Programação Redução de Defeitos, Lista dos 10 melhores.

(Software Defect Reduction Top 10 List)

Recentemente, um Instituto Nacional de Ciências A subvenção da Fundação permitiu-nos estabelecer o Centro de Engenharia de Software de Base Empírica. CeBASE busca transformar engenharia de software, tanto quanto possível, de uma prática baseada em moda para uma prática baseada em engenharia por meio de derivação, organização e disseminação de dados empíricos sobre desenvolvimento de software e fenomenologia da evolução. A frase “tanto quanto possível” reflete a fato de que o desenvolvimento de software deve continuam a ser um campo intensivo em pessoas e em constante mudança. Descobrimos, no entanto, que os pesquisadores estabeleceram dados objetivos e quantitativos, relacionamentos e modelos preditivos que ajudam desenvolvedores de software evitam armadilhas previsíveis e melhoram sua capacidade de prever e controlar projetos de software eficientes.

**Dois**  
  
Os projetos de software atuais gastam cerca de 40 a 50 por cento de seu esforço em retrabalho evitável. Esse retrabalho consiste no esforço despendido corrigir dificuldades de software que poderiam foram descobertos anteriormente e corrigidos menos dispendiosa ou totalmente evitada. Reduzir o retrabalho evitável pode fornecer melhorias significativas no software produtividade. Em nossa análise comportamental de como os drivers de custo de software afetaram esforço para o modelo Cocomo II , descobriram que a maior parte da economia de esforço gerada pela melhoria do processo de software maturidade, arquiteturas de software e gerenciamento de risco de software vieram de reduções no retrabalho evitável.   
  
**Tres**  
  
Cerca de 80% do retrabalho evitável vem de 20 por cento dos defeitos. Um sistema de rastreamento de relatórios de problemas de software que registram o esforço para corrigir cada defeito permite analisar os dados com bastante facilidade para determinar e abordar as principais fontes adicionais de retrabalho.  
  
**Quatro**  
  
Cerca de 80 por cento dos defeitos vêm de 20 por cento dos módulos, e cerca de metade dos módulos estão livres de defeitos.Estudos de diversos ambientes ao longo de muitos anos mostraram, com incrível consistência, que entre 60 e 90 por cento dos defeitos surgem de 20 por cento dos módulos, com uma mediana de cerca de 80 por cento. Com igual consistência, quase todos os defeitos se agrupam em cerca de metade dos módulos produzidos.  
  
**Cinco**  
  
Por exemplo, uma análise do histórico de falhas de software de nove grandes softwares IBM produtos revelaram que cerca de 0,3 por cento dos defeitos representaram cerca de 90 por cento do tempo de inatividade.  
  
**Sete**  
  
Revisões baseadas em perspectiva alcançam 35 por cento mais defeitos do que não direcionados avaliações. Uma técnica de leitura baseada em cenários oferece um conjunto de procedimentos formais para detecção de defeitos com base em várias perspectivas. Essa abordagem busca gerar técnicas focadas para metas específicas de detecção de defeitos, tomando vantagem de uma organização existente histórico de defeitos. Melhorias na detecção de falhas as taxas variam de 15 a 50 por cento.   
  
**Oito**  
  
Estes incluem a Sala Limpa da Harlan Mills processo de desenvolvimento de software e Watts Processo de software pessoal de Humphrey . O forte foco do PSP na análise da causa raiz dos defeitos de software de um indivíduo e superações e no desenvolvimento pessoal listas de verificação e práticas para evitar futuras recorrência, reduziu significativamente as taxas de defeitos pessoais. Dependem de fatores como a maturidade de software existente da organização nível e os funcionários e organizações vontade de operar dentro de um cultura de software estruturada.Quando você par PSP com o fortemente compatível Processo de software da equipe , defeito as taxas de redução podem subir para fatores de 10 ou superior para uma organização que opera em um nível de maturidade modesto.   
  
**Dez**  
  
Cerca de 40 a 50 por cento dos programas do usuário contêm defeitos não triviais. Um estudo de 1987 nesta área descobriu que 44 por cento de 27 programas de planilhas produzidos por desenvolvedores experientes de planilhas continham defeitos não triviais - principalmente erros de fórmulas de planilha. Experimentos de laboratório subsequentes relataram taxas de planilhas com defeito entre 35 e 90 por cento. As fileiras de programadores de usuários «aprendiz de feiticeiro» também inchará rapidamente,dando a muitos que têm pouco treinamento ou experiência em como evitar ou detectar defeitos de alto risco enorme poder para criar defeitos de alto risco.   
  
Se classificarmos os desenvolvedores de páginas da Web ativos como programadores de usuários, essa previsão parece estar na pista. Este desafio de pesquisa de engenharia de software é um dos vários identificados por um National Estudo da Science Foundation,«Gaining Controle Intelectual do Desenvolvimento de Software»,que resumimos recentemente em Computador .

Teste de software e Necessidades da indústria.

(Software Testing and Industry Needs)

Perdoe minha resposta jocosa, mas entender o estado de qualquer prática de software é difícil. Editei uma edição especial do IEEE Software sobre o tema geral da prática de engenharia de software em 2003. O que ela queria dizer que a prática é tão variada que é difícil generalizar. Esse é certamente o caso do estado da prática de teste de software.   
  
Como um dos artigos nesta edição afirma, pode haver uma compreensão geral comum do que teste é, mas as práticas variam tremendamente em todas as empresas. No balanço, eu realmente acredito que a maioria dos testes de software na indústria atende efetivamente às necessidades da indústria. Baseio meu viés na observação de que isso é a era da computação - uma era que não ter sucesso sem software de sucesso e na minha crença de que o software não pode ser bem sucedido sem testes eficazes. A maioria dos projetos de software, eu afirmar fortemente, produzir produtos que o que eles devem fazer.   
  
**Uma Busca de Carreira**  
  
Meu palpite é que 65 por cento dos profissionais experientes avaliam práticas como medíocres, com 25 por cento mais avaliá-lo como negativo, deixando um franja de otimistas. Vejamos a contratação e o desenvolvimento de carreira como um guia presumido para todas as práticas de teste. Para generalizar demais,a prática em essas áreas é muitas vezes medíocre. Os trabalhos de teste são muitas vezes prêmios de consolação para aqueles que não considerado bom o suficiente para contratar como engenheiros de software e a diferença salarial entre desenvolvedores e testadores permanece significativo.   
  
Testadores com frequência reclamar de falta de respeito,credibilidade e influência. A maioria dos testadores é autodidata, e muitos nunca leram um livro sobre o sujeito. Muitos testadores amam o trabalho e não trocaria suas carreiras por nada. Apesar as complicações e equívocos, os testadores têm oportunidades extraordinariamente fortes de contribuir para uma sociedade que depende cada vez mais de software confiável.   
  
O teste eficaz é uma busca e, como qualquer missão, inclui intelectual desafio, debate apaixonado e a emoção da descoberta.Recompensas para os melhores 20% dos testadores normalmente incluem crescimento exponencial de carreira, compensação monetária substancial, e ampla influência. Há bastante espaço para mais bons testadores.   
  
**Fechando uma lacuna**  
  
Embora todos concordem que um existe uma grande lacuna entre a pesquisa de teste de software e a prática da indústria, não há acordo quanto às razões, responsabilidades e possíveis soluções.Vamos usar o teste de cobertura como exemplo . Pergunte aos pesquisadores sobre isso, e eles estarão familiarizados com um espectro de critérios de teste de caixa branca com base no fluxo de controle ou fluxo de dados e elegantemente ordenados em uma hierarquia de subsunção. A essência do teste de software está na comportamento de amostragem sistemática, e o esforço da pesquisa está em encontrar meios eficazes para buscar essa sistematicidade - os muitos critérios de cobertura confiar na exploração de código para fornecer diferentes níveis de rigor.   
  
Mas as soluções de pesquisa propostas ainda exigem investimentos substanciais antes que possam ser posto a trabalhar. Protótipos de prova de conceito podem ser suficientes para demonstrar a ideia em um artigo, mas testes de cobertura, por exemplo, ainda precisa de recursos sofisticados ferramentas para instrumentação de código e monitoramento, bem como avaliação empírica de métricas relativas. Isso vale para qualquer campo de pesquisa. De um lado,os praticantes que estão cronicamente com falta de tempo ou recursos tendem a perceber a sistemática teste como um luxo.   
  
**Cuidado com os sábios**  
  
Não existe uma prática de software teste. Em vez disso, existem muitas práticas e comunidades de prática. O comportamento que funciona bem testando vídeo jogos na Electronic Arts cairiam separados ao testar o planejamento de voo software na Base Aérea de Eglin. Ainda, Eu vejo uma habilidade subjacente que todas as pessoas que se esforçam para ser grandes testadores devem desenvolve.   
  
Esta habilidade é mais conhecida como geral análise de sistemas ou sistemas gerais pensamento. Se mais testadores realmente lerem testes livros didáticos ou prestou atenção aos padrões, isso seria um grande problema. No entanto, a maioria dos testadores ignora a maioria dos o conselho que está sendo oferecido sobre práticas de teste, então o fato de que é principalmente conselhos ruins não estão fazendo muito prejuízo. Poucos testadores na indústria sabem sobre isso.   
  
A boa prática não pode ser legislada à distância, mais do que um médico pode diagnosticar a visão de um paciente despercebidas.   
  
**Investigação, Em vez de controlar**  
  
A prática de teste de software é evoluindo estranhamente. Sob esta visão alternativa, que compartilho, oteste de software é uma investigação empírica, conduzida para fornecer aos stakeholders informações de qualidade sobre o produto ou serviço em teste. Os desafios técnicos dos testes crescem não apenas porque os programas são maior, mas também porque esperamos que o software funcione corretamente mesmo quando o código é multithread, quando os dados são passados para frente e para trás em tempo real em sistemas distribuídos e quando o software é executado por mais e mais tempo vezes sem reinicialização. Não há muita orientação para esses esforços existe na engenharia de software atual padrões, pesquisas acadêmicas sobre testes ou as empresas de certificação de testadores que se tornaram tão populares em esta década.   
  
Para testes de software para atender às necessidades da indústria,isso terá que mudança.