DO MONÓLITO AOS MICROSERVIÇOS: OS RISCOS INVISÍVEIS QUE TRAVAM A TRANSFORMAÇÃO

COMO A **FALTA DE VISIBILIDADE** SOBRE DEPENDÊNCIAS — ESPECIALMENTE NO BANCO DE DADOS — **ELEVA O RISCO**, RETARDA DECISÕES E COMPROMETE A MODERNIZAÇÃO

FAUSTO BACCHI NETO

ABRIL-2026 | REV 2

EXECUTIVE BRIEF

Muitas empresas acreditam que estão modernizando seus sistemas — quando, na prática, não conseguem enxergar o que estão tentando transformar.

A promessa da modernização

Por muitos anos, **arquiteturas monolíticas sustentaram com eficiência os sistemas corporativos**. Em estágios iniciais, sua simplicidade favorece o desenvolvimento, a implantação e a operação. No entanto, à medida que esses sistemas crescem, **tornam-se cada vez mais complexos, rígidos e difíceis de evoluir**.

Com o aumento das demandas por inovação, escalabilidade e velocidade, o modelo monolítico começa a apresentar sinais claros de limitação. Mudanças passam a exigir maior coordenação entre equipes, ciclos de desenvolvimento se tornam mais longos e **o risco associado a cada alteração aumenta significativamente**. O que antes era simples e funcional passa a ser um fator de atrito.

É nesse contexto que **arquiteturas baseadas em microserviços ganham espaço**. Segundo O'Reilly Media, sua adoção cresceu de forma consistente nos últimos anos, impulsionada pela necessidade de reduzir acoplamento, aumentar autonomia dos times e acelerar a entrega de valor. Paralelamente, análises da Gartner reforçam que arquiteturas cloud-native e microservices já são consideradas pilares da transformação digital nas organizações.

A modernização, portanto, deixa de ser uma escolha puramente técnica e passa a refletir uma necessidade estratégica. Em um cenário onde velocidade e capacidade de adaptação definem competitividade, evoluir a arquitetura torna-se inevitável.

O desafio é que, na maioria dos casos, **as empresas não estão migrando sistemas — estão tentando transformar algo que já não compreendem completamente**.

QUOTES INTERESSANTES:

> *Most organizations fail to realize that the biggest barrier to digital transformation is not technology — it is complexity*
>
> **McKinsey & Company**

> *70% of digital transformation initiatives fail to reach their goals*
>
> **McKinsey & Company**

A realidade: por que a migração trava

Apesar da clareza estratégica e dos benefícios amplamente reconhecidos, a migração de arquiteturas monolíticas para microserviços raramente ocorre de forma simples. **Na prática, o que muitas organizações encontram não é aceleração — é fricção**. Projetos se estendem por anos, custos ultrapassam previsões iniciais e a complexidade cresce em um ritmo difícil de controlar.

Esse cenário não é exceção. Estudos consolidados na literatura de engenharia de software apontam que **a migração para microserviços é, por natureza, altamente complexa e pouco padronizada**. Cada sistema carrega sua própria história, suas decisões acumuladas e um nível único de acoplamento, o que torna inviável aplicar abordagens genéricas com previsibilidade. **O que parecia um problema de arquitetura rapidamente se revela um problema de entendimento**.

Na prática, isso já foi observado em empresas que tornaram pública sua jornada de modernização. A SoundCloud relatou dificuldades em evoluir seu monólito e enfrentou anos de complexidade durante a transição. A Zalando destacou os desafios na definição de limites e separação de dependências ao migrar para microserviços, enquanto a Shopify evidenciou o impacto do acoplamento interno na evolução de seu sistema. A Amazon, por sua vez, precisou promover uma transformação estrutural profunda — técnica e organizacional — para viabilizar a decomposição de seu monólito. Em todos os casos, um padrão se repete: **a dificuldade não está apenas em mudar a arquitetura, mas em compreender, com precisão, como o sistema realmente funciona.**

MODERNIZAR SEM ENTENDER

"Sem compreender como o sistema realmente funciona, qualquer tentativa de modernização deixa de ser estratégia — passa a ser risco."

O ponto central é que sistemas monolíticos não são apenas grandes blocos de código — são estruturas vivas, moldadas por anos de decisões, integrações e adaptações. Ao longo do tempo, acumulam dependências que raramente estão visíveis ou documentadas de forma confiável. Quando essas relações não são compreendidas com precisão, decisões passam a ser tomadas com base em aproximações.

E é nesse momento que **a transformação deixa o campo da estratégia e entra no campo do risco**. Porque, quando uma organização não compreende completamente o próprio sistema, ela não está conduzindo uma modernização — está tentando compreender, em tempo real, um sistema que já deveria conhecer.

QUOTES INTERESSANTES:

> *Lack of visibility into application dependencies is a primary cause of system failures in modern architectures*
>
> **IDC**

> *As systems evolve, complexity grows faster than the ability to manage it*
>
> **Gartner**

O ponto cego: dependências invisíveis

Se a migração para microserviços é complexa por natureza, existe um fator que amplifica essa dificuldade de forma silenciosa — e, muitas vezes, ignorada: **a falta de visibilidade sobre as dependências reais do sistema.** Não aquelas que aparecem nos diagramas ou nos slides — mas aquelas que efetivamente governam o comportamento do código em produção.

Essas dependências não estão restritas ao banco de dados. Elas se manifestam em toda a arquitetura — entre serviços, dentro do próprio código, nas integrações e nas regras de negócio distribuídas ao longo do istema. Chamadas indiretas, contratos implícitos e fluxos não documentados criam um emaranhado que, embora funcional, raramente é compreendido em sua totalidade.

Ao longo do tempo, sistemas monolíticos deixam de ser estruturas organizadas e passam a se comportar como organismos vivos. Regras de negócio se espalham, integrações são adaptadas, exceções viram padrão. **O resultado não é apenas complexidade — é opacidade**. A arquitetura documentada passa a ser, no máximo, uma aproximação confortável da realidade.

VOCÊ NÃO ESTÁ VENDO TUDO

"Sistemas operam por dependências — muitas delas invisíveis, não documentadas e decisivas para o comportamento em produção."

O problema é que essa diferença entre o que se acredita e o que de fato existe raramente é percebida com clareza. **Equipes tomam decisões baseadas em uma visão parcial do sistema**, assumindo limites que não estão realmente definidos e ignorando dependências que nunca foram explicitadas. E, **enquanto tudo permanece estável, essa ilusão de controle se sustenta. Ela só se rompe quando algo muda.**

É nesse momento que surgem efeitos inesperados, impactos não previstos e comportamentos difíceis de explicar. Não porque o sistema seja imprevisível — mas porque ele nunca foi completamente compreendido. Dependências indiretas, acessos compartilhados e relações implícitas deixam de ser invisíveis apenas quando passam a causar problemas.

Sem visibilidade real, decompor um sistema deixa de ser um exercício de engenharia e passa a ser um exercício de tentativa e erro. E, em ambientes críticos, tentativa e erro não é estratégia — é exposição. No fim, o maior desafio não está em desenhar uma nova arquitetura. Está em encarar um fato desconfortável: a maior parte das organizações não conhece, com precisão, o sistema que já opera.

QUOTES INTERESSANTES:

> *Data gravity is one of the most underestimated challenges in distributed architectures*
>
> **AWS Cloud Architecture papers**

> *Tightly coupled systems are the biggest barrier to scaling software delivery*
>
> **ThoughtWorks Technology Radar**

O verdadeiro gargalo: o banco de dados

Se as dependências invisíveis explicam por que a migração trava, **é fato que o problema não está restrito a um único ponto da arquitetura**. Ele se manifesta em todo o sistema — **nas interações entre serviços, no próprio código e nas integrações** que sustentam seu funcionamento. Mas é normalmente no banco de dados que esse emaranhado se torna ainda **mais denso, mais crítico e, muitas vezes, mais negligenciado.**

Em arquiteturas monolíticas, o banco de dados raramente é apenas um repositório de informações. **Ele se torna um ponto de convergência** onde diferentes partes do sistema compartilham estruturas, acessam as mesmas tabelas e, muitas vezes, **dependem das mesmas regras implícitas**. Ao longo do tempo, o que deveria ser um componente de suporte passa a operar como um elo estrutural entre múltiplos domínios.

A RAIZ DO PROBLEMA

"Enquanto serviços podem ser reorganizados, o banco de dados concentra dependências invisíveis que não se decompõem facilmente."

O problema é que essas relações não são visíveis de forma clara. Queries espalhadas no código, acessos indiretos, dependências cruzadas entre módulos — **tudo isso cria um nível de acoplamento que não aparece em diagramas** e dificilmente está documentado com precisão. E, na maioria dos casos, nem mesmo é plenamente conhecido pelas equipes responsáveis pelo sistema.

É por isso que, na prática, separar serviços é apenas parte do desafio. **O ponto mais crítico está na separação dos dados**. Porque, enquanto o código pode ser reorganizado, o banco de dados carrega consigo a história das decisões, integrações e exceções acumuladas ao longo dos anos — e essa história não se decompõe facilmente.

Sem compreender com precisão como o sistema utiliza seus dados — quem acessa o quê, de que forma e com quais dependências — qualquer tentativa de decomposição estrutural se torna frágil. O risco não está apenas em erros isolados, mas em comprometer o funcionamento do sistema como um todo.

No fim, o banco de dados deixa de ser apenas um componente técnico e passa a ser o principal fator limitante da transformação. E ignorar esse papel é, na prática, subestimar o problema mais crítico da modernização.

QUOTES INTERESSANTES:

> *Poor architecture decisions are among the top causes of system outages and business disruption*
>
> **Gartner**

Quando a arquitetura vira risco!

Até aqui, o desafio pode parecer técnico — arquitetura, dependências, banco de dados. Mas, na prática, **seus efeitos se manifestam diretamente no negócio**. Quando uma organização não compreende com precisão o funcionamento do próprio sistema, ela não apenas enfrenta complexidade — **ela passa a operar sob risco.**

O primeiro impacto é na previsibilidade. **Projetos de modernização deixam de seguir estimativas confiáveis** e passam a se estender além do planejado, consumindo mais tempo e recursos. O que deveria acelerar a capacidade de inovação acaba, muitas vezes, retardando decisões e travando iniciativas estratégicas.

O segundo impacto é operacional. Mudanças passam a carregar um nível maior de incerteza, **aumentando o risco de falhas, inconsistências e efeitos colaterais**. Em ambientes críticos, isso não significa apenas retrabalho — significa potencial interrupção de serviços, impacto em clientes e exposição da organização.

QUANDO A ARQUITETURA VIRA RISCO

"A maioria das empresas não percebe quando começa a operar sob risco. Percebe apenas quando o impacto já aconteceu."

Há ainda um efeito menos visível, mas igualmente relevante: a perda de velocidade. Quando cada decisão depende de validações adicionais, quando cada mudança exige cautela excessiva, a organização deixa de operar com fluidez. A inovação desacelera não por falta de intenção, mas por falta de confiança no sistema que sustenta o negócio.

E é nesse ponto que **a arquitetura deixa de ser um tema técnico e passa a ser um tema de governança**. Porque, no fim, não se trata apenas de como o sistema é construído — mas de quanto controle a organização realmente tem sobre ele.

Sem esse controle, a modernização não falha de forma explícita. Ela simplesmente não entrega o que prometeu.

> *More than 50% of application modernization efforts exceed budget or timeline expectations*
>
> **IDC**

QUOTES INTERESSANTES:

> *Microservices introduce distributed complexity that is harder to debug, test and govern*
>
> **Forrester**

> *Organizations that invest in observability reduce incident resolution time by up to 50%*
>
> **Gartner**

Da incerteza a decisão

Se o desafio da modernização está naquilo que não é visto — dependências, relações implícitas, uso real dos dados — **a resposta não está apenas em redesenhar a arquitetura, mas em torná-la visível**.

Organizações que conseguem avançar com mais segurança nesse processo compartilham uma característica comum: **substituem suposição por evidência**. Em vez de inferir como o sistema funciona, passam a observar, mapear e **compreender suas dependências reais — especialmente aquelas que não estão documentadas** de forma explícita.

Isso exige uma mudança de abordagem. A arquitetura deixa de ser apenas um modelo conceitual e passa a ser tratada como um ativo observável, construído a partir do comportamento real do sistema. Código, integrações e dados deixam de ser analisados de forma isolada e passam a ser compreendidos como parte de uma estrutura interdependente que precisa ser explicitada.

CONTROLE COMEÇA COM VISIBILIDADE

"Quando a organização enxerga como seu sistema realmente funciona, não apenas reduz riscos — retoma o controle sobre sua própria evolução."

É nesse contexto que surgem soluções capazes de apoiar essa jornada. Ferramentas como o **BELA (Browser for Enterprise-Level Architecture)**, da Jux, permitem **extrair e visualizar a arquitetura real a partir do próprio código,** evidenciando dependências entre serviços, componentes e, de forma crítica, o uso do banco de dados. Ao transformar relações implícitas em informação explícita, esse tipo de abordagem reduz incerteza e aumenta a capacidade de tomada de decisão.

No fim, a modernização deixa de ser um exercício de tentativa e erro e passa a ser um processo orientado por compreensão. Porque, quando uma organização enxerga com clareza como seu sistema realmente funciona, ela não apenas reduz riscos — **ela recupera o controle sobre a própria evolução.**

Saiba mais em:

folheto do **BELA**

contact-us@massimafrog.com

www.massimafrog.com

linkedin.com/company/massimafrog/