**Caso de Estudio**

**A7-E Avionics System**



**Realizado por:**

|  |  |  |
| --- | --- | --- |
| **Persona** | **Rol** | **Código Uniandes** |
| Carlos Ernesto González Vargas | Líder del Grupo | 200819123 |
| Sandra Milena Gómez Ríos | Líder de Planeación | 201110951 |
| Andrés Mauricio Erazo Benavides | Líder de Soporte | 201110949 |
| David Pérez Chibuque | Líder de Calidad | 201117818 |
| Willian Alejandro Idrobo Luna | Líder de Desarrollo | 201110544 |
| Erik Fernando Arcos Franco | Líder de Desarrollo | 201110856 |

**Control de versiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Autor** | **Descripción del Cambio** |
| 1.00 | 5 de Junio | Ingenium | Creación del documento |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

 

**Caso de Estudio**

**A7-E Avionics System**

1. **¿Cuáles propiedades de la arquitectura de un sistema se pueden ver en una estructura de descomposición de módulos (Module Descomposition)? ¿Cuales propiedades se exponen en el caso de estudio?**

En una estructura de descomposición por módulos las únicas propiedades que son visibles son aquellas que se han definido como públicas y que se presentan al exterior del módulo a través de las interfaces que expone el mismo y que pueden ser modificadas por medio de los procedimientos que también son expuestos por el modulo. El resto de las propiedades son exclusivamente internas al módulo y no pueden ser accedidas y/o modificadas por ningún agente externo al módulo en sí.

Para el caso de estudio, las propiedades expuestas son aquellas que exponen cada uno de los sub-módulos de tercer nivel que hacen parte de los 3 módulos principales (Hardware Hiding, Behavior Hiding, Software Decision), puesto que la información llega del mundo exterior a través de los sub-módulos del módulo Device Interface Module, esta información es luego suministrada a los correspondientes sub-módulos del módulo Function Driving Module, el cual se encarga de realizar las operaciones y cálculos necesarios con los datos suministrados a través del sub-módulos Data Banker con los tipos de dato proporcionados por el sub-módulos Application Data Type Module, soportados a través de las funcionalidades que ofrecen los otros sub-módulos del módulo Software Decision Module, y todo esto apoyado en las interacciones que provee el módulo Extended Computer Module para acceder al procesador.

1. **¿Qué ventajas conlleva dividir una arquitectura en módulos que exponen una interfaz pública y esconden la implementación de esa interfaz? ¿Qué desventajas tiene?**

Una de las ventajas de exponer una interfaz publica es que toda la interacción hacia el modulo se hace a través de esa interfaz, esto provee una forma segura y consistente de manejar los datos, las estructuras de datos y los procedimientos al interior del módulo, evitando que los mismos sean modificados y/o llamados por un procedimiento externo que no debería tener acceso a esta información; de esta manera la información al interior del módulo permanece en un estado consistente.

Otra ventaja de esconder la implementación del módulo es que de esta manera también se mantienen privadas las interacciones que tiene el modulo entre sí y con otros módulos, proveyendo únicamente la información que es relevante para el ente que la está solicitando sin que este ente tenga información adicional de cómo funciona el modulo para sí mismo, evitando que este ente pueda inducir procedimientos o datos que si bien están permitidos dentro del módulo podrían conllevar a una falla del mismo.

Quizás una desventaja es, que al tener un número limitado de puntos de acceso al módulo, suministrados por las interfaces, la interacción con estos módulos es más limitada, lo que puede incurrir en el uso de interfaces de otros módulos (por ejemplo, un módulo de utilidades), y de esta manera castigar un poco el desempeño de la aplicación.

1. **¿Qué ventajas conlleva un bajo acoplamiento entre los módulos de una arquitectura?**

Entre las ventajas de tener bajo acoplamiento entre los módulos de una arquitectura se encuentra el reducir la interdependencia entre los distintos módulos de la misma, de esta manera se reduce el impacto que pueda causar en la aplicación el cambio de un módulo en particular; esto se consigue puesto que el único acceso a un módulo es a través de su interfaz, si un módulo es modificado al interior expondría la misma interfaz (en la mayoría de los casos a no ser que se trate de un cambio mayor), lo cual no tendría repercusión alguna en los otros módulos que interactúan con esa interfaz, ya que estos no notarían ningún cambio en la forma como se conectan entre sí. Además esta opción también favorece la reutilización, ya que como los módulos no se encuentran fuertemente ligados entre sí pueden ser utilizados por cualquier modulo que requiera de sus operaciones en proyectos y sistemas diferentes.

1. **Para el caso de estudio, ¿Por qué se dividió la estructura en módulos responsables del “Hardware”, “Software Decisions” y “Visible Behavior”? ¿Porque no 2 módulos? ¿Porque no otros módulos?**

El módulo **Hardware** era imprescindible puesto que se necesitaba un módulo que conectara la información del mundo real, desplegada a través de sensores, hacia el interior de la aplicación. También era imprescindible para que se realizara la comunicación con el procesador de la aeronave, puesto que no existía un compilador y toda debía ser expresada en leguaje de máquina.

A pesar de que los módulos de **Software Decision** y **Visible Behavior** se encargan de la manipulación de la información suministrada por el módulo de hardware, se construyeron separadamente con el fin de separar los requerimientos, los cuales son manejados por el modulo **Visible Behavior**, de las funciones de cálculo, que son manejadas por el modulo **Software Decision**. De esta manera un cambio en los requerimientos, es decir, el comportamiento específico de un dispositivo se haría a través de cambios en el módulo **Visible Behavior**. Los cambios en las funciones de cálculo, los cuales estarían determinados por hechos físicos o matemáticos, o por mejoras en el desempeño o en la precisión se harían en el módulo de **Software Decisions**. Esta separación se hace necesaria debido a que los cambios en los requerimientos son motivados por agentes externos a los diseñadores del software, mientras que los cambios en la toma de decisiones (cálculos internos) no son determinados por los requerimientos sino por los diseñadores del software como tal.

El uso de más módulos hubiera sido contraproducente, debido a las restricciones de memoria que tenía el sistema en sí, y a los requerimientos de desempeño que estaban siendo solicitados los cuales eran altamente exigentes.

1. **¿Cuáles propiedades de la arquitectura de un sistema se pueden ver en una estructura de usos (Uses)? ¿Cuales propiedades se exponen en el caso de estudio?**

En una estructura de usos las propiedades visibles son los procedimientos que expone cada módulo y cuales módulos están autorizados a usar dichos procedimientos, esto nos da una visión general de como los módulos interactúan entre si y de las relaciones que existen entre ellos.

Para el caso de estudio se exponen los usos que hacen los diferentes módulos y sus sub-módulos de los procedimientos de otros sub-módulos. Los usos que se muestran en el caso de estudio son los siguientes, pero es tan solo una extracción del total.

|  |  |
| --- | --- |
| **Modulo/ Submodulo** | **Procedimientos expuestos** |
| EC: Extended Computer Module | DATA, PGM, IO, PAR |
| DI: Device Interface Module | ADC, IMS, PMDS, PNL, SWB |
| FD: Function Driver Module |  |
| SS: Shared Services Module | MODE, PNL, PNL.CONFIG, PNL.FORMAT, |
| AT: Application Data Type Module | NUM, STE |

1. **¿Qué ventajas conlleva una alta cohesión entre los módulos de una arquitectura? ¿Que lleva a tener una alta cohesión?**

Las ventajas de una alta cohesión entre los módulos de una arquitectura es que la información del módulo solo será relevante para el mismo modulo y estar relacionada en la medida de lo posible con el mismo modulo. Esto permite al módulo realizar una tarea sencilla y específica, de manera que interactúa poco o nada con los demás módulos, minimizando la dependencia entre los mismos.

La cohesión está directamente relacionada con el acoplamiento, pues ante mayor sea la cohesión de los módulos de una arquitectura, menor será el acoplamiento entre los mismos. Esto relación favorece la modificabilidad del sistema en general, pues los cambios que sean requeridos solamente son aplicados en los módulos afectados, sin impactar a los demás módulos. Incluso se puede hacer una modificación total en el sistema manejando pequeños cambios en cada uno de los diferentes módulos.

1. **Dibuje una vista en la que exponga la arquitectura del sistema en capas. Discuta las ventajas que tiene haber definido tales capas.**



Definir la arquitectura en capas nos da la ventaja de tener una visualización de cómo se organiza nuestra arquitectura y cómo interactúan los módulos entre ellas. Por ejemplo para el caso de estudio, tenemos una capa conformada por el modulo de computación **Computer Extended** y el modulo de datos **Application Data Type** los cuales nos proporcionan una maquina virtual para acceder los procedimientos de más bajo nivel. Los módulos **Function Driver** y **Shared Services** nos proporcionan la campa mas externa desde donde el sistema proveerá y recibirá información de los capas subyacentes. En la capa media tenemos los módulos **Physical Models**, **Filter Behavior**, y **Data Banker** los cuales son los encargados de manipular la información que se entregara hacia la capa superior usándolas implementación de la capa inferior (Maquina virtual). Paralela a estas capas esta la capa **Software Utilities**, la cual se encuentra ubicada de esta manera puesto que todas las capas pueden acceder a sus procedimientos como soporte a los requerimientos necesarios de sus propios módulos o sub-módulos.

1. **¿Cuáles de las propiedades de la arquitectura de un sistema se pueden ver en una estructura de procesos? ¿Cuales propiedades se exponen en el caso de estudio?**

En la estructura de procesos se pueden ver, valga la redundancia, el conjunto de procesos que componen el sistema. Un proceso es un conjunto de pasos que pueden ser ejecutados de manera periódica en un intervalo de tiempo, o en respuesta a un evento. Este conjunto de procesos proporciona una vista de la prioridad de los procesos y cómo deben ejecutarse y programarse para evitar bloqueos entre los mismos, es decir, nos proporciona la forma como deben sincronizarse entre ellos.

En el caso de estudio se pueden destacar los siguientes procesos:

* **Periódico:** Un proceso que se ejecuta periódicamente cada 40 milisegundos con el fin de obtener los valores de las entradas del sistema, realizar los cálculos correspondientes a estas entradas y llamar el procedimiento adecuado de la interfaz de dispositivos para enviar el resultado obtenido.
* **Por demanda:** Un proceso que espera el desencadenamiento de un evento, calcula el resultado del evento, y llama al procedimiento adecuado de la interfaz del dispositivos y envía la respuesta al evento
* **Procesos pesados:** Son aquellos que realizan se encargan de llamar a algún procedimiento cuya ejecución es costosa en términos computacionales. Con el fin de no hacer este tipo de procesos en demanda, estos se ejecutan en un segundo plano y almacenan el resultado obtenido, de esta manera, otro procedimiento puede acceder al último valor calculado de un procedimiento en particular sin tener que recurrir a calcularlo en el momento.

1. **Suponga que la versión del software ahora se quiere instalar sobre una máquina de entrenamiento de la aeronave. Esta máquina se encuentra en un ambiente diferente y no carga armas. ¿Que estructuras cambiarían en la arquitectura? ¿Cómo cambiarían?**

Puesto que el único cambio que existe en la aeronave es a nivel de armamento, es decir, físico, no de comportamiento, la única estructura que cambiaría seria la estructura de descomposición por módulos, específicamente el módulo de **Hardware**.

Los cambios que se requerirían sobre este módulo implicarían que las interfaces que estén relacionadas a armamento no sean públicas, por lo tanto ningún otro modulo tendría opción de relacionarse con ellas, por ende ningún procedimiento de la estructura de usos podría llamar los procedimientos que se relacionen a los sub-módulos de armamento, porque para el resto del sistema sencillamente no existirían. Esa es una de las ventajas de esconder la implementación de un módulo, ya que en el momento en que la interface deja de ser pública, este módulo deja de existir para el resto del sistema.