1. INTRODUÇÃO

Nos últimos quatro anos, no ambiente universitário, tive oportunidade de conversar com um grande número de funcionários das empresas de TI da região. Esse fato levou-me a perceber como os testadores de software estão descontes e desmotivados com as atividades que estão exercendo nessa área. As principais reclamações surgem da repetição de tarefas maçantes que são executadas com muita frequência e a falta de autonomia nas atividades de verificação.

Qualidade de software não está ligada apenas a encontrar mais defeitos, mas também uma questão de melhoria sistêmica da perspectiva de uso do software. Por isso é essencial que as equipes de testes estejam motivadas na execução de suas tarefas, visando a redução dos custos associados a depuração e simulação dos incidentes.

Neste contexto, a mudança de perspectiva em relação ao teste de software constitui em um dos grandes desafios da empresa. Atualmente, o teste é uma atividade totalmente manual, realizada ao final do desenvolvimento por uma equipe independente. Na nossa área, passamos a vida toda automatizando os processos de nossos clientes e muitas vezes não somos capazes de automatizar os nossos processos.

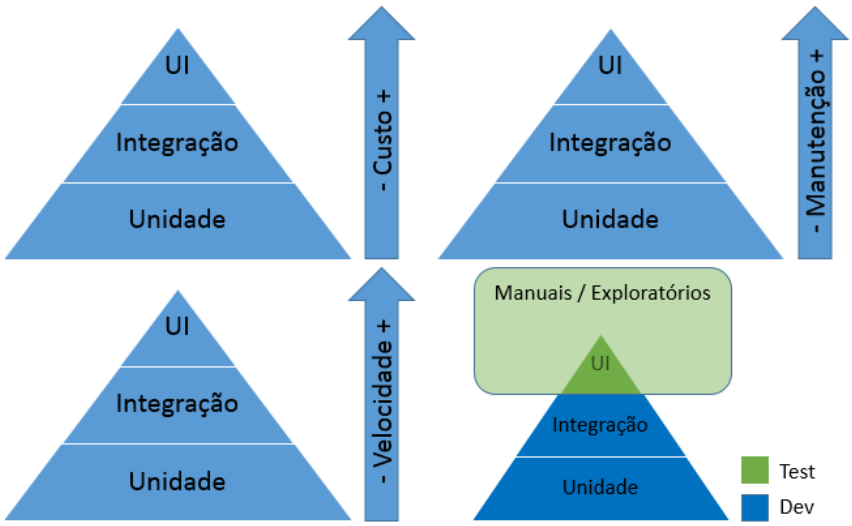
2. OBJETIVOS

Ofertar um meio de:

- **Aproveitar o conhecimento de programação de alguns testadores:**
  - Testadores criando os principais testes automatizados de interface;
  - Criar um conceito de multidisciplinaridade (programador/testador);
  - Motivar o profissional da área com tarefas criativas;
- **Facilitar e simplificar o teste:**
  - Reduzir o número de tarefas repetitivas e maçantes executadas manualmente pelo testador;
  - Dedicar maior tempo para testes exploratórios;
  - Maior tranquilidade e segurança nos testes;
- **Reduzir os custos de desenvolvimento e teste.**
  - Não perder tempo com verificação trivial;
  - Identificar com maior velocidade a ocorrência de problemas;
  - Fornecer maior segurança nos testes de regressão;
  - Dos custos e tempos consumidos nos testes que antecedem as liberações;

3. APELOS DO MODELO

O autor Mike Cohn descreveu um perfil de investimento em testes automatizados, no qual ele considera o esforço ideal e viável. Ele propôs um triângulo de testes automatizados, sendo que deve-se concentrar maior esforço no nível de unidade e depois reduzir esse esforço nos próximos níveis.



Nesta pirâmide fica claro que a maior parte dos testes automatizados são os testes de unidade, na parte inferior. Ao mover-se para cima da pirâmide, o número de testes (a largura de sua pirâmide) fica menor. Um estudo divulgado pela equipe de testes do Google sugere uma divisão 70/20/10: 70% testes unitários, 20% testes de integração e por fim 10% de testes de interface.

Em suma, os testes unitários e especialmente os testes de integração necessitam de maior conhecimento de programação, pois nesses testes é necessário um entendimento prévio das classes de infraestrutura e negócio do sistema. Os testes de interface são diferentes, podem ser gravados por ferramentas de automação de testes ou podem ser programados, usando apenas o framework de automação. Portanto, exige menor conhecimento de programação.

APLICAÇÃO

- **Mais de 50 (cinquenta) testadores de software:**
  - Pelo menos 10 (dez) possuem conhecimento básico de programação;
  - Evitar o desperdício desse conhecimento;
  - Existem casos de sucesso de testadores que viraram programadores;
- **Mas a ideia é automatizar tudo?**
  - Testador atuaria na automação de 10% dos testes da aplicação;
  - Utilizar a ferramenta de testes Telerik / Ranorex para automação de interface;
  - Focar nas funcionalidades principais/críticas que geram maior esforço nos testes manuais;