Nome: Gabriel Reis Lebron de Oliveira

Criteria for Modularization

O artigo *On the Criteria to be Used in Decomposing Systems into Modules*, escrito por David Parnas em 1972, é um dos textos mais influentes da engenharia de software, especialmente no tema de modularização. Nele, Parnas discute como dividir um sistema em módulos de forma a torná-lo mais compreensível, flexível e fácil de manter. O ponto central do trabalho é mostrar que não basta simplesmente dividir o sistema em partes; é preciso adotar critérios corretos para que a modularização realmente traga benefícios.

Logo no início, o autor explica quais são as vantagens esperadas da programação modular: reduzir o tempo de desenvolvimento (já que equipes podem trabalhar em módulos diferentes ao mesmo tempo), aumentar a flexibilidade (permitindo mudanças em um módulo sem afetar os outros) e facilitar a compreensão do sistema (uma vez que cada módulo pode ser estudado de forma isolada). Esses objetivos parecem óbvios hoje em dia, mas, na época, eram bastante inovadores, já que os sistemas eram pensados de forma muito mais linear e acoplada.

Parnas apresenta dois exemplos de decomposição de um mesmo sistema, chamado de KWIC Index Production System (um sistema que gera índices circulares de palavras em linhas de texto). No primeiro exemplo, a divisão segue um critério mais tradicional, no qual cada etapa do processamento corresponde a um módulo: entrada de dados, geração de shifts, ordenação alfabética, saída de dados, entre outros. Esse tipo de modularização é intuitivo e bastante comum, mas tem o problema de deixar decisões de design espalhadas por vários módulos. Isso significa que qualquer mudança no formato dos dados ou no modo de armazenamento, por exemplo, exigiria alterações em quase todos os módulos, dificultando a manutenção.

No segundo exemplo, a decomposição segue um critério mais avançado, baseado no conceito de information hiding (ocultamento de informações). Nesse caso, cada módulo é responsável por esconder certas decisões de design dos demais. Por exemplo, o módulo de armazenamento de linhas é o único que conhece os detalhes de como as linhas são guardadas na memória. Se essa estrutura mudar, só esse módulo precisa ser modificado, sem impacto direto nos outros. Esse modelo torna o sistema mais robusto e preparado para mudanças futuras.

A comparação entre as duas abordagens mostra que modularizar de acordo com as etapas do processo (como em um fluxograma) não é suficiente para lidar com sistemas grandes e complexos. Para Parnas, o critério correto de modularização deve estar baseado em ocultar decisões de projeto que provavelmente vão mudar.

Assim, cada módulo deve ser desenhado de forma a proteger os outros dessas mudanças, garantindo que o impacto seja sempre o menor possível.

O autor também comenta que a implementação da segunda forma de modularização pode parecer menos eficiente em termos de desempenho, já que exige mais chamadas entre módulos. No entanto, essa perda pode ser compensada por técnicas de implementação mais avançadas, e os ganhos em flexibilidade e manutenibilidade superam a preocupação inicial com eficiência. Essa visão é bastante prática, pois reconhece que um sistema não deve ser pensado apenas para rodar, mas também para evoluir e ser compreendido no longo prazo.

Outro aspecto importante do artigo é que ele antecipa conceitos que hoje são fundamentais no desenvolvimento de software, como separação de responsabilidades, baixo acoplamento e alto encapsulamento. A noção de que módulos devem esconder suas decisões internas é um princípio que ainda guia arquiteturas modernas, como a orientação a objetos, os microsserviços e o design de APIs.

Em resumo, o artigo de Parnas defende que a modularização deve ser feita com base em critérios sólidos, priorizando o ocultamento de informações críticas de design, e não simplesmente dividindo o sistema em etapas de processamento. Essa visão ajuda a criar sistemas mais compreensíveis, fáceis de modificar e que possam evoluir sem grandes impactos. Mesmo escrito há mais de 50 anos, o texto continua extremamente atual e útil para estudantes e profissionais da área.

Para nós, universitários, a principal lição é clara: ao desenvolver software, não devemos pensar apenas em “fazer funcionar”, mas também em como o sistema será mantido e modificado no futuro. O artigo mostra que boas decisões de modularização desde o início podem evitar muitos problemas mais tarde, tornando-se um aprendizado essencial para quem quer se preparar para o mercado de tecnologia.