O QUE É UM DESIGN ORIENTADO A DOMÍNIO

Os processos de negócios automatizados ou os problemas do mundo real do software onde o desenvolvimento de software é frequentemente aplicado são o domínio do software. Devemos entender desde o início que o software é originado e profundamente relacionado a este domínio.

Para desenvolver um software não podemos apenas sentar e começar a digitar o código, isso pode até funcionar para casos triviais, mas não para criar um software complexo. É necessário planejar, escrever especificações, criar modelos, testar e então criar todas as partes do código e reuni-las.

Para criar um bom software você precisa entender sobre aquilo que vai ser programado, ao criar um software de sistema bancário você precisa entender o *domínio* do banco, mas quem é que conhece cada detalhe do banco? Os bancários é claro. É aqui que devemos sempre começar: o domínio. OOOOOOOOOOOOOOOOOo

Todo o propósito do software é se encaixar harmoniosamente com o domínio para o qual foi criado, a melhor forma é incorporar no software os conceitos básicos e elementos do domínio, para realizar precisamente as relações entre eles.

Para começar precisamos criar uma abstração do domínio. Essa abstração é um modelo do domínio, De acordo com Eric Evans, um modelo de domínio não é um diagrama particular, é a ideia de que o diagrama se destina a transmitir. Não é apenas o conhecimento na cabeça de um especialista no domínio, isto é uma abstração rigorosamente organizada e seletiva daquele conhecimento.

O modelo do domínio é muito cheio de detalhes para lembrarmos de tudo durante o processo de design, por isso precisamos organizar a informação, sistematizar, dividir em pedaços menores, para agrupar esses pedaços em módulos lógicos, e lidar posteriormente com cada um deles. Um domínio contém informações demais para se incluir no modelo, então deve ser incluído somente o essencial.

O modelo é a essência do software, mas precisamos criar maneiras para expressá-lo, comunicá-lo aos outros. Para isso pode-se criar um gráfico, escrever a visão sobre o modelo, criar uma linguagem para comunicar problemas específicos sobre o domínio. OOOOOOOOOO

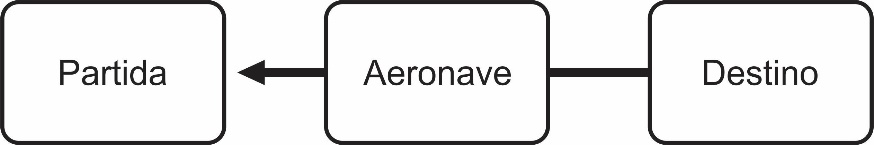
Quando o modelo está pronto, precisamos começar a criar o design do código. O design do código trabalha nos detalhes, um erro nele é mais fácil de corrigir do que no design do software. Boas técnicas de codificação ajudam a criar um código limpo e sustentável.

Existem diferentes abordagens para o design de software. Uma é o método de design de cascata. Os especialistas passam um conjunto de requisitos para os analistas, os analistas criam um modelo e passam para os desenvolvedores que começam a codificar. Mas essa técnica tem falhas e limites. O principal é que como é uma técnica unidirecional, não há feedback entre especialistas, analistas e desenvolvedores.

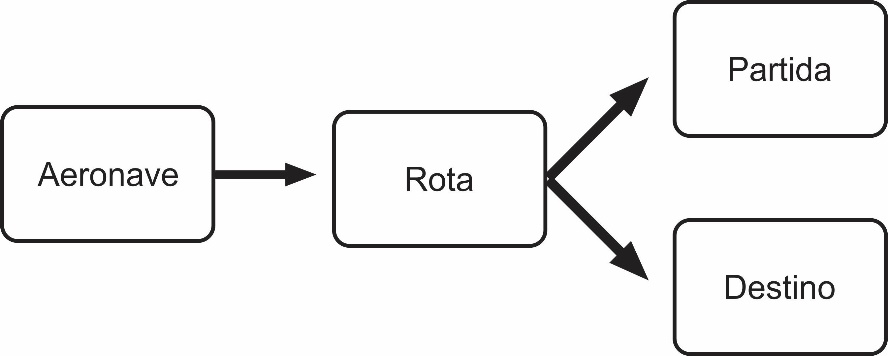
Outra abordagem é a metodologia Agile, como a Programação Extrema. Ela tenta criar um modelo adiantado que cobre todos os aspectos do domínio, mas isso pode resultar em efeitos colaterais negativos. Outro problema que a Agile tenta resolver é o chamado “Paralisia analítica”, com os membros da equipe com tanto medo de fazer decisões de projeto que eles não fazem progresso algum. A equipe tem que empregar uma grande quantidade de flexibilidade de implementação e através do desenvolvimento iterativo com a participação contínua das partes interessadas no negócio e muita refatoração a equipe de desenvolvimento começa a aprender mais sobre domínio do cliente e pode produzir melhor um software que atenda às necessidades dos clientes.

**Construindo Domínio do Conhecimento**

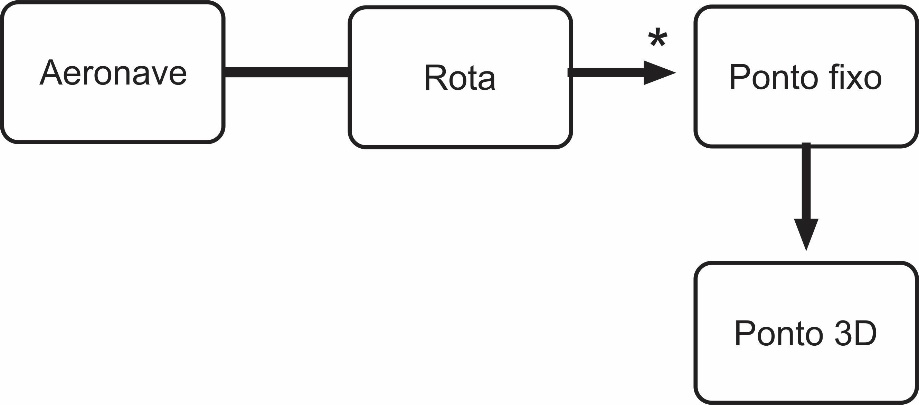
Vamos considerar o exemplo de um projeto de sistema de controle de voo e como o conhecimento de domínio pode ser construído. Ao conversar com um controlador de voo, vocês concordam que cada aeronave tem uma partida e um destino, conforme mostrado na figura abaixo.



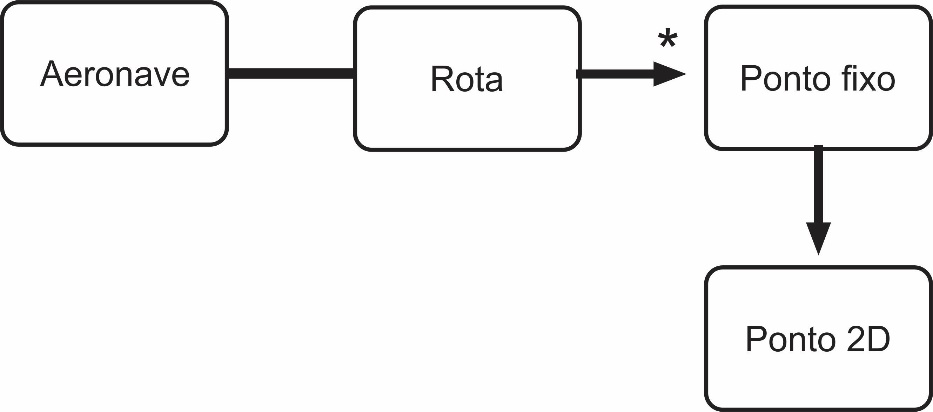
O controlador diz que cada avião recebe um plano de voo que deveria descrever toda a viagem aérea. Os aviões seguem então uma rota. É óbvio que o ponto de partida e destino da aeronave são também o ponto inicial e final da rota. Então, ao invés de associar a aeronave com os pontos de partida e destino, parece mais natural associá-lo a uma rota, que por sua vez é associado à partida e ao destino correspondente.



Conversando com o controlador sobre os percursos dos aviões, você descobre que, na verdade, a rota é composta de pequenos segmentos, que juntos constituem algum tipo de linha “torta” da partida ao destino. A linha deve passar através de pontos fixos predeterminados. Então, uma rota pode ser considerada como uma série de correções consecutivas. As mudanças resultantes com base nessas descobertas são:



O controlador vê a rota como a projeção na terra do voo do avião. As correções são apenas pontos na superfície da Terra unicamente determinada pela sua latitude e longitude. Então, o diagrama correto é:



Conversando com o controlador você vai descobrindo conceitos, estes conceitos podem ser desorganizados, mas ajudam a entender o domínio. E assim você e o controlador/especialista esboçam uma visão do domínio, um modelo do domínio.

Esta é uma parte importante do design. Geralmente existem longas discussões entre arquitetos de software ou desenvolvedores e os especialistas em domínio, o que resulta em um feedback, o que ajuda a criar um modelo melhor, com uma compreensão melhor do domínio. Os especialistas em software (arquitetos de software e desenvolvedores) e os especialistas do domínio, estão criando o modelo do domínio em conjunto, e o modelo é o lugar onde aquelas duas áreas de especialização se encontram. Isso pode gastar muito tempo do processo, mas é assim que deve ser, porque, no final, o objetivo do software é resolver os negócios/problemas em um domínio da vida real, por isso tem que se misturar perfeitamente com o domínio.