USO DE INSTRUMENTAÇÃO E DISCIPLINA DE TRABALHO COMO MECANISMO DE AUMENTAR A CAPACIDADE DE DETECÇÃO DE FALHAS

Aluno: Thiago Pinheiro de Araújo
Orientador: Arndt von Staa

Introdução

Este trabalho é uma continuação do estudo de caso apresentado no ano passado. O presente estudo consiste em calcular quanto esforço é gasto com manutenção e depuração de código, quando este é construído com uso de instrumentação e com disciplina de trabalho. A partir destas medições desejamos ser capazes de avaliar o quanto é vantajoso utilizar estas técnicas na construção de sistemas de missão crítica

O objeto estudado é um software de missão crítica que tem por objetivo adquirir e visualizar sinais oriundos de sensores através de uma eletrônica embarcada em um robô. A aplicação prática deste software é utilizada para adquirir, tratar e exibir sinais de sensores de ultra-som montados em um robô de inspeção de dutos chamado PIG.

Objetivos

O objetivo deste trabalho é o de desenvolver um sistema de software de missão crítica para ser utilizado na inspeção de dutos utilizando a tecnologia de ultra-som. Ao concluir etapas, medir o quanto robusto e fiel é o sistema e medir quanto esforço foi gasto em manutenção e depuração de código.

Metodologia

Sistemas de missão crítica são sistemas de software que requerem um elevado nível de qualidade e têm como principais requisitos não-funcionais a necessidade de uma elevada confiabilidade e robustez. Entende-se por confiável o software que, sempre que solicitado, produz resultados corretos, precisos e exatos. O requisito robustez diz respeito ao tratamento eficiente de exceções, podendo recuperar o sistema ou cancelar a operação informando a origem da falha. Outro requisito importante é a fidelidade das informações exibidas, que diz respeito à veracidade das informações visualizadas. Para alcançar estes objetivos foram utilizadas as técnicas descritas a seguir.

A primeira delas é a utilização de padrões de projeto [1], que são soluções de eficiência já comprovadas e amplamente utilizadas no desenvolvimento de software. Estas soluções são desenvolvidas por especialistas e tornam-se padrões por poderem ser reutilizadas em diversos projetos e por terem sua eficácia comprovada.

Sua utilização ajuda no processo de definição da interface de cada componente do software e conseqüentemente na divisão de responsabilidades entre eles. Um dos benefícios obtido a partir da utilização de padrões de projeto é a facilidade na manutenção e evolução tendo em vista que para adicionar um novo componente, é necessário apenas conectá-lo no modelo existente, com um custo muito baixo em termos de codificação.

Outro benefício trazido é a facilidade de correção de falhas a partir do momento de sua detecção. Devido à arquitetura bem cuidada, o esforço gasto para a correção da maioria das falhas resume-se à alteração de poucos módulos.

Outra técnica é a utilização de *Design by Contract*, em que a idéia central é a criação de contratos entre os módulos do software [1, 2], ou seja, definir com precisão as condições que devem ser atendidas pelos clientes (código que chama um método) e servidores (os métodos

chamados). Para verificar a corretude da execução, estas condições devem ser testadas antes (pré-condições) e depois (pós-condições) da execução de uma rotina.

O mecanismo para utilizar esta técnica é a inserção de assertivas executáveis e métodos de validação da integridade das estruturas [3]. Para isto são utilizados verificadores estruturais que consistem em métodos especificamente projetados e implementados para verificar se as estruturas satisfazem as suas assertivas estruturais. Uma condição não satisfeita cancela a execução do software, indicando a condição não satisfeita, os parâmetros utilizados para esta verificação quando possível e o local (linha de código e nome do módulo) em que foi realizada a verificação. Chamamos isso de programação defensiva [4].

E por fim a técnica de teste automatizado de módulos [3]. O objetivo de utilizar esta técnica é garantir a corretude dos componentes criados. Esta técnica se faz necessária tendo em vista o volume muito grande de funções que cada componente proporciona. Como os componentes são criados de forma incremental, um teste manual é muito suscetível a falhas já que deve ser realizado a cada implementação de uma nova funcionalidade.

Conclusões

O software desenvolvido é eficaz, pois atende aos requisitos: confiabilidade e robustez. O sucesso obtido deve-se a utilização instrumentação e disciplina de trabalho como mecanismos de detecção de falhas.

O requisito de facilidade de evolução também foi atendido devido à distribuição de responsabilidades e à utilização de padrões de projetos na sua arquitetura. A adição de um novo componente limita-se à escrita dos módulos que o compõe, seguindo de um esforço desprezível para conectá-los ao software.

Concluindo, o esforço adicional realizado em um desenvolvimento utilizando mecanismos para detecção de falhas é muito pequeno quando comparado com os seus benefícios. Isto pode ser comprovado por este estudo de caso: obtivemos um sistema de grande porte, cuja maior parte das falhas foi descoberta a partir de assertivas. Como esperado a maioria das falhas puderam ser resolvidas alterando uma fração muito pequena de código, reduzindo assim o trabalho excedente, ou seja, o esforço gasto além do realizado na produção do software.

Referências

1 - MEYER, B.; Eiffel: The Language. Prentice Hall; 1992

2 - MEYER, B.; “Applying Design by Contract”; IEEE Computer 25(10); Los Alamitos, CA: IEEE Computer Society; 1992; pags 40-51

3 - STAA, A.v.; Programação Modular; Rio de Janeiro: Campus; 2000

4 - Defensive programming, http://en.wikipedia.org/wiki/Defensive_programming