O artigo apresenta uma proposta prática para resolver um problema recorrente em equipes de desenvolvimento de software: a perda do histórico das razões que motivaram determinadas escolhas arquiteturais. Segundo o autor, em muitos projetos, principalmente aqueles que seguem metodologias ágeis, há uma tendência de evitar documentação extensa e burocrática. Contudo, a ausência de registros mínimos sobre decisões importantes leva a dois riscos. O primeiro é aceitar cegamente uma decisão herdada sem compreender seus pressupostos. O segundo é alterar uma decisão sem avaliar seus efeitos colaterais, o que pode comprometer a qualidade do sistema. Para evitar esses cenários, Nygard propõe o uso dos chamados Architecture Decision Records (ADRs), pequenos documentos voltados para registrar cada decisão de caráter arquitetural que realmente impacta o projeto.

O ADR é pensado para ser leve e objetivo. Cada documento deve ter um título curto e descritivo, um contexto que explique quais fatores estavam em jogo no momento da decisão, a descrição clara da decisão adotada, o status indicando se ela foi proposta, aceita, descontinuada ou substituída, e as consequências, que podem ser positivas, negativas ou neutras. A ideia não é escrever longos relatórios, mas manter registros curtos, de uma ou duas páginas, que permitam compreender rapidamente o raciocínio da equipe em cada escolha. O autor recomenda inclusive que esses arquivos sejam mantidos em formato simples, como Markdown, dentro do repositório do código, lado a lado com a evolução do software. Dessa forma, eles permanecem acessíveis, versionados e fáceis de revisar.

Outro ponto importante levantado no artigo é a manutenção do histórico. Quando uma decisão deixa de ser válida, ela não deve ser apagada, mas sim marcada como substituída. Assim, futuras gerações de desenvolvedores conseguem entender por que a equipe decidiu de um jeito no passado e por que posteriormente optou por outro caminho. Essa rastreabilidade é especialmente valiosa em sistemas que evoluem ao longo de anos e que passam pelas mãos de diferentes profissionais. Com os ADRs, torna-se possível olhar para trás e compreender não apenas o que foi feito, mas sobretudo por que foi feito.

O autor também aponta benefícios adicionais obtidos pela prática. Em suas experiências, equipes que adotaram ADRs relataram maior clareza e confiança ao lidar com decisões antigas, além de uma comunicação mais transparente entre membros novos e antigos do time. Isso porque o registro escrito elimina a dependência de memória ou de conversas informais, que frequentemente se perdem. Em contrapartida, Nygard reconhece que a adoção exige disciplina. Há o risco de que a equipe não crie os registros de forma consistente, o que tornaria a iniciativa ineficaz. Também é preciso avaliar o que de fato merece ser registrado, já que nem toda decisão de programação é uma decisão arquitetural. Mesmo assim, o balanço final é positivo: um esforço pequeno de documentação evita problemas sérios de manutenção e garante maior solidez na evolução do software.

Em síntese, o artigo defende que documentar decisões arquiteturais não precisa significar burocracia ou excesso de papelada. Com registros curtos, claros e organizados junto ao código, é possível preservar a memória do projeto, dar transparência às escolhas, facilitar revisões futuras e apoiar a comunicação entre equipes. Trata-se de uma prática leve, mas que gera um impacto significativo na qualidade do desenvolvimento.

Para ilustrar a aplicação prática da proposta, podemos imaginar um cenário em que uma equipe de e-commerce precisa definir o mecanismo de autenticação de usuários. Após analisar diferentes opções, como sessões tradicionais de servidor, integração com OAuth2 e uso de tokens JWT, a equipe decide adotar JWT por considerar que o sistema exige escalabilidade horizontal e comunicação com clientes móveis. Essa decisão é registrada em um ADR com a descrição do contexto, os fatores considerados, a escolha pelo JWT e as consequências esperadas, como a facilidade de validar tokens em microserviços e o risco de precisar lidar com revogação de chaves. Com o tempo, se houver necessidade de migrar para outra tecnologia, por exemplo para OAuth2, um novo ADR é criado explicando a mudança e marcando o anterior como substituído. Dessa maneira, tanto a decisão inicial quanto a posterior ficam registradas, permitindo que qualquer novo desenvolvedor compreenda a lógica de cada momento. Esse exemplo mostra como o modelo de Nygard pode ser integrado ao dia a dia de um projeto, garantindo clareza, continuidade e evolução arquitetural consistente.