Relatório: Automação Industrial I

Professora: Yona Lopes Meira de Vasconcellos

Aluno: Rodrigo Gonçalves Leite

Processo:

Automação de teste de software com simulação de falhas de infraestrutura.

Objetivo:

Devido a sua natureza intangível, o desenvolvimento de software sofre com uma série de incertezas. Frederick Brooks, em seu livro "O Mítico Homem Mês" [1]descreve o panorama do desenvolvimento de software no passado e conclui que a aderência do produto final às funcionalidades para as quais ele foi desenvolvido não é uma tarefa fácil.

Para vencer esta dificuldade, a disciplina de testes de software foi desenvolvida. Sua importância é tamanha que toda uma metodologia de desenvolvimento de software baseado em testes foi abraçada pela comunidade de programação. Mesmo do ponto de vista mais abstrato ela ganha seu próprio capítulo em uma das publicações mais influentes sobre o assunto o *"Software Engineering Body of Knowledge"*[2].

Teste manual de software é uma tarefa cara e repetitiva, logo, é candidato perfeito para o processo de automação. Consequentemente, diversas ferramentas tanto proprietárias como de uso livre se consolidaram no mercado. Uma parte importante desta equação não pode ser descartada: todo sistema tem uma infraestrutura de hardware associada. Esta sinergia esta intimamente ligada aos riscos envolvidos no processo de construção, mas testar esta integração geralmente significa dispor de componentes reais análogos ao projeto final. Normalmente o custo deste passo é proibitivo para projetos menores. Por este motivo construímos uma simulação via software que aliada a técnicas de virtualização de ambientes digitais atenda as especificações do hardware final.

Funcionamento do Sistema:

A filosofia é incluir uma camada de software responsável por simular falhas de infraestrutura de acordo com a configuração. Para tal, foi desenvolvido um arcabolso que não interfere em nenhuma das camadas envolvidas, isso significa que tanto o processo tradicional de testes como a estrutura interna do software testado não precisam sofrer nenhuma alteração.

Para alcançar este objetivo uma instância do componente Controlador ficar responsável por orquestrar a simulação. O primeiro passo é enfileirar as requisições dos Scripts de Teste em uma fila capaz de gerenciar cada *thread* – no modelo uma por execução de script em uma Fila de Testes.

---

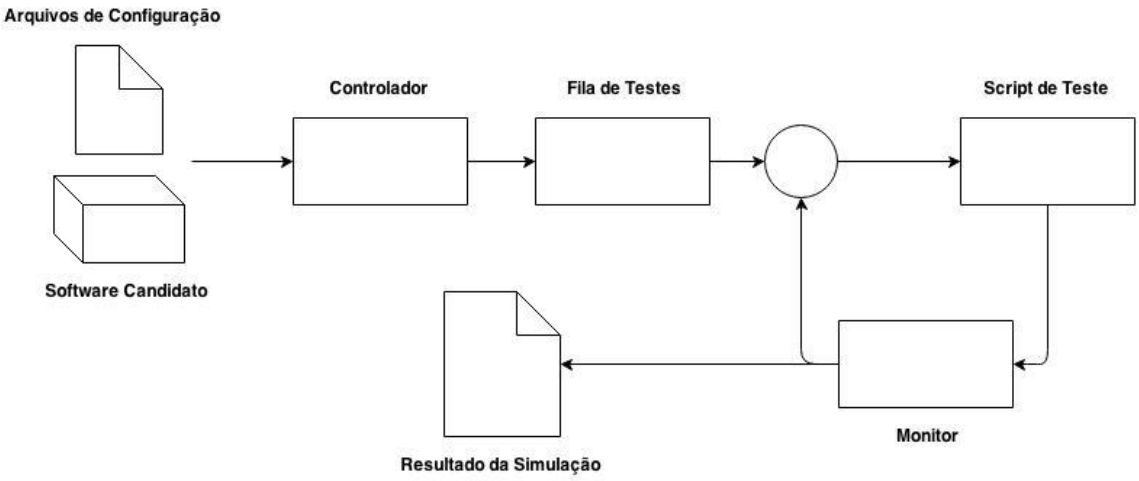
[1] ISBN-*10: 853523487X*

[2] http://www.computer.org/portal/web/swebok

Um componente Monitor guarda o estado atual da simulação e é consultado cada vez que um Script de Teste for executado, assim o sistema consegue julgar se é o momento de interromper o funcionamento correto do sistema para simular uma falha de infraestrutura.

Ao final do processo um relatório de Resultado da Simulação é emitido, permitindo uma melhor compreensão do comportamento do software testado durante todo o processo.

Principais Componentes:



Todo o arcabolso de testes foi desenvolvido utilizando o paradigma de orientação a objetos, logo podemos facilmente identificar as peças e suas responsabilidades.

a. Software Candidato: o arcabolso é conectado como uma dependência do software desenvolvido e não interfere em sua estrutura. Recomenda-se que esta relação de dependência seja mantida apenas durante os testes, pois não há necessidade de mantê-la no produto final.
b. Arquivos de Configuração: a simulação deve ser calibrada para atender os níveis de restrição e características do ambiente real. Para tal, arquivos de configuração permitem que se definam estes parâmetros de acordo com métricas consolidadas no mercado de gestão de sistemas da informação – para uma descrição mais precisa veja apêndice A.
c. Controlador: representa a peça que constrói e orquestra a simulação, é responsável por mediar a comunicação entre seus principais componentes.
d. Monitor: guarda o estado atual da simulação e, mantendo aderência aos parâmetros dos arquivos de configuração, atua como sensor central do sistema. Serve para que o componente que efetivamente dispara os testes saiba se a simulação deve o não simular uma falha de infraestrutura.
e. Fila de Testes: é um dos atuadores do sistema, através das configurações fornecidas enfileira os scripts de teste automatizado.
f. Script de Teste: atuador principal do sistema. Através de um consulta ao estado atual da simulação, o componente interromper sua execução

mimetizando uma falha de hardware que, em seguida, é registrada no monitor da simulação.

g. Resultado da Simulação: como saída, o sistema publica as estatísticas para a avaliação da eficácia da simulação.

Detalhamento:

Este projeto foi desenvolvido para apoiar o projeto acadêmico, do curso de graduação em engenharia da computação do instituto infnet, intitulado: “Arquitetura de Sistemas Distribuídos com Múltiplos Requisitos de Integração”. Durante seu desenvolvimento foi necessário garantir alguma fidedignidade na coleta de dados de desempenho de diversas arquiteturas de sistemas de software candidatas. Como se trata de um projeto estudantil desenvolvido com recursos limitados, não existia a possibilidade de uso de uma infraestrutura análoga a diversos sistemas diferentes comunicando-se em uma gama hardwares diferentes.

Tecnologia envolvida na simulação e coleta de dados:

a. Plataforma de Desenvolvimento: Java Enterprise Edition 6ª. Versão
   https://www.jcp.org/en/jsr/detail?id=316

b. Ferramenta de Virtualização: instancias de máquinas virtuais Virtual Box preparadas com sistema operacional Linux Debian
   https://www.virtualbox.org/
   https://www.debian.org/

c. Ferramenta de Testes automatizados: arcabolso SeleniumHq
   http://docs.seleniumhq.org/

A possibilidade de simular falhas de hardware através de software viabilizou a execução do projeto. Espera-se que, através do refinamento deste processo, os custos de testes de integração possam ser reduzidos aumentando a mitigação de riscos de desenvolvimento em projetos com restrições similares.

Repositório do Projeto:

https://github.com/Gunisalvo/Automaton

Apêndice A – Métricas de Confiabilidade de Software:

O embasamento na escolha das métricas de confiabilidade de software se apoia no livro "Introdução e Engenharia de Software[3]" de Ian Sommerville. Estas métricas encontram paralelos em diversas publicações, incluindo a especificação ITIL.

a. Disponibilidade / *Availability*: porcentagem de tempo que o serviço estará disponível em relação a seu tempo total de operação (em razão proporcional, 1,0 representa 100% de disponibilidade).

b. Tempo médio entre falhas / *Mean time to fail*: quantidade de tempo média entre a ocorrência de cada falha (em milissegundos).

c. Taxa de Ocorrência de Falhas / *Rate of Occurrence of Failures*: porcentagem de distribuição de ocorrência de falhas pela amostra total de tempo (em razão proporcional, 1,0 representa 100% do tempo uma requisição falhará).

d. Probabilidade de Falhas em Uso / *Probability of Failure on Demand*: porcentagem de requisições não atendidas em relação ao número total de requisições (em razão proporcional, 1,0 representa 100% das requisições serão falhas).

Apêndice B – Outras Configurações:

a. Requisições por Minuto / Requests per Minute: número de testes enfileirados por minuto.

b. Tempo de Simulação / Simulation Runtime: tempo total de simulação (em milissegundos).

[3] ISBN*: 9788579361081*