On the Criteria to Be Used in Decomposing Systems into Modules

O artigo On the Criteria to Be Used in Decomposing Systems into Modules, de David Parnas, é um daqueles textos clássicos que mudaram a forma de pensar sobre software. Enquanto a maioria dos programadores da época seguia a lógica de dividir o sistema em etapas de processamento (como num fluxograma), Parnas propôs algo diferente: não modularize pelo fluxo de execução, modularize pelo que pode mudar.

Essa ideia simples abriu caminho para o famoso conceito de ocultamento de informações (*information hiding*). A visão dele é que cada módulo deve esconder uma decisão de projeto que pode se alterar no futuro — por exemplo, como os dados são armazenados, como a ordenação é feita ou como uma estrutura é representada. Assim, quando chegar a hora de mudar, o impacto fica restrito a um módulo, sem quebrar o resto do sistema.

O autor apresenta um exemplo didático: um sistema de geração de índice (KWIC). Primeiro, mostra a forma tradicional de dividir em módulos — entrada, circular shift, ordenação, saída — e depois uma abordagem alternativa, em que os módulos não seguem etapas do fluxo, mas encapsulam decisões de projeto, como armazenamento de linhas, gerador de shifts e ordenação. A diferença é gritante: no modelo tradicional, uma mudança simples no formato dos dados pode obrigar a alterar todo o sistema; já no modelo de Parnas, a mudança se restringe ao módulo responsável.

De forma prática, Parnas aponta três grandes benefícios dessa forma de modularização:

- **Flexibilidade**: alterações são localizadas e menos custosas.
- **Compreensibilidade**: é possível entender o sistema por partes, sem precisar conhecer tudo de uma vez.
- **Desenvolvimento independente**: equipes podem trabalhar em paralelo, sem ficarem travadas por detalhes técnicos umas das outras.

O mais interessante é que Parnas já reconhecia um desafio que ecoa até hoje: modularizar desse jeito pode ser menos eficiente se for implementado ingenuamente. Ele sugere que precisamos de ferramentas e técnicas que escondam essa complexidade, para não sacrificar desempenho.

No fim, o artigo mostra que modularizar não é só “dividir o código em caixinhas”, mas uma decisão estratégica sobre **onde esconder as incertezas**. A lição que fica é direta: sistemas grandes sempre mudam, então a modularização deve ser pensada desde cedo para tornar essas mudanças menos dolorosas.