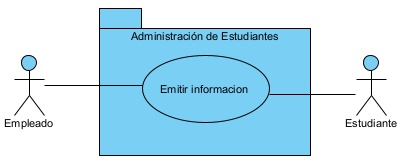
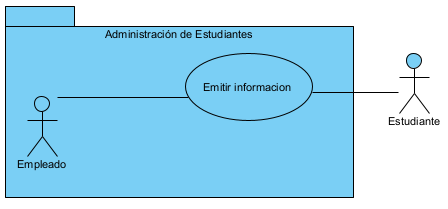
1. Proceso de Modelado: Es un error común querer modelar todos los procesos o flujos de trabajo en un solo diagrama, pero esto es incorrecto, cada caso de uso debe ser independiente el uno del otro, la funcionalidad de uno de los casos de uso no es la funcionalidad de otro caso de uso.

Podemos observar en el siguiente gráfico que a pesar de que los casos de uso están conectados cronológicamente correctamente, es un error representarlos en un solo diagrama de casos de uso.

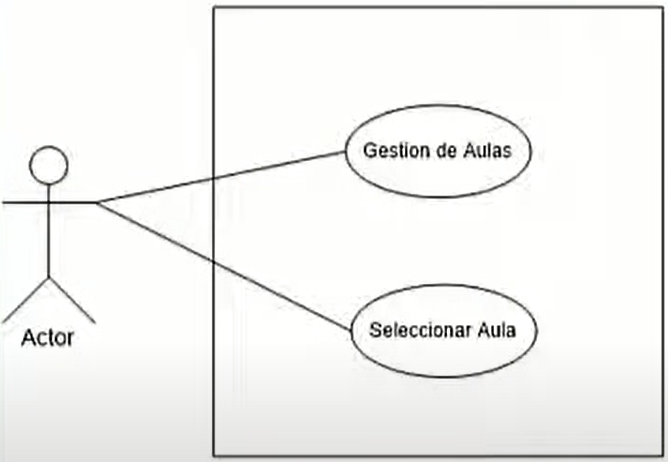


1. Ajuste de los límites del sistema: Hay que tener muy en cuenta que los actores en un diagrama siempre irán fuera del sistema.

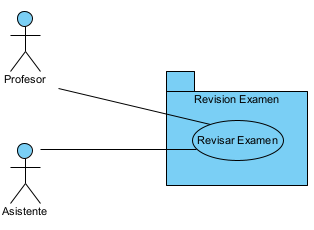
Aquí podemos ver un ejemplo si el empleado es una parte del sistema no debe modelarse como actor, más bien alguna otra entidad fuera del sistema sería el actor. Si el empleado no es parte del sistema pero se necesita para la ejecución del caso de uso, entonces se modela como actor fuera de los límites del sistema.

1. Mezcla de niveles de Abstracción: Evitar representar los casos de uso de “Alto Nivel” con los de orientación técnica en el mismo diagrama.

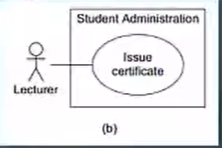
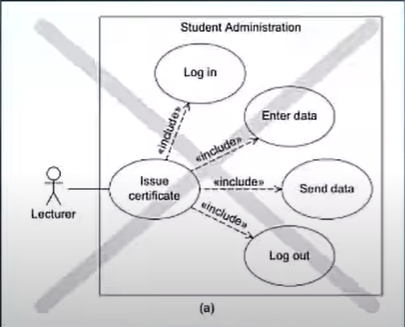


1. Asociaciones Incorrectas: Se entiende que que al asociar dos o más actores no decimos que para que el caso de uso sea ejecutado se necesita uno u otro actor, lo que decimos es que se necesitan esos actores para su ejecución.



Se lo realiza de forma más rápida y de forma limpia es una solución que se debe ir puliendo desde un principio pero también es importante aceptar que la primera iteración será un borrador para mejorar o a su vez ir corrigiendo pequeños desperfectos.

* **En la identificación** de casos de uso tenemos que asegurarnos de que esté al mismo nivel de abstracción.
* **Descomposición funcional**: Puede ejecutarse de forma independiente pero debe hacer referencia a su descripción



* No debemos olvidar que la programación se trata de personas. Claro que la máquina ejecuta el código, pero la programación está resolviendo problemas para las personas, traduciendo sus requisitos en código que será leído, mantenido y modificado por nosotros.
* Saltarse los fundamentos : Termina aprendiendo sobre el estilo de paso de continuación pero aún no puede hacer uniones sin un mapeo relacional de objetos.
* Perfeccionismo: Dejar que lo perfecto sea enemigo lo suficientemente. Nunca enviando nada, quedando atascado en la fase de "toques finales". Pero el equilibrio es importante, así que no envíe software terriblemente roto por el bien del envío.

pitfalls de programación

Los errores de programación afectan el [código del software](https://tecnologiandroid.com/que-es-un-error-de-software/). Esto se implementa incorrectamente, por lo que la aplicación no funciona como se esperaba.

El término error de programación implica que el código del software se ha implementado incorrectamente. En algún momento los desarrolladores ignoraron uno de los requisitos del lenguaje en el que escribieron el código del programa.

tipos de errores de programación

* Error léxico

Es un error simple pero muchas veces difícil de encontrar, se cometió un error tipográfico en algún lugar del código y a menudo suele ser causado por mayúsculas o minúsculas.

* Error sintáctico:

la «gramática» del idioma es incorrecta. En lenguaje sencillo, esto significa: Un error sintáctico describe una estructura de oración incorrecta. Positivo: el compilador y el intérprete pueden reconocer esto y rechazar inmediatamente el código en la ejecución de prueba para llamar la atención sobre el error.

* Error semántico

Este error ocurre cuando se utilizan variables que no estaban previamente definidas o declaradas incorrectamente por lo cual el código pierde todo sentido.

* Errores de tiempo de ejecución:

Estos problemas ocurren durante la ejecución del programa y no pueden ser reconocidos por los compiladores. Por lo general, conducen a la finalización de la aplicación. Los errores de tiempo de ejecución ocurren con las llamadas «operaciones imposibles», por ejemplo, al intentar dividir por cero. Se pueden encontrar usando el depurador.

* Error lógico:

Las operaciones aritméticas se intercambian, lo que significa que la cadena de mando no tiene sentido. Con el cálculo de porcentaje, por ejemplo, el programa solicita una suma en lugar de una multiplicación. El problema con esto es que ni el depurador ni el compilador pueden detectar un error lógico. Esto solo es posible mediante una prueba de programa detallada.

**Errores más comunes de Java**

**Fugas de memoria:**

Uno de los beneficios principales de Java es el recolector de basura de java, que administra la memoria de los objetos en el montón. Siempre que un objeto es inalcanzable, se libera automáticamente.

Sin embargo, un error común tanto para los programadores nuevos como para los experimentados es evitar que se libere la memoria, al permitir que se pueda acceder a los objetos que ya no están en uso. Esta puede ser una gran desventaja para un proyecto, ya que las pérdidas de memoria bloquean los recursos y degradan el rendimiento de la aplicación.

**Los casos comunes son:**

Campos estáticos Declarar un campo estático y olvidar establecerlo en nulo después de que sus datos ya no sean necesarios

Secuencias no cerradas La máquina virtual Java asigna memoria para cada conexión abierta. Olvidar cerrar la conexión consume memoria. Dichas conexiones pueden ser: Flujos de entrada , Conexiones de bases de datos , Sesiones y otras.

El método finalize () Cuando anulamos el método finalize () , el objeto de esa clase ya no se recolecta directamente como basura. El recolector de basura está esperando la finalización, lo que ocurre en un momento posterior.

**PREGUNTAS:**

1. ¿Si llamo a una variable a mi programa que no estaba definida previamente que tipo de error aparecerá?

* Error lógico
* ~~Error semántico~~
* Error de tiempo de ejecución
* Error léxico

1. Complete: El término error de programación implica que el código del software se ha implementado …………………

R: incorrectamente.

1. Complete: En un diagrama de …… de ……... los actores siempre deben ir …….. del sistema.

R: Casos, Uso, Fuera

1. Un error común es querer modelar todos los procesos o flujos de trabajo en un solo diagrama, este error corresponde a un error de:

* ~~Proceso de Modelado.~~
* Ajuste de los límites del Sistema.
* Mezcla de niveles de abstracción.

1. Al asociar dos o más actores en un diagrama de casos de uso, para que el caso de uso sea ejecutado ¿se necesita uno u otro actor?

* Verdadero
* ~~Falso~~
* No responder

1. ¿Cuál es el error más común de java?

r: Un error común tanto para los programadores nuevos como para los experimentados es evitar que se libere la memoria, al permitir que se pueda acceder a los objetos que ya no están en uso

1. Las pérdidas de………………… bloquean los recursos y degradan el……………………………………………….

r: memoria, rendimiento de la aplicación

1. Complete: En la identificación de casos de uso tenemos que asegurarnos de que esté al mismo nivel de ………………….

R: Abstracción.

1. Complete: Es importante aceptar que la primera iteración será un ………….. para mejorar o a su vez ir corrigiendo pequeños desperfectos.

R:Borrador.

1. El error más común en la declaración de arreglos es:

* OneLetter = MyString.charAt(0); creer que la primera posición es el 0
* ~~OneLetter = MyString.charAt(1); creer que la primera posición es el 1~~
* OneLetter = MyString.charAt(); creer que la primera posición no se inicializa

Bibliografia

Stuber, C. (22 de Marzo de 2021). *Tecnologia Android*. Recuperado el 06 de Julio de 2021, de <https://tecnologiandroid.com/que-es-un-error-de-programacion/>

Cangiano, A. (2017, 9 noviembre). *25 Pitfalls When Learning to Program*. Programming Zen. <https://programmingzen.com/pitfalls-when-learning-to-program/>

Becerra, M. Gloaria, V. Govea, J Limas, J Torres, V. (03 de Julio de 2016). *Pitfalls Casos de Uso*. Recuperado el 06 de Julio de 2021, de <https://www.youtube.com/watch?v=RJrkQewL1ZE&ab_channel=MartinBecerra>

5 Most Common Java Pitfalls - GeeksforGeeks. (2021). Retrieved 6 July 2021, from <https://www.geeksforgeeks.org/5-most-common-java-pitfalls/>