Sistemas en Tiempo Real

1º PEC

Nombre: Jorge Martinez Pazos

DNI: 39510046W

Correo: jmartinez5741@alumno.uned.es

Centro Asociado: Vigo (Pontevedra)

Índice

Ejercicio 1

Analizar en qué medida la elección de un método de diseño para sistemas en tiempo real debe estar influenciada por: (a) el posible lenguaje de implementación, (b) las herramientas de apoyo, (c) los requisitos de fiabilidad de la aplicación, (d) los requisitos de capacitación del personal, (e) consideraciones de marketing, (f) experiencias previas, (g) el coste.

a) Para comenzar, los sistemas en tiempo real tienen que interactuar con el entorno externo que los rodea, por lo que es necesario el uso de sensores, y la información percibida por estos debe ser procesada dentro de un tiempo límite, ya que si los datos procesados se obtienen fuera de plazo pueden producirse graves consecuencias. Para poder llevarse a cabo, se debe asegurar que el tiempo de ejecución del código esté por debajo de un valor especificado, y debe estar en constante cambio y mantenimiento. Asimismo, la seguridad es una de las cualidades más importantes que deben poseer los sistemas en tiempo real, por lo que el lenguaje empleado debe permitir introducir instrucciones que impidan que se produzcan errores inesperados.

Uno de los principales beneficios de usar un lenguaje de alto nivel es que le permite al programador abstraerse de los detalles de implementación y concentrarse en resolver el problema en cuestión. Pero los programadores de sistemas en tiempo real deben fijarse constantemente en el costo de las funciones y características del lenguaje.

Dentro de los sistemas en tiempo real existen 3 tipos de lenguajes:

- Secuenciales
- Ensamblaje
- Alto nivel

Dentro de estos lenguajes, las propiedades más importantes serían:

- Seguridad
- Legibilidad
- Flexibilidad
- Simplicidad
- Portabilidad
- Eficiencia

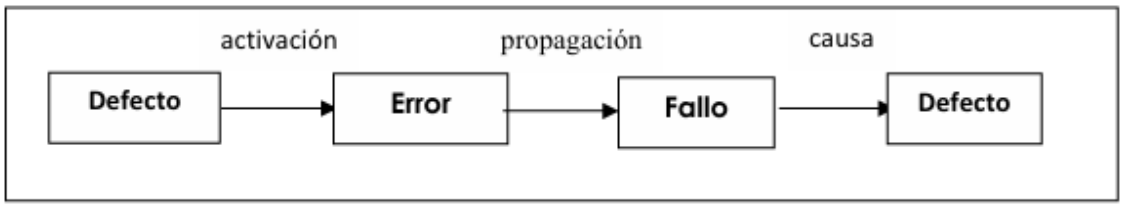
Debido a todos estos requerimientos, es importante la elección de un lenguaje adecuado.

b) A mayores de todo lo ya dicho, el propio lenguaje debería disponer de soporte multitarea y de sincronización entre estas, ya que puede simplificar mucho la traslación del diseño al código. Aunque muchos sistemas operativos tienen estas capacidades de multitarea, destacan por encima las aplicaciones empotradas que se escriben en un lenguaje que da soporte en tiempo real suficiente para la ejecución del programa en

tiempo real. Requiere menos memoria y puede ser adaptado a una aplicación, incrementando así el rendimiento. Partiendo de esta idea, y de las propiedades comentadas previamente, se pueden emplear con eficacia muchos lenguajes de programación de propósito general, como C, Java, Pascal… pero existen lenguajes específicos, denominados lenguajes en tiempo real; Ada, Jovial, Chill… caracterizados, precisamente, por su capacidad multitarea, implementación directa de funciones en tiempo real y características modernas de programación que ayudan al programador a asegurar el correcto funcionamiento del programa. Algunos de estos lenguajes también incluyen unas estructuras conocidas como módulos, que soportan los procesos de descomposición y abstracción, los cuales permiten definir subcomponentes del programa con papeles bien definidos, así como interconectados y relacionados entre sí, de forma clara y sin ambigüedades.

c) Los Sistemas en Tiempo Real, al igual que la gran mayoría de sistemas, constan de características críticas, importantes de manera genérica y universal, pero que lo son aún más debido a la naturaleza de los sistemas que estamos tratando, ya que, pecar en alguno de estos aspectos puede traer consecuencias catastróficas. En este caso, hablamos de la fiabilidad y la tolerancia a fallos.

La fiabilidad se define como una medida del éxito con el que el sistema se ajusta a la especificación dada para su comportamiento, y cuando no se ajusta a esa especificación, se dice que se ha producido un fallo. Se puede decir que un sistema altamente fiable es sinónimo de una baja tasa de fallos. Estos fallos son el resultado y la manifestación de problemas internos del sistema, llamados errores.



Muchos de estos sistemas en tiempo real controlan algunos de los sistemas críticos de la sociedad, por ende, el software debe estar diseñado con la máxima integridad, y los programas deben ser tolerantes a fallos y continuar funcionando, aunque sea con un servicio degradado. En el peor de los casos, un sistema en tiempo real debe garantizar la seguridad del medio antes de apagarse de una forma controlada. En la mayoría de las ocasiones, un fallo en el sistema podría afectar a vidas humanas, por lo que la fiabilidad debe estar medida lo mejor posible.

d) Para poder crear un Sistema en Tiempo Real, se necesita un personal capacitado y especializado, ya que estos sistemas suelen ser críticos y deben cumplir con requisitos estrictos de tiempo, fiabilidad, seguridad… como vimos antes. Deberían estar familiarizados con conceptos como la gestión de tareas, la sincronización, la concurrencia y la priorización de procesos, y, al menos, entender los lenguajes que pueden llegar a emplear, C, Ada, Java y lo que puede aportar cada uno según sea necesario, multitarea, gestión de recursos…. Asimismo, deben tener los conocimientos para diseñar estos

sistemas y que cumplan con los requisitos de tiempo real, descomponiéndolos en módulos, priorizando tareas y garantizando la sincronización entre los distintos procesos. También tienen que mantenerse al día con las últimas tendencias y avances en el campo de los sistemas en tiempo real, ya que evolucionan constantemente y hay que realizar mantenimiento a estos sistemas, pudiendo llegar a implementar algunos de estos avances, siempre y cuando mejoren alguna de las características del sistema, y no las empeoren. El personal debe ser capaz de identificar y resolver rápidamente posibles errores en el programa, especialmente en los sistemas críticos, y, además, deben ser capaces de optimizar el uso de estos recursos sin comprometer el rendimiento del sistema. Si el personal encargado de un proyecto de sistema en tiempo real no está cualificado, podrían producirse consecuencias terribles.

e) A pesar de que pueda parecer que no están muy relacionados, los sistemas en tiempo real y el mercado suelen ir de la mano, ya que las decisiones de diseño, implementación y comercialización del sistema están influenciadas por las necesidades del mercado, las expectativas de clientes y la competencia, y su éxito depende no solo de su funcionalidad, sino también de cómo se adaptan a las demandas y cómo se diferencian. Identificar tendencias, como la necesidad de sistemas más eficientes energéticamente, o la integración con Internet, por ejemplo, pueden influir en gran medida al diseño del sistema, y, para destacar en el mercado, se tienen que ofrecer, además de un sistema funcional, características innovadoras y fiables que diferencien de los competidores.

f) Respecto a las experiencias previas, un poco lo que comentamos previamente de que el personal debería estar familiarizado con las mejores prácticas de la industria para ofrecer y garantizar una mayor seguridad y fiabilidad, siendo mucho más sencillo si cuentan con experiencia previa en el desarrollo de estos sistemas, ya que les permitiría enfrentarse a desafíos, ya sean nuevos o más comunes, con una mayor confianza y probabilidad de éxito en la implementación.

g) Suponiendo que, en cuanto al coste, no se refiere exclusivamente al monetario, sino también a los recursos, el tiempo y el esfuerzo necesario para desarrollar, implementar y mantener el sistema, es claramente un factor crítico en el diseño y desarrollo de sistemas en tiempo real, ya que suelen ser complejos y requieren recursos significativos, tanto a nivel de hardware como de software. Se requiere un análisis detallado de los requisitos y una planificación cuidadosa para garantizar que el sistema cumpla con las restricciones propias de un sistema en tiempo real, lo que implica costes asociados a la contratación de personal y la adquisición de herramientas de diseño y simulación, así como otros lenguajes o herramientas específicas y la complejidad de la programación. En algunas ocasiones simplemente se requiere hardware especializado, como procesadores de alto rendimiento, sensores y otras piezas, las cuales son costosas. En otras, se deben tener en cuenta costes de formación para el uso específico de herramientas o lenguajes, para las cuales el personal debe estar capacitado, a mayores hay que tener en cuenta las pruebas exhaustivas, esenciales para los sistemas en tiempo real, pero que pueden ser altamente costosas en tiempo y recursos, y existen las situaciones donde los retrasos en el desarrollo pueden resultar en costes de oportunidad significativos, especialmente si la competencia

está desarrollando un producto similar. Es necesaria una planificación y gestión eficaz de los costes, no solo el monetario, para el éxito del proyecto.

Ejercicio 2

a) Explicar en qué suposición se basa, cuando deja de ser válida y que hay que realizar para que la suposición sea cierta. b) ¿De qué es responsable el proceso director (driver process)? c) ¿Qué se entiende por votación inexacta (inexact voting)? Como se puede resolver y en que consiste el problema de la comparación consistente (consistent comparison problema). d) Enumerar y comentar los tres factores (aspectos= issues) principales del éxito de la programación de las N-versiones.

a) La idea de que la programación de N-versiones es una técnica útil que realmente mejora la tolerancia a fallos de un sistema, se asienta sobre dos suposiciones: la primera, que es posible especificar un programa de forma completa, consistente y sin ambigüedad, y la segunda, que los programas que son desarrollados de manera independiente fallan de forma independiente. Esta última suposición puede no ser correcta en ciertas condiciones, como, por ejemplo, el caso de caso de escribir las versiones en el mismo lenguaje, pudiendo haber errores derivados de la implementación del lenguaje que sean comunes a distintos programas de las distintas N versiones. Se deberían usar distintos lenguajes y entornos de programación para cada versión, y, de usar el mismo lenguaje, al menos deberían usarse distintos compiladores y entornos de fabricantes. Por otro lado, para proteger el sistema de posibles fallos físicos, las N versiones se deben distribuir en distintas máquinas que tengan líneas de comunicación tolerantes a fallos.

b) El proceso director controla el programa con N versiones, y es responsable de:

- Invocar a cada una de las versiones.
- Esperar a que las versiones completen su tarea.
- Comparar los resultados y actuar en base a éstos.

El director y las N versiones deben comunicarse durante el transcurso de sus ejecuciones y no, simplemente, cuando éstas finalizan. La comunicación entre ambos dependerá del lenguaje de programación de cada versión, así como sus modelos de concurrencia.

La eficiencia y la facilidad con la que el proceso director puede comparar los votos y decidir cuando existe un desacuerdo es crucial en la programación de N versiones, y, aunque a veces es sencillo realizar esta comparación de votos, existen situaciones donde las operaciones no producen resultados de una naturaleza exacta o con una única respuesta correcta. Para poder hacer frente a estas situaciones, el proceso director cuenta con algoritmos de "votación inexacta", que comprueba si los valores arrojados por las versiones están dentro de un rango dado, usando para ello una estimación previa, o bien un valor medio de los N resultados obtenidos. Aunque esta técnica es sencilla, puede resultar difícil encontrar un enfoque general válido para la votación inexacta.

c) Existen otros problemas asociados a la aritmética de los números reales mediante precisión finita, uno de ellos es el llamado "problema de la comparación consistente", el cual se produce cuando se tiene que realizar una comparación basada en un valor finito

dado en la especificación del sistema y el resultado de dicha comparación determina el curso de la acción. Un ejemplo es un sistema de control de procesos que monitoriza sensores de temperatura y de presión, y toma las decisiones apropiadas de acuerdo con los valores obtenidos de éstos para asegurar la integridad del sistema. Cuando cualquiera de las lecturas de alguno de los sensores sobrepasa un valor umbral, se debe realizar una acción correctora. Suponiendo 3 versiones, que monitorizan ambos sensores y deciden la toma de alguna acción en base a las medidas que realizan de dichos sensores, se realiza la votación del resultado y, en una aritmética de precisión finita, provocará que cada versión obtenga valores distintos. El problema de comparación consistente se da cuando ambas lecturas están alrededor de sus valores umbral. Teniendo T1,T2,T3,P1,P2 y P3, si T1 y T2 están por debajo del umbral, pero T3 no, será sólo V3 quien tome medidas correctoras, pero si V1 y V2 pasan a otro punto de comparación, esta vez con el sensor de presión, es posible que P1 esté por debajo del umbral y P2 por encima. Al final, el resultado será que las tres versiones han seguido rutas diferentes y producirán resultados diferente, pero totalmente válidos.

d) La correcta aplicación del método de programación de N versiones depende de que se cumplan varios factores:

Especificación inicial: Una gran cantidad de los fallos existentes en el software provienen de una especificación inadecuada. En este sentido, ayudan en gran medida los métodos de especificaciones formales.

Independencia en el diseño: La hipótesis de que el software producido de manera independiente presentará fallos distintos no está del todo probada. Allí donde la especificación resulta compleja, se producirá inevitablemente una falta de comprensión de los requisitos por parte de todos o algunos de los equipos al cargo del desarrollo de las distintas versiones que se producen de forma independiente. Si estos requisitos se refieren también a datos de entrada que son poco frecuentes, entonces es posible que no se detecten errores de diseño comunes en la fase de pruebas. A pesar de todo lo anterior, hay estudios que indican que un sistema de 3 versiones es entre cinco y nueve veces más fiable que un sistema de una única versión.

Presupuesto adecuado: En la mayoría de los sistemas empotrados, el principal coste de desarrollo del sistema se debe al software. Un sistema de tres versiones casi triplicará el presupuesto en comparación al sistema de una versión, y, lo que, es más, aumentará mucho la complejidad de este, provocando problemas de mantenimiento.

Por todo lo anterior, aunque la programación de N versiones desempeña un papel muy importante en la producción de software fiable, debe ser utilizada con cuidado y, siempre que sea posible, junto con otras técnicas distintas.

Bibliografía

- Material adicional – Sistemas en Tiempo Real parte 1.pdf