Artigo 4 - On the Criteria To Be Used in Decomposing Systems into Modules

Nome do estudante: João Victor Filardi Souza Pinto

O artigo de David Parnas, considerado um marco na engenharia de software, apresenta a modularização como um princípio essencial para melhorar a flexibilidade, a compreensibilidade e a manutenibilidade dos sistemas. A ideia central é que a forma como dividimos um sistema em módulos não deve ser arbitrária, mas sim guiada por critérios sólidos que garantam que mudanças futuras possam ser realizadas de maneira localizada, sem comprometer o funcionamento do sistema como um todo. Esse pensamento se mostra extremamente atual, mesmo décadas após a publicação do artigo.

Parnas começa destacando que modularizar não é apenas uma questão de dividir o sistema em partes menores. O valor dessa prática está em reduzir a interdependência entre os módulos, de forma que cada um esconda certas decisões de projeto que podem mudar ao longo do tempo. Esse conceito, que mais tarde se consolidou como o princípio do *information hiding*, é um dos pontos mais fortes do artigo. Diferentemente das abordagens mais comuns da época, baseadas em fluxogramas que dividiam o sistema segundo as etapas de processamento, Parnas propõe que cada módulo seja responsável por encapsular decisões de design que são voláteis ou de difícil definição.

O exemplo prático utilizado por ele é o sistema de geração de índice KWIC (Key Word in Context). Parnas apresenta duas decomposições possíveis: a primeira, mais tradicional, divide o sistema em módulos de acordo com o fluxo de entrada, processamento e saída; já a segunda, considerada por ele mais eficiente, cria módulos que escondem detalhes como a forma de armazenamento das linhas, os métodos de rotação e a ordenação alfabética. O ponto crucial é que, na segunda decomposição, mudanças como alteração no formato de entrada ou na estratégia de armazenamento ficariam restritas a um único módulo, sem necessidade de alterar os demais. Isso reduz a complexidade do sistema, melhora a independência entre equipes de desenvolvimento e facilita a evolução futura do software.

Essa visão vai além da simples organização do código. Parnas argumenta que, quando os módulos são definidos a partir de passos do processamento, qualquer pequena mudança em requisitos pode gerar alterações em diversos pontos do sistema, aumentando custos e riscos. Já ao adotar a ocultação de informações como critério, os módulos se tornam mais coesos e menos acoplados, permitindo que sejam entendidos, testados e evoluídos de forma independente. Esse raciocínio conecta-se diretamente ao que estudamos em sala sobre coesão e acoplamento, conceitos que ainda hoje guiam boas práticas de design de software.

Do ponto de vista de aplicabilidade prática no mercado, os ensinamentos de Parnas são bastante claros. Imagine, por exemplo, uma empresa de tecnologia que desenvolve um sistema de gestão empresarial utilizado por clientes de diferentes portes. Ao modularizar o

sistema com base no fluxo de operações, qualquer alteração no formato de dados ou em regras de negócio poderia exigir modificações em múltiplos módulos interdependentes, tornando a manutenção cara e arriscada. Por outro lado, ao estruturar os módulos de modo a esconder decisões suscetíveis a mudanças, como a forma de persistência dos dados ou as regras de cálculo de impostos, a empresa poderia atualizar ou substituir partes do sistema sem impactar o restante. Esse princípio está diretamente ligado ao desenvolvimento de arquiteturas modernas, como microservices, que seguem justamente a ideia de dividir responsabilidades e encapsular detalhes.

Em resumo, o artigo de Parnas defende que o critério de modularização deve ir além da intuição ou da ordem lógica do processamento. Ele deve ser pautado pela identificação de decisões de projeto voláteis e pela ocultação dessas decisões dentro de módulos específicos. Essa abordagem promove maior flexibilidade, independência de desenvolvimento e facilidade de manutenção. Apesar de escrito na década de 1970, o texto continua extremamente relevante, pois antecipa práticas que ainda hoje orientam o design de sistemas complexos e sustentáveis no mercado de software.