O artigo On the Criteria To Be Used in Decomposing Systems into Modules, escrito por David L. Parnas em 1972, é um dos trabalhos mais influentes da engenharia de software. Sua relevância atravessa décadas, pois aborda um tema fundamental para o desenvolvimento de sistemas robustos, escaláveis e de fácil manutenção: a modularização. Parnas questiona os critérios tradicionalmente utilizados para dividir um sistema em módulos e propõe uma abordagem inovadora baseada no conceito de information hiding (ocultação de informações). Ao longo do artigo, o autor demonstra como uma escolha inadequada de critérios pode comprometer a flexibilidade, compreensibilidade e evolução do software, além de aumentar custos e riscos de manutenção.

Desde o início, Parnas deixa claro que modularização não deve ser entendida apenas como uma separação física de código ou criação de sub-rotinas, mas sim como uma divisão lógica que facilite a independência entre as partes do sistema. O objetivo não é apenas dividir o trabalho entre equipes, mas garantir que futuras mudanças possam ser feitas de forma localizada, sem impactar outros módulos. Para fundamentar sua análise, o autor apresenta dois exemplos de decomposição aplicados a um sistema chamado KWIC (Key Word in Context), que gera todas as permutações circulares de uma lista de linhas e as organiza em ordem alfabética.

Na primeira decomposição, Parnas descreve um modelo convencional, no qual os módulos são definidos com base nas etapas do processamento: entrada, circular shift, ordenação, saída e controle. Essa estrutura reflete um fluxograma clássico, onde cada módulo corresponde a uma fase da execução do programa. Apesar de parecer lógica e intuitiva, essa abordagem cria dependências fortes entre módulos, já que mudanças no formato de armazenamento ou em estruturas internas exigem alterações em diversos pontos do sistema. Por exemplo, se o método de empacotamento de caracteres ou a forma de indexação mudar, quase todos os módulos precisarão ser adaptados, elevando o custo e a complexidade de manutenção.

Já na segunda decomposição, Parnas adota um critério baseado em information hiding, onde cada módulo é responsável por ocultar decisões que podem mudar, como representações internas, formatos de dados ou algoritmos. Em vez de organizar os módulos pelas etapas do processo, eles são definidos pelo tipo de conhecimento que isolam. Por exemplo, existe um módulo dedicado ao armazenamento de linhas, outro para o gerenciamento dos deslocamentos circulares, outro para a ordenação e assim por diante. A grande vantagem desse modelo é a redução do acoplamento: se a estrutura de armazenamento mudar, apenas o módulo específico será alterado, sem impacto direto nos demais. Essa abordagem antecipa conceitos modernos, como encapsulamento e baixo acoplamento, essenciais para sistemas que precisam evoluir de forma segura e sustentável.

Parnas também discute os benefícios esperados da modularização, como redução do tempo de desenvolvimento, facilidade de compreensão e flexibilidade para mudanças, ressaltando que esses ganhos só ocorrem se os critérios corretos forem aplicados. Um ponto importante levantado no artigo é que os fluxogramas, comuns na época, não são critérios adequados para definir módulos, pois se baseiam na ordem temporal da execução e não na estabilidade das decisões de projeto. Em contrapartida, o uso do information hiding garante maior independência entre módulos, permitindo desenvolvimento paralelo e minimizando retrabalho em caso de alterações.

Contudo, o autor reconhece desafios na aplicação dessa técnica, especialmente relacionados ao desempenho. Em arquiteturas da época, a implementação de módulos como funções independentes poderia gerar sobrecarga devido ao número elevado de chamadas. Para mitigar esse problema, Parnas sugere que compiladores ou montadores possam otimizar essas chamadas, preservando a separação lógica sem comprometer a eficiência. Essa discussão mostra que, mesmo propondo ideias avançadas para a época, Parnas estava atento às limitações práticas da tecnologia disponível.

A aplicabilidade desse artigo no mercado de trabalho atual é imensa. A filosofia de Parnas é a base de diversas práticas modernas, como programação orientada a objetos, princípios SOLID, design patterns e arquitetura de microsserviços. Em ambientes corporativos dinâmicos, onde requisitos mudam constantemente e sistemas precisam evoluir rapidamente, ocultar decisões voláteis é fundamental para reduzir custos e riscos. Por exemplo, em arquiteturas orientadas a microsserviços, cada serviço é projetado para ser autônomo e encapsular suas regras e dados, permitindo mudanças internas sem impacto no restante da aplicação — exatamente o que Parnas defendia há mais de 50 anos.

Além disso, os conceitos de Parnas são essenciais para o desenvolvimento ágil e DevOps, pois facilitam a integração contínua e a entrega frequente de software. Quando módulos têm interfaces bem definidas e ocultam suas implementações internas, os times podem trabalhar de forma paralela e implantar atualizações de maneira incremental, reduzindo riscos de regressão. Outro exemplo prático está em APIs modernas: quando uma empresa define contratos estáveis para suas APIs e esconde detalhes internos, ela garante compatibilidade mesmo quando a lógica interna evolui — mais uma aplicação direta do information hiding.

Em um cenário onde novas tecnologias surgem constantemente — como cloud computing, contêineres e inteligência artificial —, a mensagem de Parnas continua extremamente atual: mudanças são inevitáveis, e sistemas bem projetados são aqueles que isolam essas mudanças, evitando que se espalhem pelo código. Organizações que ignoram esse princípio correm o risco de criar sistemas frágeis, difíceis de escalar e caros de manter. Portanto, adotar os critérios corretos para

modularização não é apenas uma boa prática técnica, mas uma estratégia de negócio.

Em resumo, o artigo de Parnas vai muito além de um guia técnico: ele propõe uma filosofia de design orientada à previsibilidade e à resiliência. Sua lição central é clara: modularização não deve seguir a sequência do processamento, mas sim proteger o sistema contra as incertezas do futuro, isolando decisões suscetíveis a mudanças. Essa visão é indispensável para qualquer organização que deseje construir software durável, adaptável e competitivo em um mercado que evolui rapidamente e exige inovação contínua.