On the Criteria to Be Used in Decomposing Systems into Modules (David L. Parnas, 1972)

Esse artigo é considerado um marco na Engenharia de Software porque propõe uma nova forma de pensar a modularização de sistemas. Até então, o mais comum era dividir o software em módulos seguindo o fluxo de execução (como se fosse um fluxograma transformado em código). Parnas critica essa prática e mostra que ela gera sistemas rígidos, difíceis de manter e pouco flexíveis.

Ele traz como exemplo um sistema de geração de índices KWIC (Keyword in Context) e mostra duas formas de modularizar. Na primeira (a convencional), cada etapa do processamento é um módulo: entrada, shift circular, ordenação, saída, etc. Já na segunda (a inovadora), os módulos são definidos com base no que ele chama de information hiding (esconder decisões de projeto que podem mudar no futuro). Assim, ao invés de expor detalhes de implementação, cada módulo encapsula uma responsabilidade e só revela o mínimo necessário para interação.

A comparação é bem clara: na primeira abordagem, qualquer alteração em detalhes de armazenamento ou formatação exige mudanças em quase todos os módulos. Na segunda, essas mudanças ficam restritas ao módulo responsável, o que aumenta a manutenibilidade, compreensão e independência do desenvolvimento.

O ponto central do texto é justamente esse: um bom critério de modularização não deve se basear em etapas do processamento, mas sim em proteger decisões suscetíveis a mudanças. Isso se conecta diretamente ao que estudamos em Engenharia de Software: princípios como baixo acoplamento, alta coesão e abstração. É praticamente a base do que depois inspirou conceitos modernos como orientação a objetos e design patterns.

O autor também reconhece que a abordagem dele pode parecer menos eficiente em termos de execução, mas defende que com técnicas adequadas (como otimizações no compilador/assembler) é possível equilibrar eficiência e modularidade.

No fim, a conclusão de Parnas é visionária: projetar sistemas pensando na evolução futura é mais importante do que pensar apenas na execução imediata. Isso mostra como a Engenharia de Software não é só codar, mas principalmente tomar boas decisões de projeto que facilitam manutenção e extensão do sistema ao longo do tempo.