

O desenvolvimento de software sempre foi uma das disciplinas mais importantes da Tecnologia da Informação, mas sua relevância ganhou uma dimensão ainda maior conforme produtos, serviços e processos empresariais passaram a depender diretamente de aplicações digitais.

Durante as últimas décadas, testemunhamos uma evolução extraordinária nas linguagens de programação, arquiteturas, metodologias, ferramentas e modelos de entrega. O desenvolvimento tradicional passou a conviver com abordagens ágeis, Cloud Computing modificou a maneira como aplicações são disponibilizadas e práticas de DevOps e DevSecOps transformaram os mecanismos de integração, testes, implantação e operação.

Mais recentemente, a Artificial Intelligence introduziu outra transformação significativa, permitindo que atividades anteriormente realizadas exclusivamente por desenvolvedores passassem a contar com ferramentas capazes de produzir código, elaborar testes, identificar problemas, documentar soluções e executar sequências complexas de tarefas através de agentes especializados.

Entretanto, apesar de todos esses avanços, alguns dos principais desafios enfrentados pelas organizações continuam surpreendentemente semelhantes àqueles observados há muitos anos.

Projetos ainda são iniciados com requisitos insuficientemente compreendidos. Arquiteturas continuam sendo selecionadas sem considerar adequadamente suas consequências futuras. Aplicações entram em produção apresentando limitações de desempenho, segurança e manutenção. Metodologias são implementadas de maneira burocrática, enquanto equipes precisam administrar prioridades contraditórias e dependências organizacionais excessivas.

Essa aparente contradição merece atenção porque demonstra que desenvolver software não pode ser compreendido exclusivamente como uma atividade técnica.

Produzir soluções empresariais exige integrar tecnologia, processos, pessoas, arquitetura, gestão, governança e conhecimento do negócio. A qualidade do resultado depende tanto das decisões tomadas antes da primeira linha de código quanto daquelas realizadas durante a implementação e ao longo de toda a vida útil da aplicação.

Em ambientes empresariais complexos, especialmente naqueles sujeitos a exigências regulatórias e elevados requisitos de disponibilidade, essa realidade se torna ainda mais evidente. Uma aplicação não deveria ser considerada bem-sucedida simplesmente porque suas funcionalidades foram implementadas. Ela também precisa apresentar segurança, escalabilidade, resiliência, eficiência econômica e capacidade de evolução.

É interessante perceber como muitas organizações investem significativamente na adoção de novas ferramentas e metodologias, mas dedicam relativamente menos atenção à construção das capacidades organizacionais necessárias para utilizá-las adequadamente.

Uma ferramenta sofisticada não elimina requisitos mal formulados. Uma arquitetura moderna não compensa automaticamente a ausência de governança. Uma metodologia ágil não resolve prioridades contraditórias. Da mesma forma, AI pode aumentar a velocidade de produção de código sem necessariamente melhorar a qualidade daquilo que está sendo desenvolvido.

Foi justamente por isso que considerei interessante revisitar uma matéria publicada pela CIO Online em 2023, que apresenta sete erros recorrentes capazes de comprometer projetos e produtos tecnológicos.

A reflexão continua atual, mas merece ser ampliada diante das transformações ocorridas desde sua publicação, especialmente com a consolidação de Platform Engineering, Software Supply Chain Security, AI-Assisted Development e novas modalidades de automação.

Talvez a pergunta mais relevante não seja simplesmente como evitar sete erros conhecidos, mas por que esses problemas continuam acontecendo mesmo em organizações que possuem profissionais qualificados, investimentos significativos e acesso a tecnologias cada vez mais sofisticadas.

A resposta provavelmente está menos nas ferramentas isoladas e mais na maneira como as diferentes competências da engenharia de software são organizadas, administradas e integradas para produzir resultados sustentáveis.

Os sete erros do desenvolvimento de software segundo a CIO Online

Em outubro de 2023, Peter Wayner publicou na CIO Online o artigo “7 sins of software development”, explorando decisões e comportamentos capazes de comprometer o trabalho de equipes de desenvolvimento, mesmo quando existem profissionais tecnicamente competentes e recursos aparentemente suficientes.

A matéria original pode ser consultada em:

https://www.cio.com/article/654284/7-sins-of-software-development.html

O aspecto que considero especialmente interessante nessa abordagem é que os problemas apresentados não estão concentrados exclusivamente na programação. Eles envolvem decisões relacionadas à metodologia, arquitetura, gestão de dados, fornecedores, qualidade e planejamento, demonstrando como as dificuldades enfrentadas no desenvolvimento frequentemente possuem causas que transcendem a competência técnica dos programadores.

Os sete erros destacados são:

1. **Escolher a metodologia inadequada**, impondo práticas que não correspondem às características do projeto, à realidade organizacional ou às competências da equipe.

2. **Ignorar a escalabilidade**, desenvolvendo aplicações incapazes de acompanhar adequadamente o crescimento de usuários, transações e volumes de dados.
3. **Sucumbir às tendências tecnológicas**, adotando linguagens, frameworks ou plataformas simplesmente por representarem novidades de mercado.
4. **Reter dados em excesso**, acumulando informações sem justificativa suficiente e ampliando custos, complexidade e riscos.
5. **Terceirizar o trabalho errado**, criando dependências inadequadas de fornecedores ou transferindo atividades que deveriam permanecer sob maior controle organizacional.
6. **Negligenciar os testes**, comprometendo a qualidade das soluções pela ausência de mecanismos adequados de validação.
7. **Subestimar o planejamento**, iniciando iniciativas sem compreender suficientemente objetivos, requisitos, dependências e restrições.

Embora esses problemas possam ser analisados individualmente, existe uma característica particularmente importante: eles frequentemente se reforçam mutuamente.

Uma metodologia inadequada pode comprometer planejamento e qualidade. Uma arquitetura mal concebida pode prejudicar escalabilidade e manutenção. Decisões precipitadas de sourcing podem criar dependências que dificultam a evolução das soluções. A ausência de testes pode transformar limitações relativamente simples em incidentes operacionais relevantes.

Da mesma forma, a pressão por entregas rápidas pode estimular a adoção indiscriminada de tecnologias, reduzir investimentos em qualidade e provocar o acúmulo de Technical Debt.

Portanto, não estamos tratando apenas de sete erros independentes, mas de diferentes manifestações de problemas relacionados à maturidade organizacional.

Desde 2023, a indústria também passou por transformações relevantes. Ferramentas de AI-Assisted Development ampliaram sua presença nas rotinas de engenharia, agentes passaram a executar atividades mais complexas e a segurança da cadeia de suprimentos de software ganhou importância adicional.

O relatório DORA de 2025 reforçou uma conclusão especialmente interessante: Artificial Intelligence tende a amplificar características previamente existentes nas organizações. Ambientes com boas práticas possuem melhores condições para

capturar benefícios, enquanto organizações com fragilidades estruturais podem aumentar a velocidade com que seus problemas são reproduzidos.

Também ocorreram mudanças nas referências de segurança. O OWASP Top 10 de 2025 passou a destacar explicitamente Software Supply Chain Failures, enquanto o NIST continuou evoluindo suas orientações para desenvolvimento seguro, mantendo o SSDF 1.1 como versão final publicada e disponibilizando uma proposta de revisão para a versão 1.2 em dezembro de 2025.

Essas transformações não invalidam os problemas apresentados pela CIO Online. Pelo contrário, tornam ainda mais importante compreender suas causas e estabelecer mecanismos capazes de impedir que continuem se repetindo.

Primeiro erro: escolher a metodologia inadequada

A escolha da metodologia de desenvolvimento continua sendo uma decisão importante, principalmente porque diferentes abordagens pressupõem maneiras distintas de organizar equipes, estabelecer responsabilidades, administrar prioridades e coordenar entregas.

Durante muitos anos, a discussão foi frequentemente apresentada como uma oposição entre metodologias tradicionais e ágeis, quase como se fosse necessário escolher definitivamente entre dois modelos incompatíveis.

De um lado, processos baseados em planejamento detalhado, fases sequenciais e mecanismos formais de aprovação. Do outro, ciclos curtos, feedback contínuo, colaboração e adaptação.

Com o amadurecimento da indústria, ficou evidente que essa oposição não representa adequadamente a diversidade de situações encontradas nas organizações.

Existem iniciativas nas quais requisitos são relativamente estáveis, dependências são conhecidas e alterações precisam seguir controles rigorosos. Existem outras nas quais o produto ainda está sendo descoberto, hipóteses precisam ser testadas e a capacidade de adaptação possui importância significativamente maior.

O problema começa quando a metodologia se transforma em objetivo, em vez de permanecer como instrumento para alcançar resultados.

Quando o framework substitui o pensamento crítico

Uma organização pode implementar Scrum, Kanban, SAFe ou qualquer outro modelo de trabalho e continuar enfrentando problemas de produtividade, qualidade, colaboração e previsibilidade.

Isso acontece frequentemente quando a implementação concentra seus esforços na reprodução de cerimônias, nomenclaturas e artefatos, sem necessariamente modificar os mecanismos que realmente determinam a capacidade de execução.

Equipes passam a realizar reuniões de planejamento, refinamento, retrospectivas e acompanhamento, mas continuam enfrentando dependências excessivas, processos lentos de aprovação e prioridades contraditórias.

Nesse cenário, a metodologia existe formalmente, mas os problemas estruturais permanecem praticamente inalterados.

O mesmo acontece quando modelos de escala são implementados adicionando níveis sucessivos de coordenação e governança sem que exista uma avaliação suficientemente clara sobre o valor produzido por essa complexidade adicional.

Agilidade não deveria ser confundida com quantidade de cerimônias, da mesma forma que governança não deveria ser confundida com quantidade de aprovações.

Ambas precisam contribuir para melhorar decisões, reduzir incertezas e aumentar a capacidade de entrega.

A metodologia precisa respeitar o contexto

Uma escolha madura deveria considerar a natureza do produto, a estabilidade dos requisitos, a criticidade operacional, o grau de autonomia da equipe, as dependências organizacionais e as exigências regulatórias.

Uma plataforma digital orientada à experimentação pode se beneficiar de ciclos curtos e mecanismos contínuos de descoberta.

Por outro lado, determinadas iniciativas de modernização de sistemas críticos podem exigir planejamento integrado, gestão formal de dependências e mecanismos específicos de validação.

Isso não significa que projetos críticos não possam utilizar práticas ágeis. Significa que diferentes requisitos precisam ser considerados na escolha das práticas mais adequadas.

Modelos híbridos podem funcionar muito bem quando utilizados de maneira consciente, combinando mecanismos de planejamento corporativo com práticas iterativas de execução.

O importante é evitar que o modelo operacional seja determinado exclusivamente pela preferência de uma liderança ou pela popularidade de determinado framework.

Cultura e modelo operacional precisam estar alinhados

A cultura organizacional também precisa ser considerada porque metodologias pressupõem determinados comportamentos.

Uma empresa que declara autonomia, mas mantém todas as decisões relevantes concentradas em níveis hierárquicos superiores, dificilmente produzirá agilidade real apenas modificando suas cerimônias.

Da mesma forma, uma organização que incentiva colaboração, mas avalia seus gestores exclusivamente por resultados funcionais, pode criar conflitos entre os comportamentos desejados e os mecanismos de reconhecimento existentes.

Desenvolvimento de software é uma atividade sociotécnica, na qual pessoas, ferramentas, processos e estruturas organizacionais interagem continuamente.

A metodologia adequada será aquela capaz de funcionar dentro dessa realidade ou de apoiar uma transformação deliberada e suficientemente sustentada.

A escolha correta envolve compreender como competências, responsabilidades e mecanismos de decisão precisam funcionar conjuntamente para que a organização consiga entregar soluções com qualidade e consistência.

Segundo erro: ignorar a escalabilidade

Escalabilidade continua sendo um dos atributos fundamentais de aplicações destinadas a atender volumes crescentes de usuários, transações, eventos e informações.

Uma solução pode funcionar perfeitamente durante testes realizados com poucos usuários e apresentar problemas significativos quando submetida às condições reais de produção.

Essas dificuldades não decorrem necessariamente de limitações de infraestrutura. Muitas vezes estão relacionadas à arquitetura, aos algoritmos utilizados, à modelagem

de dados, às dependências entre componentes ou à maneira como determinadas operações consomem recursos.

A adoção de Cloud Computing ampliou significativamente as possibilidades de provisionamento e elasticidade, mas não eliminou a necessidade de desenvolver aplicações capazes de utilizar adequadamente esses recursos.

É perfeitamente possível construir uma aplicação em Cloud que apresente problemas de escalabilidade, inclusive com custos crescentes e resultados insatisfatórios.

Escalabilidade começa na arquitetura

Uma das primeiras atividades deveria envolver a definição dos requisitos não funcionais da solução.

Qual será o volume esperado de usuários? Quantas transações precisarão ser processadas? Qual será o crescimento dos dados? Existem períodos de concentração de demanda? Quais níveis de latência são aceitáveis?

Essas perguntas precisam ser respondidas considerando tanto as necessidades atuais quanto cenários razoáveis de evolução.

Entretanto, existe uma distinção importante entre projetar para escalabilidade e construir antecipadamente uma arquitetura excessivamente complexa.

Uma aplicação destinada a atender algumas centenas de usuários internos dificilmente necessita da mesma arquitetura de uma plataforma global responsável por milhões de transações.

A busca indiscriminada por escalabilidade também pode gerar desperdício de recursos e complexidade desnecessária.

O objetivo deveria ser construir soluções proporcionais aos requisitos conhecidos, preservando caminhos viáveis para evolução.

Escalabilidade vertical, horizontal e arquitetural

Existem diferentes mecanismos para ampliar a capacidade de uma solução.

A escalabilidade vertical consiste em aumentar os recursos disponíveis para determinado componente, como processamento, memória ou armazenamento.

A escalabilidade horizontal envolve distribuir a carga entre múltiplas instâncias, frequentemente utilizando balanceamento, particionamento e processamento distribuído.

Existe ainda uma dimensão arquitetural, relacionada à maneira como os componentes foram estruturados para permitir crescimento sem que determinadas dependências se transformem em gargalos.

Uma arquitetura monolítica pode apresentar excelente escalabilidade quando suas características são adequadas ao workload.

Da mesma forma, uma solução baseada em microserviços pode apresentar desempenho insatisfatório quando o excesso de comunicação entre componentes introduz gargalos e aumenta a complexidade operacional.

Portanto, microserviços não deveriam ser tratados como sinônimo de escalabilidade.

São uma alternativa arquitetural que pode oferecer vantagens relevantes, mas que exige maturidade em integração, observability, automação e gestão operacional.

Escalabilidade também precisa ser economicamente sustentável

Outro aspecto frequentemente negligenciado está relacionado ao custo de crescer.

Uma solução pode ser tecnicamente capaz de processar volumes muito elevados e, simultaneamente, apresentar uma estrutura de custos economicamente insustentável.

Isso acontece quando cada novo usuário, transação ou evento exige uma quantidade desproporcional de recursos computacionais, armazenamento ou serviços especializados.

A discussão precisa combinar desempenho e eficiência financeira.

Conceitos como FinOps, unit economics e custo por transação ajudam a compreender se o crescimento da demanda será acompanhado por uma evolução sustentável da infraestrutura.

Uma arquitetura tecnicamente sofisticada não representa necessariamente uma arquitetura adequada.

A solução precisa atender aos requisitos de desempenho, segurança e resiliência dentro de parâmetros econômicos compatíveis com os objetivos empresariais.

A realização de testes de carga, estresse e capacidade continua sendo essencial, mas esses testes precisam representar comportamentos próximos da realidade operacional, incluindo integrações, bancos de dados e dependências externas.

Terceiro erro: sucumbir às tendências tecnológicas

O mercado de tecnologia possui uma característica bastante particular: praticamente todos os anos surgem novas ferramentas, linguagens, arquiteturas e plataformas apresentadas como potenciais transformadoras da maneira como desenvolvemos software.

Essa dinâmica é positiva porque demonstra a vitalidade da indústria e estimula a evolução contínua das capacidades tecnológicas.

O problema aparece quando inovação passa a ser confundida com adoção indiscriminada de novidades.

Uma tecnologia não deveria ser considerada superior simplesmente por ser recente, da mesma forma que uma solução consolidada não deveria ser considerada inadequada apenas por possuir muitos anos de existência.

O critério principal precisa continuar sendo sua capacidade de atender aos requisitos e produzir valor de maneira sustentável.

O custo oculto da diversidade tecnológica

Um dos impactos mais relevantes da adoção excessiva de novas tecnologias está relacionado à fragmentação do stack corporativo.

Cada nova linguagem, framework, banco de dados, ferramenta de integração ou plataforma exige conhecimento especializado, mecanismos de suporte, práticas de segurança, documentação, atualização e capacidade de manutenção.

Quando diferentes equipes escolhem tecnologias sem considerar a arquitetura empresarial, a organização pode desenvolver um ambiente extremamente heterogêneo.

Essa diversidade não é necessariamente negativa, mas precisa ser administrada.

O problema surge quando o custo de manter múltiplas soluções supera os benefícios proporcionados pela flexibilidade.

Equipes passam a enfrentar dificuldades para compartilhar profissionais, reutilizar componentes, estabelecer padrões e garantir continuidade operacional.

Em situações mais complexas, determinados sistemas tornam-se dependentes de poucas pessoas que conhecem tecnologias específicas, criando riscos de concentração

de conhecimento.

A inovação precisa ser acompanhada de critérios

A exploração de novas tecnologias deve ser incentivada, mas precisa acontecer dentro de um processo estruturado.

Uma abordagem razoável envolve identificar o problema, compreender as limitações das alternativas existentes, estabelecer critérios de avaliação e realizar experimentações controladas.

Proofs of Concept podem validar hipóteses técnicas, mas precisam possuir objetivos e critérios de sucesso claramente definidos.

Uma demonstração funcional não comprova automaticamente que determinada tecnologia possui maturidade suficiente para utilização empresarial.

Devem ser avaliados suporte, segurança, integração, observability, documentação, disponibilidade de profissionais, custos operacionais e perspectivas de evolução.

Também considero importante distinguir ambientes experimentais daqueles responsáveis por suportar workloads críticos.

Uma organização pode explorar tecnologias emergentes e, simultaneamente, preservar critérios rigorosos para plataformas utilizadas em operações de missão crítica.

Artificial Intelligence amplia esse desafio

A popularização de AI-Assisted Development introduziu uma nova categoria de decisões tecnológicas.

Ferramentas capazes de gerar código, sugerir alterações e executar atividades de engenharia oferecem oportunidades relevantes para ampliar produtividade.

Entretanto, sua adoção precisa considerar segurança, privacidade, propriedade intelectual, qualidade, rastreabilidade e dependência de fornecedores.

Existe também o risco de produzir volumes crescentes de código sem desenvolver uma compreensão proporcional sobre sua arquitetura, funcionamento e consequências operacionais.

A utilização de AI deveria ampliar as capacidades dos profissionais de engenharia, e não substituir os mecanismos necessários para garantir responsabilidade sobre as soluções produzidas.

A questão não é impedir a inovação, mas incorporá-la de maneira disciplinada.

Quarto erro: reter dados em excesso

Durante muitos anos, tornou-se comum ouvir que os dados representariam um dos ativos mais importantes das organizações.

Essa percepção ganhou ainda mais força conforme Data & Analytics e Artificial Intelligence ampliaram as possibilidades de gerar conhecimento a partir das informações disponíveis.

Entretanto, existe uma diferença fundamental entre reconhecer o valor dos dados e assumir que toda informação deveria ser armazenada indefinidamente.

O acúmulo indiscriminado pode aumentar custos, complexidade operacional, riscos de segurança e dificuldades relacionadas à privacidade e conformidade regulatória.

Em determinadas situações, manter informações por períodos prolongados constitui uma obrigação legal ou regulatória. Em outras, a retenção prolongada pode representar uma decisão desnecessária.

O ponto central está em compreender por que cada conjunto de dados precisa existir, como será utilizado e por quanto tempo deverá permanecer armazenado.

Data Retention começa no desenho da solução

Políticas de retenção não deveriam ser definidas apenas depois que a aplicação entra em produção.

O desenho precisa contemplar o ciclo de vida das informações desde a concepção, incluindo coleta, classificação, processamento, armazenamento, compartilhamento, arquivamento e descarte.

Isso exige compreender quais informações são necessárias para executar os processos empresariais e quais obrigações precisam ser atendidas.

No contexto brasileiro, a Lei Geral de Proteção de Dados estabelece princípios relevantes para o tratamento de dados pessoais, incluindo finalidade, adequação e necessidade.

Entretanto, seria incorreto interpretar minimização como autorização para eliminar informações que precisam ser preservadas por outras obrigações legais ou regulatórias.

Em instituições financeiras, por exemplo, existem exigências específicas relacionadas

à manutenção de registros, rastreabilidade, prevenção a fraudes e atendimento aos órgãos supervisores.

Uma política madura precisa harmonizar essas diferentes necessidades.

Os custos vão além do armazenamento

O crescimento dos dados pode afetar diferentes componentes da arquitetura.

Bancos de dados volumosos podem exigir estratégias específicas de particionamento, indexação, arquivamento e otimização de consultas.

Backups e mecanismos de recuperação tornam-se mais complexos quando os volumes aumentam continuamente sem uma política adequada de gerenciamento.

Ambientes analíticos também podem apresentar custos relevantes relacionados a processamento, replicação, transferência e utilização de serviços Cloud.

Existe ainda o impacto sobre segurança, pois informações armazenadas precisam ser protegidas, monitoradas e eventualmente recuperadas em situações de incidentes.

Quanto maior a quantidade de informações sensíveis mantidas sem necessidade, maior pode ser a exposição da organização em caso de comprometimento.

Uma estratégia adequada de Data Lifecycle Management pode produzir benefícios simultâneos de eficiência, desempenho, segurança e conformidade.

AI exige atenção adicional à governança dos dados

A evolução da Artificial Intelligence acrescenta outra dimensão ao problema.

Organizações passaram a utilizar informações corporativas em mecanismos de busca semântica, Retrieval-Augmented Generation, modelos preditivos e aplicações inteligentes.

Essas soluções frequentemente dependem de novas estruturas de armazenamento, indexação e recuperação.

Isso exige atenção não apenas aos dados originais, mas também às representações derivadas e às cópias criadas durante o processamento.

Uma informação removida de determinado sistema pode continuar presente em índices, caches, backups ou outros componentes, dependendo da arquitetura utilizada.

A governança precisa considerar essas particularidades para evitar inconsistências

entre políticas corporativas e implementação tecnológica.

O gerenciamento de dados não deveria ser percebido exclusivamente como responsabilidade das equipes de infraestrutura ou banco de dados. Ele constitui parte essencial da engenharia de soluções.

Quinto erro: terceirizar o trabalho errado

A terceirização continua sendo uma das principais alternativas utilizadas pelas organizações para ampliar capacidade, acessar competências especializadas, acelerar entregas e administrar custos.

Em muitos contextos, fornecedores externos desempenham um papel fundamental no desenvolvimento e na manutenção de soluções empresariais.

Entretanto, a decisão de terceirizar não deveria ser baseada exclusivamente na comparação entre o custo de profissionais internos e externos.

Existe uma dimensão estratégica relacionada à preservação de conhecimento, autonomia, capacidade de decisão e controle sobre componentes críticos do ambiente tecnológico.

O problema não está necessariamente em terceirizar determinada atividade, mas em transferir responsabilidades ou competências sem compreender adequadamente as consequências de longo prazo.

A diferença entre execução e ownership

Uma organização pode contratar fornecedores para desenvolver aplicações e continuar mantendo internamente o ownership das soluções.

Isso significa preservar responsabilidade sobre arquitetura, decisões técnicas relevantes, segurança, priorização, conhecimento funcional e evolução do produto.

Por outro lado, existem situações nas quais praticamente todas as informações importantes sobre determinada aplicação ficam concentradas no fornecedor.

A organização contratante passa a depender dele para estimar mudanças, analisar incidentes, compreender componentes e administrar evoluções.

Essa dependência pode se tornar especialmente problemática quando contratos precisam ser renegociados, fornecedores são substituídos ou novas prioridades exigem alterações significativas.

A transferência da execução não deveria significar automaticamente a transferência da

capacidade de compreender e administrar o próprio ambiente tecnológico.

Build, Buy e Partner

A decisão entre desenvolver internamente, adquirir uma solução de mercado ou estabelecer uma parceria precisa considerar diferentes dimensões.

O desenvolvimento próprio pode oferecer maior controle e capacidade de diferenciação, mas exige investimentos em profissionais, tecnologia, manutenção e evolução.

A aquisição de soluções prontas pode reduzir o esforço de desenvolvimento e acelerar a disponibilização de funcionalidades consolidadas, embora introduza dependências relacionadas ao fornecedor e ao modelo comercial.

Parcerias podem oferecer acesso a competências especializadas e acelerar transformações, mas exigem mecanismos adequados de governança e integração.

Nenhuma dessas alternativas é universalmente superior.

A escolha deveria considerar o grau de diferenciação competitiva proporcionado pela solução, sua criticidade, os recursos disponíveis e as capacidades que a organização pretende preservar.

Vendor Lock-in e continuidade tecnológica

Outro risco relevante está relacionado ao Vendor Lock-in.

Determinadas soluções apresentam custos iniciais competitivos e, posteriormente, criam dificuldades significativas de migração.

Essas dificuldades podem decorrer de formatos proprietários, integrações complexas, contratos restritivos ou ausência de alternativas equivalentes.

O problema não se limita às aplicações tradicionais. Plataformas Cloud, ferramentas de desenvolvimento, serviços de AI e soluções SaaS também podem introduzir dependências importantes.

Isso não significa que tecnologias proprietárias devam ser evitadas indiscriminadamente, pois muitas oferecem vantagens relevantes.

O importante é compreender os trade-offs e estabelecer decisões conscientes sobre o nível de dependência aceitável.

Contratos detalhados são importantes, mas não substituem governança.

É necessário acompanhar qualidade das entregas, aderência arquitetural, segurança, documentação e evolução das competências.

Repositórios, pipelines, infraestrutura como código e históricos de decisões precisam permanecer devidamente administrados conforme as responsabilidades acordadas.

Uma terceirização bem-sucedida deve produzir benefícios sem comprometer o controle estratégico sobre os ativos tecnológicos.

Sexto erro: negligenciar os testes

A realização de testes sempre foi uma das principais disciplinas da engenharia de software, mas continua enfrentando dificuldades quando projetos são pressionados por prazos e custos.

Em determinadas situações, testes passam a ser tratados como uma etapa que pode ser reduzida quando o cronograma apresenta desvios.

Essa decisão normalmente reflete uma compreensão limitada sobre o papel da qualidade.

Testar não deveria ser considerado apenas uma atividade destinada a identificar defeitos antes da entrada em produção.

A qualidade precisa ser construída progressivamente ao longo de todo o ciclo de desenvolvimento, desde requisitos e arquitetura até implementação, operação e evolução.

Quanto mais tarde determinados problemas são identificados, maior pode ser o esforço necessário para compreendê-los e corrigi-los, embora o custo efetivo dependa das características do sistema e do defeito.

De Quality Assurance para Quality Engineering

Existe uma distinção importante entre tratar qualidade como uma atividade de verificação e administrá-la como uma capacidade integrada ao desenvolvimento.

No modelo tradicional, equipes especializadas frequentemente recebiam aplicações já desenvolvidas e executavam testes para identificar problemas.

Embora essa abordagem continue necessária em determinados contextos, ela apresenta limitações quando a identificação de defeitos ocorre muito tarde.

Quality Engineering procura incorporar mecanismos de qualidade desde as primeiras decisões, envolvendo requisitos testáveis, arquitetura, práticas de desenvolvimento,

automação e monitoramento.

Isso não elimina a importância dos profissionais especializados.

Pelo contrário, amplia sua contribuição ao permitir que participem da definição de estratégias, critérios de aceitação e mecanismos de prevenção de defeitos.

Automação e diversidade dos testes

A automação representa uma das principais capacidades necessárias para sustentar ciclos frequentes de desenvolvimento e implantação.

Testes unitários ajudam a verificar componentes específicos. Testes de integração avaliam a interação entre módulos e serviços. Testes end-to-end permitem validar jornadas mais amplas.

Testes de contrato ajudam a identificar incompatibilidades entre interfaces, enquanto testes de desempenho, carga, estresse e resiliência contribuem para avaliar características não funcionais.

A combinação adequada desses mecanismos depende da arquitetura e da criticidade da solução.

Uma automação extensa, mas instável ou mal desenhada, pode prejudicar produtividade quando produz falhas intermitentes e exige manutenção excessiva.

O objetivo não deveria ser simplesmente maximizar a quantidade de testes, mas construir uma estratégia de qualidade proporcional aos riscos e às necessidades do produto.

Segurança também precisa ser testada

A evolução de DevSecOps ampliou a integração entre desenvolvimento e segurança.

Ferramentas de Static Application Security Testing, Dynamic Application Security Testing e Software Composition Analysis podem contribuir para identificar problemas em diferentes etapas do ciclo de vida.

Entretanto, ferramentas automatizadas não substituem avaliação arquitetural, Threat Modeling ou análises especializadas quando necessárias.

O NIST Secure Software Development Framework reforça justamente a importância de incorporar práticas de segurança aos diferentes modelos de desenvolvimento.

O OWASP Top 10 de 2025 também demonstra que vulnerabilidades podem surgir em múltiplas dimensões, incluindo controle de acesso, configuração, criptografia, desenho

das aplicações e cadeia de suprimentos.

A segurança precisa ser tratada como atributo estrutural da solução.

A utilização de Artificial Intelligence também introduziu possibilidades para geração de testes e revisão de código, mas existe o risco de código e testes serem produzidos a partir das mesmas premissas equivocadas.

Isso reforça a necessidade de preservar critérios de aceitação independentes, validações empresariais e mecanismos adequados de revisão humana.

Sétimo erro: subestimar o planejamento

O planejamento continua sendo uma das atividades mais importantes e, paradoxalmente, mais frequentemente mal compreendidas no desenvolvimento de software.

Em alguns ambientes, planejar é associado à produção de documentação extensa, cronogramas detalhados e processos formais que poderiam retardar o início das atividades.

Em outros, a ausência de planejamento é interpretada como demonstração de agilidade, como se começar rapidamente fosse necessariamente melhor do que compreender adequadamente o problema.

Nenhuma dessas interpretações representa uma abordagem suficientemente madura.

Planejamento não precisa significar rigidez, da mesma forma que adaptação não deveria significar improvisação.

O objetivo consiste em reduzir incertezas relevantes, estabelecer prioridades, compreender dependências e criar condições para que as equipes possam executar suas atividades com eficiência.

Planejar começa por compreender o problema

Uma das causas recorrentes de dificuldades está relacionada à distância entre aquilo que o usuário realmente precisa e aquilo que foi especificado para a equipe técnica.

Requisitos podem ser formulados como soluções antecipadas, sem que o problema empresarial tenha sido suficientemente compreendido.

Nesse cenário, a organização desenvolve funcionalidades que aparentemente atendem às solicitações recebidas, mas não necessariamente produzem os resultados esperados.

Uma abordagem mais adequada consiste em compreender objetivos, jornadas, necessidades, restrições e critérios de sucesso antes de definir detalhadamente a solução.

Isso não exige necessariamente longos períodos de análise.

Conversas estruturadas, protótipos, experimentos e validações rápidas podem produzir conhecimento suficiente para orientar decisões iniciais.

O importante é compreender que a qualidade da definição do problema influencia profundamente a qualidade da implementação.

Planejamento progressivo e adaptativo

Em iniciativas complexas, dificilmente será possível conhecer todos os requisitos e dependências desde o início.

Isso não significa que o planejamento seja inútil. Significa que ele precisa acontecer em diferentes níveis de detalhamento.

A visão estratégica estabelece objetivos, benefícios esperados e restrições.

O planejamento tático organiza releases, dependências, recursos e prioridades.

O planejamento operacional detalha atividades e critérios de aceitação em horizontes menores.

Essa abordagem permite combinar direção com flexibilidade.

À medida que novas informações são produzidas, requisitos podem ser refinados, riscos reavaliados e alternativas tecnológicas reconsideradas.

O planejamento deveria funcionar como mecanismo contínuo de coordenação e aprendizado, e não como um documento estático destinado exclusivamente a justificar o início de um projeto.

O planejamento financeiro também importa

Projetos de desenvolvimento frequentemente são avaliados a partir dos custos de implementação, mas as soluções continuarão exigindo investimentos durante sua vida útil.

Infraestrutura, licenciamento, sustentação, Cybersecurity, observability, atualização tecnológica e atendimento a incidentes representam componentes importantes do Total Cost of Ownership.

Uma solução aparentemente barata para desenvolver pode se tornar extremamente onerosa para manter.

Da mesma forma, decisões destinadas a reduzir investimentos iniciais podem introduzir Technical Debt que comprometerá a capacidade futura de evolução.

O planejamento precisa considerar essas consequências e estabelecer uma relação clara entre investimentos, riscos e resultados empresariais.

Também precisa reconhecer que determinadas decisões serão necessariamente tomadas com informações incompletas. A maturidade não consiste em eliminar toda incerteza, mas em identificar quais incertezas precisam ser resolvidas antes de comprometer recursos significativos.

Artificial Intelligence e os novos riscos da engenharia

A evolução da Artificial Intelligence representa provavelmente a transformação mais relevante ocorrida no desenvolvimento de software desde a publicação original da matéria da CIO Online.

Ferramentas de AI-Assisted Development passaram a apoiar atividades como geração de código, análise de repositórios, criação de documentação, elaboração de testes e identificação de problemas.

Mais recentemente, agentes capazes de executar sequências de atividades ampliaram essa transformação, permitindo automatizar tarefas que envolvem análise de arquivos, execução de comandos, integração com ferramentas e validação de resultados.

Essas possibilidades podem modificar significativamente a produtividade das equipes.

Entretanto, considero importante evitar uma interpretação excessivamente simplificada segundo a qual produzir mais código significa necessariamente gerar mais valor.

AI pode acelerar problemas existentes

Imagine uma equipe que recebe requisitos incompletos, trabalha com arquitetura inadequada e enfrenta dificuldades para administrar dependências.

A introdução de AI pode aumentar significativamente sua capacidade de produzir implementações.

Mas, se os problemas originais permanecerem, a organização poderá simplesmente

gerar mais código que precisará ser revisado, corrigido ou descartado.

Essa é uma das conclusões fundamentais do relatório DORA de 2025, que apresentou a AI como amplificadora das características existentes no sistema organizacional.

Os benefícios não dependem exclusivamente da ferramenta, mas também da qualidade dos processos, plataformas, arquitetura e mecanismos de colaboração.

Portanto, a adoção de AI precisa ser acompanhada de investimentos em engenharia, governança e qualidade.

A tecnologia pode ampliar produtividade individual, mas os resultados empresariais dependem de como essa produtividade se transforma em soluções efetivamente utilizadas e sustentáveis.

O papel do desenvolvedor está mudando

A evolução das ferramentas também pode modificar a distribuição das atividades realizadas pelos profissionais.

Parte do esforço anteriormente dedicado à produção manual de código poderá ser direcionada para análise de requisitos, arquitetura, revisão, integração e avaliação dos resultados produzidos pelas ferramentas.

Isso não significa que conhecimento técnico aprofundado deixará de ser necessário.

Pelo contrário, quanto maior a capacidade das ferramentas de produzir implementações complexas, maior pode se tornar a importância de profissionais capazes de avaliar sua adequação.

Será necessário compreender estruturas de dados, algoritmos, integração, desempenho, segurança e comportamento operacional para identificar problemas que não são evidentes na aparência inicial do código.

Uma implementação produzida rapidamente pode apresentar excelente organização sintática e ainda assim incorporar decisões arquiteturais inadequadas.

A evolução profissional provavelmente exigirá uma combinação cada vez maior entre domínio técnico, compreensão de arquitetura, capacidade analítica e habilidade para avaliar resultados produzidos por sistemas automatizados.

Agentic Software Development exige nova

governança

A evolução de agentes de AI amplia a discussão porque determinados sistemas passaram a executar atividades de maneira mais autônoma.

Em vez de apenas sugerir trechos de código, um agente pode analisar uma solicitação, identificar componentes, modificar arquivos, executar testes e produzir uma proposta de implementação.

Esse modelo oferece oportunidades relevantes, mas também introduz riscos relacionados a permissões, execução de comandos, exposição de informações e alterações não autorizadas.

O OWASP Top 10 for Agentic Applications de 2026, publicado em dezembro de 2025, reforçou a necessidade de tratar riscos específicos associados a agentes capazes de utilizar ferramentas e executar ações.

A utilização dessas capacidades em ambientes empresariais precisa contemplar limites de autonomia, controles de acesso, revisão de alterações e mecanismos de rastreabilidade.

A questão central deixa de ser apenas aquilo que a ferramenta consegue produzir e passa a incluir aquilo que ela está autorizada a executar.

Human-in-the-Loop continua relevante, especialmente em decisões de maior criticidade, mas os controles devem ser proporcionais aos riscos para evitar que a automação seja inviabilizada por burocracia desnecessária.

Platform Engineering e Software Supply Chain Security

Outra evolução importante está relacionada à consolidação de Platform Engineering como disciplina destinada a melhorar a experiência das equipes de desenvolvimento e reduzir a complexidade associada à utilização de ferramentas e serviços tecnológicos.

Em grandes organizações, desenvolvedores frequentemente precisam interagir com múltiplas plataformas para provisionar ambientes, acessar repositórios, executar testes, configurar pipelines e disponibilizar aplicações.

Quando essas atividades exigem processos manuais ou conhecimento especializado sobre inúmeras ferramentas, uma parcela significativa do esforço acaba sendo direcionada para tarefas que não representam diretamente a criação de funcionalidades empresariais.

Platform Engineering procura reduzir essa complexidade através de plataformas internas que disponibilizam capacidades padronizadas e reutilizáveis.

Uma Internal Developer Platform pode oferecer mecanismos para provisionamento de ambientes, administração de configurações, execução de pipelines e implementação de controles de segurança.

Essas capacidades podem ser disponibilizadas através de interfaces self-service, reduzindo dependências operacionais e permitindo maior autonomia.

Golden Paths representam caminhos recomendados para execução de atividades recorrentes, combinando ferramentas, padrões arquiteturais e configurações previamente estabelecidas.

Entretanto, essas plataformas também precisam ser administradas como produtos, considerando experiência dos desenvolvedores, adoção, custos e evolução contínua.

Padronização precisa facilitar o trabalho, não impedir alternativas justificadas.

Paralelamente, a segurança da cadeia de suprimentos de software ganhou importância conforme organizações passaram a depender intensamente de componentes desenvolvidos por terceiros.

Uma aplicação pode possuir código próprio cuidadosamente revisado e, ainda assim, incorporar vulnerabilidades presentes em bibliotecas ou dependências externas.

Também existem riscos relacionados ao comprometimento de repositórios, ferramentas de desenvolvimento, pipelines e mecanismos de distribuição.

O OWASP Top 10 de 2025 reforçou explicitamente essa dimensão ao destacar Software Supply Chain Failures como uma categoria específica.

Software Bill of Materials, gerenciamento de dependências, verificação de proveniência, assinatura de artefatos e controles de acesso aos pipelines representam mecanismos que podem contribuir para aumentar rastreabilidade e reduzir riscos.

Essas práticas demonstram que a engenharia de software moderna precisa administrar não apenas aquilo que desenvolve diretamente, mas também todos os componentes e processos utilizados para produzir e disponibilizar suas soluções.

Medindo a maturidade da engenharia de software

A capacidade de medir desempenho representa outra dimensão importante da evolução das práticas de engenharia.

Durante muito tempo, organizações utilizaram indicadores como quantidade de linhas de código, esforço consumido, funcionalidades entregues e cumprimento de cronogramas.

Embora determinadas métricas possam possuir utilidade em contextos específicos, elas não representam necessariamente a capacidade de produzir valor.

Uma equipe pode entregar inúmeras funcionalidades pouco utilizadas, enquanto outra realiza uma quantidade menor de mudanças que produzem impactos empresariais significativos.

Da mesma forma, aumentar o volume de código pode representar tanto evolução funcional quanto crescimento desnecessário da complexidade.

DORA e a capacidade de entrega

As pesquisas DORA contribuíram para popularizar indicadores relacionados à capacidade de entregar software com velocidade e estabilidade.

Lead time de mudanças, frequência de deployment, falhas decorrentes de alterações e mecanismos de recuperação representam dimensões importantes para compreender o desempenho da entrega.

Entretanto, esses indicadores precisam ser interpretados dentro do contexto de cada organização.

Comparar indiscriminadamente equipes que desenvolvem produtos com características, riscos e restrições diferentes pode produzir conclusões equivocadas.

Também existe o risco de transformar métricas em objetivos isolados, incentivando comportamentos destinados a melhorar indicadores sem necessariamente melhorar resultados.

Da produtividade técnica ao valor empresarial

Uma abordagem mais completa deveria combinar diferentes perspectivas:

- **Eficiência:** tempo necessário para transformar uma necessidade em funcionalidade disponível.
- **Qualidade:** defeitos, recorrência de problemas e comportamento das soluções em produção.
- **Confiabilidade:** disponibilidade, recuperação de incidentes e cumprimento dos objetivos de serviço.

- **Segurança:** vulnerabilidades, tempo de correção e aderência aos controles necessários.
- **Sustentabilidade:** Technical Debt, obsolescência e capacidade de manutenção.
- **Developer Experience:** facilidade de utilização das ferramentas, autonomia e eficiência dos processos.
- **Business Outcomes:** contribuição das soluções para eficiência, receita, experiência dos clientes e redução de riscos.

Nenhuma dessas perspectivas deveria ser analisada isoladamente.

A maturidade da engenharia precisa ser avaliada pela capacidade de equilibrar velocidade, qualidade, segurança, sustentabilidade e geração de valor.

O papel da liderança na qualidade

Existe uma tendência de associar problemas de qualidade exclusivamente às equipes que efetivamente desenvolvem aplicações.

Entretanto, muitas dificuldades começam antes da primeira linha de código.

Prioridades conflitantes, prazos artificialmente reduzidos, requisitos incompletos, investimentos insuficientes em arquitetura e decisões inadequadas de sourcing frequentemente são determinados em níveis gerenciais ou executivos.

Isso significa que a qualidade da engenharia não depende exclusivamente da disciplina técnica dos desenvolvedores.

Ela também reflete decisões sobre orçamento, organização, governança, cultura e modelo operacional.

Uma liderança que exige velocidade máxima enquanto reduz continuamente investimentos em testes, modernização e segurança pode criar condições incompatíveis com os resultados esperados.

Da mesma forma, uma organização que considera todas as demandas igualmente prioritárias provavelmente enfrentará dificuldades para administrar capacidade e estabelecer foco.

A excelência não está em maximizar isoladamente velocidade ou controle. Está em construir uma capacidade sustentável de entregar soluções de qualidade, mantendo flexibilidade suficiente para responder às transformações do negócio.

CIO Codex Framework – Solution Engineering

Dentro do CIO Codex Framework, o tópico que apresenta maior convergência com os desafios discutidos neste artigo é **Solution Engineering**, uma das sete grandes camadas de IT Capabilities e também um domínio estruturante do IT Reference Model.

Sua relevância está relacionada à capacidade de organizar as competências necessárias para transformar demandas, requisitos e necessidades empresariais em soluções tecnológicas efetivamente implementadas, mantidas e evoluídas ao longo do tempo.

Essa perspectiva é particularmente importante porque engenharia de soluções não deveria ser compreendida exclusivamente como desenvolvimento de aplicações.

Trata-se de uma capacidade organizacional mais ampla, responsável por coordenar diferentes disciplinas necessárias para que soluções sejam concebidas, construídas, integradas, disponibilizadas e sustentadas com qualidade.

No CIO Codex, Solution Engineering ocupa uma posição essencial na conexão entre necessidades empresariais e materialização das soluções que serão utilizadas pela organização.

Essa conexão envolve relacionamentos com Business Partnership, Technology Visioning, New Technology Exploration, Service Excellence e Cybersecurity, demonstrando que a entrega de software não acontece isoladamente das demais capacidades de TI.

Solution Engineering como capacidade organizacional

Uma das premissas centrais do CIO Codex Framework está relacionada à distinção entre atividades individuais e capacidades organizacionais.

Uma empresa pode possuir excelentes desenvolvedores e ainda assim apresentar baixa maturidade em engenharia de soluções.

Isso acontece quando competências individuais não são acompanhadas de processos, ferramentas, modelos de responsabilidade, mecanismos de gestão e conhecimento suficientemente estruturados.

A capacidade organizacional precisa continuar funcionando mesmo quando profissionais mudam de equipe, fornecedores são substituídos ou novas tecnologias

passam a fazer parte do ambiente.

Solution Engineering deve ser compreendida como uma combinação de pessoas, competências, práticas, ferramentas, processos e mecanismos de governança que permite entregar soluções de maneira consistente.

Essa distinção ajuda a explicar por que a simples contratação de profissionais tecnicamente qualificados nem sempre resolve problemas persistentes de desenvolvimento.

A maturidade depende da maneira como essas competências são organizadas, integradas e direcionadas para resultados.

O CIO Codex Reference Model organiza esse domínio através de macro capabilities que abrangem gestão de projetos, Application Ownership e Solution Development.

Essa organização permite visualizar a engenharia como um conjunto integrado de responsabilidades, e não apenas como uma atividade de programação.

Project Office e gestão da execução

A macro capability de Project Office contempla competências necessárias para planejar, coordenar e acompanhar iniciativas de tecnologia.

Isso inclui definição de escopo, objetivos, recursos, cronogramas, dependências e critérios de sucesso.

Também contempla mecanismos de coordenação em ambientes ágeis, incluindo práticas relacionadas a Program Increment Planning e Release Train Management.

Esses mecanismos possuem relevância especial em organizações nas quais múltiplas equipes precisam colaborar para entregar soluções complexas.

O objetivo não deveria ser impor burocracia, mas garantir que iniciativas possuam condições adequadas para execução.

Uma capacidade madura de Project Office precisa contribuir para identificar riscos, administrar conflitos de prioridade e estabelecer visibilidade sobre o andamento das entregas.

Também deve permitir que diferentes metodologias coexistam quando necessário, preservando mecanismos comuns de acompanhamento e governança.

Essa abordagem conecta diretamente o domínio de Solution Engineering aos desafios relacionados à escolha de metodologias e à qualidade do planejamento.

Application Ownership e gestão do ciclo de vida

Outra macro capability fundamental é Application Ownership.

Seu propósito envolve estabelecer responsabilidade clara sobre aplicações e garantir que o conhecimento necessário para sua manutenção e evolução permaneça disponível ao longo do tempo.

O CIO Codex Reference Model contempla competências relacionadas a Application Support Management, Application Sustain Management, Application Evolution Management e Application Lifecycle Management.

Essas competências cobrem diferentes necessidades.

O suporte contribui para resolver problemas e atender usuários. A sustentação envolve correções, atualizações e manutenção. A evolução permite incorporar novas funcionalidades e adaptar aplicações às mudanças empresariais.

Application Lifecycle Management contempla a gestão integral do ciclo de vida, incluindo decisões sobre modernização, obsolescência, substituição e desativação.

Essa perspectiva é especialmente importante porque aplicações normalmente permanecem em operação durante períodos muito superiores à duração dos projetos que deram origem a elas.

Uma organização que administra apenas projetos e não acompanha adequadamente seus ativos de aplicações corre o risco de acumular soluções sem ownership claro ou estratégias consistentes de evolução.

Solution Development e engenharia integrada

A macro capability de Solution Development reúne competências diretamente relacionadas à concepção e construção das soluções.

Entre elas estão UX Design, Solution Analyze, Solution Design, Composability Design, Test Design, Deployment & Release Design, Coding, Test Execution & Automation e Developer Autonomy & DevSecOps.

Essa estrutura demonstra que desenvolvimento de software precisa ser compreendido como uma atividade muito mais abrangente do que escrever código.

UX Design contribui para compreender necessidades e experiências dos usuários.

Solution Analyze aprofunda requisitos técnicos e empresariais, enquanto Solution Design estabelece os componentes arquiteturais necessários.

Composability Design favorece modularidade e reutilização, reduzindo a necessidade de reconstruir capacidades semelhantes em diferentes soluções.

Test Design e Test Execution & Automation estabelecem mecanismos de validação e qualidade.

Deployment & Release Design contribui para planejar a disponibilização das soluções de maneira controlada.

Coding permanece como uma competência fundamental, mas passa a ser compreendida como parte de um processo mais amplo.

Developer Autonomy & DevSecOps, por sua vez, reforça a importância de disponibilizar ferramentas, práticas e mecanismos que permitam maior autonomia sem comprometer segurança e governança.

A integração com Business Partnership e Technology Visioning

Uma engenharia de soluções bem-sucedida precisa começar pela compreensão das necessidades empresariais.

No CIO Codex Framework, Business Partnership representa a camada responsável por fortalecer a conexão entre Tecnologia da Informação e negócio, incluindo gestão de demandas, relacionamento com stakeholders e alinhamento estratégico.

Essa interface é fundamental porque as equipes de engenharia precisam compreender o propósito das soluções desenvolvidas.

Uma solicitação empresarial pode chegar como uma demanda funcional, mas sua implementação exige decisões que considerem objetivos, restrições, prioridades e benefícios esperados.

Quando essa conexão é insuficiente, existe o risco de desenvolver tecnicamente aquilo que foi solicitado sem compreender adequadamente o problema que deveria ser resolvido.

Ao mesmo tempo, Solution Engineering precisa operar de maneira integrada a Technology Visioning.

Decisões sobre linguagens, frameworks, plataformas, integração, bancos de dados e arquiteturas precisam considerar não apenas requisitos imediatos, mas também sua aderência à visão tecnológica corporativa.

Enterprise Architecture contribui para estabelecer princípios e direcionadores que

reduzem fragmentação tecnológica e favorecem interoperabilidade.

Essa relação precisa ser bidirecional. A arquitetura orienta o desenvolvimento, enquanto a experiência prática das equipes contribui para aperfeiçoar a própria visão tecnológica.

Quality Engineering, DevSecOps e segurança

A qualidade representa outra dimensão fundamental de Solution Engineering.

O CIO Codex Reference Model contempla explicitamente testes, incluindo automação e User Acceptance Testing.

Essa abordagem reforça a necessidade de integrar diferentes mecanismos de validação ao ciclo de desenvolvimento.

A qualidade precisa começar pela compreensão dos requisitos e continuar através de arquitetura, implementação, testes, implantação e operação.

Testes automatizados podem reduzir o esforço necessário para validar alterações frequentes e identificar regressões.

Testes de aceitação ajudam a confirmar que a solução atende às necessidades empresariais.

Testes não funcionais permitem avaliar desempenho, segurança, resiliência e outros atributos relevantes.

DevSecOps contribui para integrar desenvolvimento, segurança e operações através de práticas que promovem entregas mais consistentes.

Continuous Integration permite incorporar e verificar alterações frequentemente, enquanto Continuous Delivery estabelece condições para disponibilização controlada.

Entretanto, DevSecOps não deveria ser interpretado exclusivamente como implantação de ferramentas.

Sua efetividade depende da maneira como responsabilidades são distribuídas e como diferentes equipes colaboram ao longo do ciclo de vida.

Dentro do CIO Codex Framework, Cybersecurity também possui natureza transversal, reforçando que proteção precisa estar presente em todas as etapas da engenharia.

A conexão com Service Excellence e operação

Uma das conexões mais importantes de Solution Engineering envolve Service

Excellence.

A implantação de uma aplicação representa apenas uma etapa de seu ciclo de vida.

Após a entrada em produção, a solução passa a depender de mecanismos de monitoramento, suporte, gestão de incidentes, capacidade, disponibilidade e continuidade.

Essas competências pertencem ao universo mais amplo de excelência na prestação de serviços de TI.

A integração entre engenharia e operação precisa começar antes da implantação.

Requisitos de observability, documentação operacional, procedimentos de recuperação e mecanismos de suporte precisam ser considerados durante o desenvolvimento.

Quando essas preocupações são deixadas para o final, a transição pode produzir dificuldades que comprometem estabilidade e continuidade.

Uma engenharia madura precisa garantir que aquilo que foi desenvolvido também possa ser operado adequadamente.

Essa perspectiva ajuda a eliminar a separação artificial entre sucesso do projeto e sucesso do serviço.

New Technology Exploration e evolução contínua

A relação entre Solution Engineering e New Technology Exploration também merece destaque.

Novas tecnologias precisam ser pesquisadas, experimentadas e avaliadas antes de serem incorporadas de maneira mais ampla ao ambiente empresarial.

A exploração tecnológica pode identificar oportunidades relacionadas a Artificial Intelligence, automação, Data & Analytics, Cloud e outras capacidades emergentes.

Entretanto, o amadurecimento dessas tecnologias exige integração com as práticas convencionais de engenharia.

Uma solução experimental pode funcionar adequadamente em laboratório e enfrentar dificuldades significativas quando submetida aos requisitos de segurança, escalabilidade, disponibilidade e governança de uma grande organização.

Por isso, existe uma transição importante entre experimentação e industrialização.

Solution Engineering contribui para transformar tecnologias emergentes em soluções sustentáveis, incorporando padrões, mecanismos de teste, integração e gestão do ciclo

de vida.

Essa conexão permite que a inovação aconteça sem criar um ambiente paralelo desconectado da arquitetura e da operação corporativa.

Solution Engineering como capacidade estratégica

Quando observamos o conjunto dessas dimensões, fica evidente que Solution Engineering representa muito mais do que uma estrutura destinada a administrar desenvolvedores.

Trata-se da capacidade de transformar necessidades empresariais em soluções tecnológicas que possam ser implementadas, operadas, protegidas e evoluídas ao longo do tempo.

Essa capacidade depende de integração com estratégia, arquitetura, inovação, segurança, operação e governança.

Também exige mecanismos adequados de gestão de pessoas, conhecimento, fornecedores e recursos financeiros.

Os sete erros apresentados pela CIO Online podem ser interpretados, nesse contexto, como manifestações de lacunas em diferentes componentes dessa capacidade organizacional.

A escolha inadequada de metodologia demonstra dificuldades de organização do trabalho. Problemas de escalabilidade refletem limitações arquiteturais. A adoção indiscriminada de novidades aponta para ausência de critérios de avaliação.

Retenção excessiva de dados evidencia lacunas de governança. Terceirização inadequada pode demonstrar problemas de ownership. Ausência de testes revela fragilidades de qualidade. Planejamento insuficiente compromete a conexão entre necessidades e execução.

A contribuição do CIO Codex Framework está justamente em permitir que esses problemas sejam observados de maneira integrada, evitando que cada dificuldade seja tratada como incidente isolado.

O objetivo final não deveria ser simplesmente eliminar erros específicos, mas desenvolver uma capacidade organizacional capaz de prevenir, identificar e corrigir sistematicamente os fatores que comprometem a qualidade das soluções.

Essa é uma das diferenças fundamentais entre possuir profissionais tecnicamente competentes e construir uma organização de engenharia verdadeiramente madura.

Concluindo

Os sete erros apresentados pela CIO Online continuam extremamente relevantes, mesmo depois das transformações ocorridas na engenharia de software desde a publicação original da matéria.

Metodologias inadequadas, limitações de escalabilidade, adoção precipitada de tecnologias, retenção excessiva de dados, decisões equivocadas de terceirização, insuficiência de testes e falhas de planejamento permanecem entre os fatores capazes de comprometer projetos e produtos tecnológicos.

Entretanto, a evolução recente da indústria reforça a necessidade de observar esses problemas dentro de um contexto mais amplo.

Cloud Native, Platform Engineering, DevSecOps, Software Supply Chain Security e Artificial Intelligence ampliaram significativamente as possibilidades de desenvolvimento, mas também introduziram novas dependências, riscos e responsabilidades.

A própria AI representa um excelente exemplo dessa dualidade.

Ferramentas capazes de gerar código e automatizar atividades podem aumentar produtividade, acelerar experimentações e reduzir determinados esforços de implementação.

Mas nenhuma dessas vantagens elimina a necessidade de compreender requisitos, estabelecer arquitetura adequada, garantir segurança e administrar qualidade.

Uma organização que utiliza AI para desenvolver software mais rapidamente, mas não possui práticas maduras de engenharia, pode acabar aumentando a velocidade com que acumula Technical Debt ou introduz problemas em seu ambiente.

Por outro lado, organizações que já possuem boas capacidades de desenvolvimento podem utilizar essas ferramentas para ampliar ainda mais sua eficiência.

Isso reforça uma conclusão que considero particularmente importante: a verdadeira vantagem competitiva não está simplesmente em possuir acesso às tecnologias mais recentes, mas em construir capacidades organizacionais capazes de utilizá-las adequadamente.

É justamente por isso que considero relevante conectar essa discussão ao CIO Codex Framework.

Solution Engineering demonstra que o desenvolvimento de software precisa ser compreendido como uma capacidade integrada, envolvendo gestão de demandas,

arquitetura, desenvolvimento, qualidade, segurança, operação, fornecedores e evolução contínua.

Essa perspectiva também ajuda a superar uma visão excessivamente centrada em projetos.

O sucesso não deveria ser definido exclusivamente pelo momento em que uma aplicação entra em produção, mas pela sua capacidade de continuar atendendo às necessidades empresariais ao longo do tempo.

Uma solução de qualidade precisa ser funcional, segura, escalável, confiável, sustentável e economicamente adequada.

Também precisa possuir condições para evoluir conforme requisitos, tecnologias, regulações e expectativas dos usuários se transformam.

Isso exige uma abordagem que combine disciplina técnica com visão estratégica.

As equipes precisam compreender não apenas o código que estão produzindo, mas também o problema que estão resolvendo, os riscos que estão assumindo e as consequências de suas decisões.

A liderança, por sua vez, precisa garantir que prioridades, investimentos, estrutura organizacional e mecanismos de governança estejam coerentes com os resultados esperados.

Talvez a principal lição dos sete erros esteja justamente no reconhecimento de que eles raramente acontecem de maneira completamente isolada.

Uma metodologia inadequada pode prejudicar planejamento e qualidade. Uma decisão arquitetural precipitada pode comprometer escalabilidade. Uma terceirização mal estruturada pode ampliar Technical Debt e dificultar manutenção. A ausência de testes pode transformar problemas relativamente simples em incidentes operacionais relevantes.

O resultado é que dificuldades aparentemente independentes acabam se reforçando mutuamente.

Por isso, a excelência em engenharia de software não deveria ser perseguida através de soluções pontuais ou da adoção sucessiva de novas ferramentas.

Ela precisa ser construída como uma capacidade permanente da organização, sustentada por competências, processos, tecnologias, cultura e mecanismos de melhoria contínua.

A tecnologia continuará evoluindo, novas ferramentas surgirão e a Artificial Intelligence provavelmente modificará ainda mais profundamente a maneira como

aplicações são concebidas e desenvolvidas.

Mas a necessidade de compreender problemas, tomar decisões adequadas, administrar riscos e produzir soluções de qualidade continuará existindo.

No fim, talvez a pergunta mais relevante não seja como desenvolver software cada vez mais rapidamente, mas como desenvolver software que realmente mereça ser colocado em produção, utilizado pelos clientes e mantido durante os próximos anos.

Porque velocidade sem qualidade apenas antecipa problemas, enquanto qualidade sem capacidade de evolução pode comprometer a competitividade.

A verdadeira excelência está em construir soluções que consigam combinar ambas as dimensões, transformando capacidade tecnológica em resultados empresariais consistentes e sustentáveis.