Ingeniería de la confiabilidad

Andres Matarrita Miranda C04668
Esteban Mora Garcia C05126

01

Introducción

¿Es necesaria la ingeniería de la confiabilidad?

02

Definición del tema

Definicion de ingenieria de confiabilidad en el área de software

- Enfoque en sistemas de software confiables y estables.
- Evitar fallos y optimizar el rendimiento del software.
- Minimizar errores e interrupciones.
- Software predecible y consistente.
- Brindar confianza y tranquilidad al usuario.

03

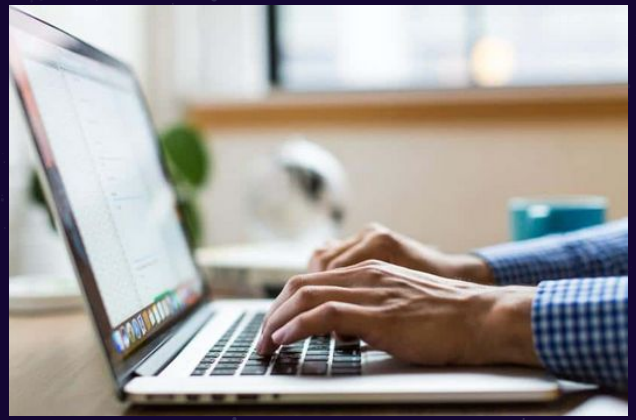
Disponibilidad y Confiabilidad

Definición de conceptos

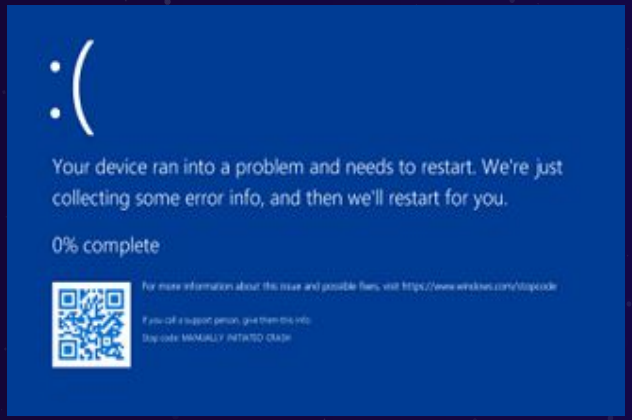
- **Confiabilidad:** Probabilidad de operación sin fallas en un tiempo específico

- **Disponibilidad:** Probabilidad de que un sistema, en un momento determinado, sea capaz de proveer los servicios solicitados

Ejemplo de Confiabilidad



Ejemplo de Disponibilidad



04

Requerimientos de Confiabilidad

Tipos de requerimientos

- **Funcionales:** Son los requisitos que describen las acciones y funciones específicas que el sistema debe realizar. Se centran en las entradas, salidas y procesamiento de datos del sistema.

- **No funcionales:** se centran en restricciones del sistema, que no están directamente relacionados con las funciones específicas. Se enfocan en aspectos como el rendimiento y la usabilidad

Métricas

Se tienen 3 tipos de métricas para la confiabilidad

- Probabilidad de fallo bajo demanda (POFOD)
- Tasa de ocurrencias de falla (ROCOF)
- Disponibilidad (AVAIL)

Requerimientos funcionales de confiabilidad

Se tienen 4 tipos requerimientos funcionales de confiabilidad

- Requerimientos de revisión
- Requerimientos de recuperación
- Requerimientos de redundancia
- Requerimientos de procesos

05

Arquitecturas Tolerantes a Fallas

Que significa Tolerante a Fallas?

En runtime los sistemas son capaces de aplicar mecanismos para continuar con su funcionamiento incluso después de que se haya producido una falla de software o hardware

Patrones de Arquitecturas Tolerantes a Fallas

Se tienen 3 patrones utilizados en arquitecturas tolerantes a fallas

- Sistemas de protección
- Arquitecturas auto-monitoreadas
- N-version programming

06

Programación para la confiabilidad

Importancia de la programación en la confiabilidad de los sistemas

Reducción de errores

Buenas prácticas de programación permiten reducir errores y asegurar la estabilidad de los sistemas.

Experiencia segura y sin problemas

La programación adecuada garantiza un funcionamiento confiable de los sistemas, ofreciendo una experiencia segura y sin problemas.

Evita fallas y comportamientos inesperados

Una programación cuidadosa y precisa ayuda a evitar dichos problemas

Estándares y directrices

Existen estándares y directrices establecidas que ayudan a crear software resistente y seguro, minimizando los riesgos de vulnerabilidad.

Directriz 1: Controlar la visibilidad de la información en un programa

- Control de acceso basado en la "necesidad de saber"
- Implementación de tipos abstractos
- Uso de interfaces:

```
using System;

interface ISecretData
{
    void AccessSecretData();
}

class SecretData : ISecretData
{
    private string secretInformation = "Confidential information";

    public void AccessSecretData()
    {
        Console.WriteLine(secretInformation);
    }
}

class Program
{
    static void Main(string[] args)
    {
        ISecretData secretData = new SecretData();
        secretData.AccessSecretData(); // Salida: "Confidential information"
    }
}
```

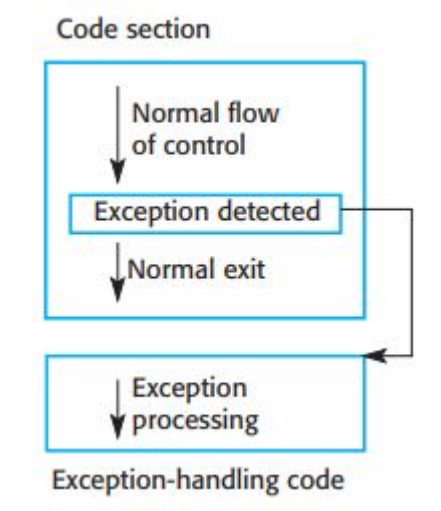
Directriz 2: Comprobar la validez de todas entradas

- Importancia de la verificación de validez
- Tipos de verificaciones
- Acciones adecuadas en caso de falla

```
decimal withdrawalAmount;
Console.Write("Ingrese la cantidad a retirar: ");
if (decimal.TryParse(Console.ReadLine(), out withdrawalAmount)
    && withdrawalAmount > 0)
{
    // Realizar la transacción de retiro
    Console.WriteLine("Retiro exitoso. ");
}
else
{
    Console.WriteLine("Cantidad de retiro invalida.");
}
```

Directriz 3: Proporcione un controlador para todas las excepciones

- Definir manejadores de excepciones
- Garantizar la detección y manejo explícito de las excepciones
- Utilizar mecanismos integrados de manejo de excepciones
- Realizar pruebas exhaustivas



Directriz 4: Proporcionar capacidades de reinicio

- Copias de formularios en sistemas de comercio electrónico:
- Guardado automático de datos en transacciones largas o intensivas
- Capacidad de retroceso y reinicio para usuarios en caso de errores
- Evitar elementos propensos a errores en la programación

Directriz 5:Incluir tiempos de espera al llamar a componentes externos

```
// Lógica para enviar el correo electrónico utilizando SmtpClient
using (SmtpClient smtp = new SmtpClient("smtp.office365.com", 587))
{
    smtp.EnableSsl = true;
    smtp.Credentials = new NetworkCredential("dinamita_PI@outlook.com", "PI_JUNQUILLAL");
    smtp.DeliveryMethod = SmtpDeliveryMethod.Network;
    smtp.UseDefaultCredentials = false;
    smtp.Timeout = 5000; // Establecer timeout en 5 segundos (5000 milisegundos)

    using (MailMessage mail = new MailMessage())
    {
        mail.From = new MailAddress("dinamita_PI@outlook.com", "Confirmacion de reserva");
        mail.To.Add(new MailAddress(correo));
        mail.Subject = "Un gusto saludarle por parte de Junquillal";
        mail.IsBodyHtml = true;
        mail.Body = mensaje;

        try
        {
            smtp.Send(mail);
        }
        catch (SmtpException ex)
        {
```

07

Conclusión

Importancia de la ingeniería de confiabilidad en la ingeniería de software

Garantiza la calidad y confiabilidad de los sistemas

Optimiza la disponibilidad y el rendimiento de los sistemas

Minimiza los costos asociados con fallas y reparaciones

Impulsa la competitividad y el éxito del negocio

Gracias!