SpiralDevelopment em Casos de Uso

Problema: desenvolver casos de uso num único passo é difícil, impede a incorporação de novos dados, podendo atrasar a descoberta de fatores de risco.

Solução: desenvolver os casos de uso num processo iterativo e em cada iteração aumentar progressivamente a precisão e objetividade dos casos de uso.

Descrever casos de uso iterativamente permite que facilmente voltemos atrás e refazê-los caso não estejam a funcionar, perdendo apenas parte do trabalho. Mais importante é o facto de conseguirmos identificar e resolver mais cedo problemas que apareçam devido à abordagem iterativa.

Os atrasos são caros. A recolha de requisitos é crítica para o sucesso do produto, mas é apenas uma parte do projeto. Os requisitos são suscetíveis de mudar como resultado da nossa análise, e ao examinar um requisito é provável descobrir informação sobre outros e muitas vezes descobrir que vários requisitos estão errados ou em falta.

O custo de erros de requisitos é alto e aumenta quanto mais tarde for descoberto. Embora seja importante terminar os casos de uso em tempo útil, é mais importante que toda a organização tenha a oportunidade de conhecer bem o sistema, pois vão suportar a maior parte do desenvolvimento. Assim:

Desenvolva casos de uso de uma maneira iterativa de ***breadth-first***, com cada iteração progressivamente aumentando a precisão e objetividade do conjunto de casos de uso.

Começar a trabalhar com ***BreadthBeforeDepth*** e pausar quando temos a lista dos atores e dos seus objetivos e trabalhar nessa lista por algum tempo. Usar a lista para estabelecer o plano do projeto, trabalho estimado, priorizar o valor dos casos de uso e ajudar a estabelecer equipas de desenvolvimento.

Continuando com ***BreadthBeforeDepth,*** escolher um subconjunto dos casos de uso para expandir e fazer uma nova pausa quando temos um conjunto principal de cenários de sucesso para avaliar a finalidade do sistema. Aproveitar a oportunidade neste momento para rever os casos de uso.

Finalmente, terminando ***BreadthBeforeDepth,*** revisitar a lista de casos de uso mais uma vez quando começar a trabalhar na extensão dos casos de uso, fazendo atenção pois novos casos de uso podem surgir. Por exemplo, ao descrever casos de uso para sistemas simples de multibanco, as pessoas normalmente não pensam no que acontece quando existe uma falha de comunicação entre o ATM e o host. Obviamente quando a falha de comunicação deixar de existir, o ATM terá trabalho específico para fazer. Tratar desta condição do sistema obriga à criação de um novo caso de uso, por exemplo "Restabelecimento de comunicações" que não existia previamente.

SpiralDevelopment interage com **BreadthBeforeDepth.** Imagine construir todos os casos de uso **BreadthBeforeDepth** sem qualquer plano de rever os casos de uso pelo caminho. A experiência mostra que enquanto as pessoas contribuem para o caso de uso, descobrem situação para as quais elas têm que descrever casos de uso, descobrem novas semelhanças entre eles e encontram melhores maneiras para estruturar o conjunto de casos. **SpiralDevelopment** recomenda à equipa que interrompam e reagrupem o trabalho e indica quando interromper para rever o trabalho.

A chave para o sucesso do desenvolvimento iterativo, do qual o **SpiralDevelopment** é um exemplo, é saber quando é tempo para parar. Pare assim que tenha a certeza que os casos de uso são bons o suficiente para ir ao encontro das necessidades dos stakeholders, de forma a evitar a lei dos rendimentos decrescentes. **QuittingTime** fornece um conjunto de critérios que podemos usar para determinar esse momento.