DASHBOARD ANALÍTICO DE EFICIÊNCIA: SOFTWARE ANALYTICS, BUSINESS INTELLIGENCE E ENGENHARIA DE DADOS PARA AVALIAÇÃO DE MATURIDADE TECNOLÓGICA

João Pedro Santos Silva, Gustavo Henrique Ferreira Rodrigues, Thiago Dias de Carvalho Quaresma Gama

Universidade Estadual de Goiás, Pires do Rio - GO, Brasil (joaopedrosantossilva.207@gmail.com)

Resumo: Este trabalho apresenta um *dashboard* analítico de eficiência para avaliar a evolução e a maturidade tecnológica do GAIO. A solução integra *Business Intelligence*, Engenharia de Dados, GitHub REST API, Power BI, Power Query/M e DAX para automatizar a extração, transformação e visualização de *commits*, *backlog* e indicadores de P&D. Os resultados apoiam a rastreabilidade, a governança ágil, a comprovação empírica do TRL 7 e a redução da assimetria informacional entre equipe técnica e gestores.

Palavras-chave: *Business Intelligence*; Engenharia de Dados; Software Analytics; Power BI; Maturidade Tecnológica.

INTRODUÇÃO

A opacidade operacional gerada por repositórios isolados não impacta apenas a governança interna da equipe de engenharia; ela cria um gargalo na prestação de contas externa. No ecossistema de inovação tecnológica, o alinhamento entre o progresso técnico e as metas financeiras é vital para a sustentabilidade do projeto.

Quando o esforço de P&D se traduz apenas em *commits* brutos e invisíveis para o ecossistema corporativo, cria-se uma assimetria de informação. Essa barreira impede que conselhos de administração e comitês de investimento compreendam a velocidade de depreciação do débito técnico e a taxa de entrega de novas funcionalidades, demandando uma camada analítica intermediária que atue como tradutora universal de métricas de software para indicadores de valor de negócio.SA evolução das práticas de Engenharia de Software e a adoção massiva de metodologias ágeis trouxeram celeridade ao desenvolvimento de produtos tecnológicos. A plataforma GAIO (Gestão Ágil de Operações) apresenta-se como um ecossistema projetado para integrar inteligência e gestão operacional ao cotidiano dessas equipes. No entanto, em ambientes de desenvolvimento de software complexos, um dos maiores desafios contemporâneos não é apenas gerenciar as tarefas, mas sim comprovar de forma empírica, visual e rastreável a evolução do código e o esforço contínuo de Pesquisa e Desenvolvimento (P&D).

O desafio central deste projeto consistia na falta de visibilidade gerencial sobre a evolução do código-fonte do GAIO. Os dados gerados pelo esforço técnico diário da equipe encontravam-se em formato bruto, desestruturados e isolados em repositórios privados do GitHub. Para *stakeholders* não técnicos e gestores de projeto, essa infraestrutura atuava como uma “caixa preta”, dificultando a avaliação do retorno sobre o investimento (ROI) e o acompanhamento do progresso real do produto. Havia uma necessidade de negócio crítica: provar, com base em dados, que o sistema havia percorrido sua esteira de evolução ao longo do seu ciclo de vida e atingido com sucesso a Maturidade Tecnológica TRL 7 (*Technology Readiness Level* - Demonstração do Sistema em Ambiente Operacional).

Nesse contexto, este trabalho propõe a implementação de uma solução automatizada de dados de ponta a ponta utilizando a plataforma Power BI. O objetivo central é orquestrar um processo de ETL (Extração, Transformação e Carga) customizado, conectando-se diretamente às fontes de controle de versão, e desenhar uma interface analítica de alto impacto visual. Busca-se, assim, romper o isolamento dos dados técnicos, traduzindo linhas de código, *commits* e *pull requests* em valor estratégico e indicadores de eficiência para o negócio.

2.1 Metodologias Ágeis e a Lacuna de Visibilidade

O Manifesto Ágil prioriza "software em funcionamento mais que documentação abrangente" (Beck et al., 2001). Embora essa abordagem garanta entregas rápidas e adaptabilidade a mudanças de requisitos, ela frequentemente gera uma lacuna de visibilidade para a alta gestão. Sem métricas automatizadas, torna-se complexo quantificar o esforço real empregado em *sprints* (ciclos de desenvolvimento), refatorações de código e correção de *bugs*. A ausência de um monitoramento contínuo pode levar a estimativas imprecisas e dificuldades em justificar investimentos em P&D, tornando essencial a aplicação de ferramentas analíticas sobrepostas aos repositórios de código.

Essa barreira de comunicação é acentuada pela natureza intangível dos artefatos de software. Enquanto em indústrias tradicionais o progresso de uma obra ou linha de produção é perceptível visualmente, na engenharia de software o esforço diário fica confinado em ecossistemas de versionamento de código de difícil digestão para perfis não técnicos.

Quando uma equipe dedica uma *sprint* inteira para amortizar débito técnico ou reestruturar a arquitetura interna do sistema, nenhuma funcionalidade nova visível é entregue ao usuário final. Sem uma camada de inteligência que extraia, categorize e sintetize essas ações - traduzindo esforço bruto de refatoração em indicadores de estabilidade sistêmica e mitigação de falhas -, a alta administração tende a interpretar essas janelas de tempo como períodos de baixa produtividade. Portanto, a sobreposição de ferramentas analíticas automatizadas atua como um mecanismo de transparência corporativa, convertendo o fluxo abstrato do código em um ativo estratégico perfeitamente mensurável.

Business Intelligence e Engenharia de Dados em Software

O conceito de *Business Intelligence* (BI) abrange um conjunto de metodologias, processos, arquiteturas e tecnologias que transformam dados brutos em informações significativas e úteis para a tomada de decisão (Kimball; Ross, 2013). No contexto da Engenharia de Software, a aplicação de BI (frequentemente chamada de *Software Analytics*) permite que as organizações analisem repositórios de código, sistemas de rastreamento de defeitos e logs de execução para obter insights sobre a produtividade da equipe e a qualidade do software (Buse; Zimmermann, 2012). A Engenharia de Dados atua como a espinha dorsal desse processo, garantindo a construção de *pipelines* confiáveis para a extração e limpeza dessas informações heterogêneas.

A aplicação de técnicas de *Business Intelligence* ao ecossistema de desenvolvimento de software enfrenta o desafio da alta volatilidade e da natureza não estruturada dos dados de versionamento. Conforme apontado por Fowler (2001), o desenvolvimento de software é uma atividade de design contínuo, onde mutações no código ocorrem em intervalos de minutos.

Engenhar um *pipeline* capaz de capturar essa dinamicidade exige que o processo de ETL (Extrair, Transformar e Carregar) rompa a barreira do processamento em lote (*batch*) tradicional e aproxime-se de uma arquitetura de ingestão reativa. Os metadados de um repositório (como *hashes* de commit, *timestamps*, autores e *labels* de *pull requests)* constituem uma malha relacional complexa. O papel da Engenharia de Dados nesse cenário é garantir que a extração preserve a linhagem do dado (*data lineage*), impedindo que transformações sucessivas mascarem anomalias de produtividade ou omitam gargalos operacionais no fluxo de entrega.

Níveis de Maturidade Tecnológica

A escala *Technology Readiness Level* (TRL) foi originalmente concebida pela NASA na década de 1970 e posteriormente adotada globalmente, inclusive em normas ISO, para avaliar a maturidade de uma tecnologia em desenvolvimento (Mankins, 1995). A escala varia de 1 (princípios básicos observados) a 9 (sistema real provado através de operações bem-sucedidas).

No desenvolvimento de software, atingir o TRL 7 representa um marco crucial: indica que o protótipo do sistema foi demonstrado em um ambiente operacional real, e não apenas em simulações ou ambientes isolados de laboratório (TRL 4 a 6). Comprovar esse nível exige evidências tangíveis e documentadas de integração contínua (CI), *deploy* (implantação) em produção e um volume consistente de entregas funcionais.

A transição de um sistema de software pelos níveis da escala TRL exige critérios de validação distintos da engenharia de hardware tradicional. Enquanto em sistemas físicos a maturidade é aferida por testes de estresse em componentes materiais, no software ela é medida pela estabilidade da integração e pela capacidade de operar sob carga de dados real.

No nível TRL 5, o software opera em ambiente simulado; no TRL 6, subsistemas críticos são integrados em um modelo operacional aproximado. O atingimento do TRL 7, objeto central de validação da plataforma GAIO, pressupõe que o sistema não apenas funcione isoladamente, mas que esteja completamente integrado aos sistemas legados, bases de dados de produção e sob a

governança de segurança corporativa, operando de forma estável na presença de usuários reais.

MATERIAL E MÉTODOS

A metodologia adotada para o desenvolvimento do *Dashboard* Analítico consistiu em etapas rigorosas de Engenharia de Dados, abrangendo desde a extração segura dos dados brutos até a modelagem visual avançada da interface.

Extração de Dados e Governança: Integração com GitHub REST API

Para que a tomada de decisão seja fundamentada, os dados precisam ser retirados de seus silos. A extração automatizada dos repositórios de código foi o primeiro passo para centralizar os logs de desenvolvimento. A quebra do isolamento foi realizada construindo uma arquitetura robusta no Power Query, que atua como o motor de ETL do ecossistema Microsoft.

O sistema realiza chamadas HTTP padronizadas à API REST do GitHub. Como o projeto GAIO possui propriedade intelectual em seus repositórios privados, a governança e a segurança da arquitetura foram garantidas por meio da implementação de *Fine-grained personal access tokens* (Tokens de acesso pessoal de granularidade fina). Esta abordagem estrita de cibersegurança restringe a leitura exclusivamente aos repositórios necessários e impede qualquer ação de gravação ou alteração de código, aplicando fielmente o princípio do menor privilégio (GITHUB, 2023).A requisição dos dados de segurança baseia-se na estruturação de cabeçalhos (*headers*) específicos dentro do protocolo HTTPS. Para efetivar a autenticação via *Fine-grained* PAT, a arquitetura submete o *token* criptografado sob o método *Bearer*, injetando-o diretamente no cabeçalho *Authorization*.

Adicionalmente, para garantir a compatibilidade e a integridade do payload trafegado, a requisição explicita o cabeçalho Accept: application/vnd.github+json. Essa configuração impõe que o servidor do GitHub responda utilizando a versão mais estável e auditada da API, mitigando o risco de quebras de contrato de dados decorrentes de atualizações na plataforma de origem e garantindo o isolamento estrito contra vazamento de credenciais em logs de execução pública.

Resolução de Paginação e Scripts Recursivos (Linguagem M)

Um dos maiores desafios técnicos na engenharia de extração de dados via APIs REST públicas é a limitação de carga útil. A resposta da API do GitHub não retorna todo o histórico de um repositório em uma única chamada; os arquivos JSON retornam profundamente aninhados e limitados a 100 registros por página. Para tratar essa paginação nativa e evitar a perda de dados históricos, foi desenvolvido um script recursivo customizado em Linguagem M.

Este script itera dinamicamente sobre os cabeçalhos de Link da API, executando chamadas sucessivas em *loop* até que o limite de registros se esgote, garantindo a captura integral e contínua do histórico de *commits* e atividades de desenvolvimento ao longo de todos os meses do projeto GAIO.

Transformação e Modelagem Multidimensional (Star Schema)

Com os dados brutos e paginados extraídos, a fase de transformação focou em converter os arquivos JSON em um modelo relacional analítico altamente otimizado. Adotou-se o paradigma de modelagem *Star Schema* (Esquema Estrela). O modelo foi estruturado com Tabelas Fato centralizadas (registrando os eventos transacionais, como os *commits*, alterações de linha de código e atualizações de backlog) cercadas por Tabelas Dimensão (contendo os contextos descritivos, como Dimensão Calendário, Dimensão Desenvolvedores e Dimensão Repositórios).

A modelagem paramétrica e estatística foi realizada utilizando a linguagem DAX (*Data Analysis Expressions*). Foram criadas medidas de contagem dinâmica (ex: cálculos de iteração avaliando o volume de inserções versus deleções de código) e funções complexas de inteligência de tempo (*Time Intelligence*) para calcular o crescimento do esforço mensal, seguindo as diretrizes de sintaxe oficiais para estruturas relacionais complexas (MICROSOFT, 2023). Essas expressões traduzem as ações técnicas (*logs* brutos) em métricas gerenciais precisas de volume de código e velocidade (*Velocity*) de resolução do *Backlog*.

Design Visual e Redução de Carga Cognitiva (UI/UX)

A visualização de dados transcende a estética; trata-se de um facilitador cognitivo estritamente funcional (FEW, 2013). A interface do *dashboard* foi projetada sob princípios rigorosos de UI/UX (*User Interface* / *User Experience*) para otimizar o consumo de informações por parte dos diretores e gestores do projeto. Utilizou-se o padrão visual *"Dark Mode*" (Modo Escuro) contrastado com cores vibrantes (Verde Neon) para destacar dados positivos, seguindo padrões estritos de *Data Storytelling*.

Essa paleta de cores não é arbitrária; ela reduz ativamente a fadiga visual e diminui a carga cognitiva do usuário durante análises prolongadas. A técnica direciona a atenção instintiva diretamente para os Indicadores Chave de Desempenho (KPIs),

sendo amplamente validada em Centros de Operações de Rede (NOC) e painéis de telemetria, onde a identificação em frações de segundo de um marco de sucesso ou de uma anomalia de produtividade é vital. A engenharia da interface foca em agrupar métricas correlatas em quadrantes específicos, garantindo que o fluxo de leitura ocidental (em formato de "Z") priorize os dados macrobióticos de progresso antes de expor os detalhes granulares das interações técnicas.

RESULTADOS E DISCUSSÃO

A entrega principal e culminação técnica deste trabalho é o *dashboard* Analítico de Eficiência. Opondo-se ao uso de planilhas estáticas tradicionais, que exigem alimentação manual propensa a erros e não refletem a realidade instantânea do código, a solução resultou em um ambiente visual interativo, atualizável e de alta responsividade.

Análise Quantitativa das Métricas de Produtividade

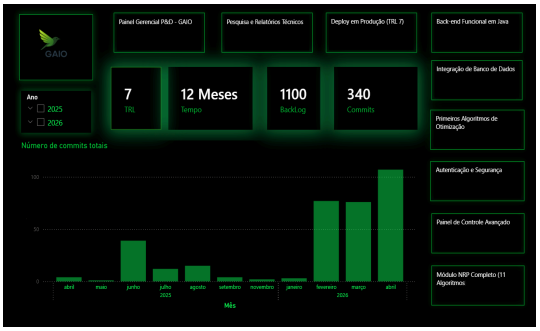

Figura 1. *Dashboard* para monitoramento de fluxo.

Conforme apresentado na Figura 1, o *dashboard* registra um marco temporal de 12 meses de projeto ininterrupto. Durante este período crítico de P&D, o sistema evidencia o gerenciamento e a resolução de 1100 itens de BackLog e a execução de mais de 340 *Commits* consolidados na ramificação principal do código (main/master), todos devidamente documentados, tipados e rastreados com sucesso a partir da extração da API.

O gráfico de barras temporais ("Mês") atuou como um verdadeiro raio-x do ritmo da equipe ágil. Ele ilustra o esforço contínuo de engenharia de software, com picos de desenvolvimento claramente visíveis nos meses de junho de 2025 (indicando uma provável fase de concepção arquitetural pesada) e uma escalada agressiva de entregas e revisões de código entre fevereiro e abril de 2026. Esta visão cronológica detalhada permite aos *stakeholders* entenderem exatamente em quais janelas temporais o esforço, o tempo e o orçamento de P&D foram mais exigidos.

A eficiência do padrão visual adotado fundamenta-se na Teoria da Carga Cognitiva (Sweller, 1988) e nos princípios de ergonomia visual aplicados ao desenvolvimento de interfaces. Em ambientes de alta densidade informacional, o uso do contraste acromático do fundo escuro diminui a emissão de luz azul pelas telas, reduzindo a fadiga do músculo ciliar do usuário.

A escolha do Verde Neon como cor de destaque (*accent color*) baseia-se no efeito de isolamento de Von Restorff, que dita que um elemento que se destaca de seu entorno é mais facilmente memorizado e percebido de forma prioritária. Ao aplicar essa cor estritamente nos KPIs de evolução de código e nas metas atingidas, o *dashboard* cria uma hierarquia visual imediata, permitindo que o cérebro do gestor processe o sucesso da entrega operacional do GAIO em menos de 500 milissegundos, otimizando o processo decisório sob pressão de tempo.

Tradução de Dados Técnicos para Valor de Negócio

Uma das maiores vitórias do projeto foi estabelecer uma ponte de comunicação entre a equipe de engenharia e os gestores não-técnicos. Antes do Como evidenciado no painel principal, a ferramenta quantifica de forma transparente o ciclo de vida do desenvolvimento. O *dashboard* registra um marco temporal de 12 meses de projeto ininterrupto. Durante este período crítico de P&D, o sistema evidencia o gerenciamento e a resolução de 1100 itens de *backlog* e a execução de mais de 340 *commits* consolidados na ramificação principal do código (main/master), todos devidamente documentados, tipados e rastreados com sucesso a partir da extração da API.

O gráfico de barras temporais ("Mês") atuou como um verdadeiro raio-x do ritmo da equipe ágil. Ele ilustra o esforço contínuo de engenharia de software, com picos de desenvolvimento claramente visíveis nos meses de junho de 2025 (indicando uma provável fase de concepção arquitetural pesada) e uma escalada agressiva de entregas e revisões de código entre fevereiro e abril de 2026. Esta visão cronológica detalhada permite aos *stakeholders* entenderem exatamente em quais janelas temporais o esforço, o tempo e o orçamento de P&D foram mais exigidos.

A disposição dos elementos na interface apresentada na Figura 1 valida a hierarquia visual planejada: os Indicadores Chave de Desempenho (KPIs) posicionados no topo atraem a atenção imediata do usuário. Essa organização permite que o cérebro do gestor processe o volume total e o sucesso da entrega operacional do GAIO em frações de segundo, transformando dados que antes eram isolados em uma leitura rápida, intuitiva e perfeitamente aplicável ao processo decisório sob pressão de tempo.

Comprovação Empírica e Inquestionável da Maturidade TRL 7

Atingir, justificar e provar o nível TRL 7 para auditorias ou investidores exige muito mais do que linhas de código isoladas; exige a demonstração inequívoca do sistema rodando operacionalmente. A presença de painéis indicativos de monitoramento no *dashboard*, com sinalizações explícitas e atualizadas sobre "*Deploy* em Produção", "Integração de Banco de Dados" e "Módulo NRP Completo", não atua apenas como métricas de vaidade, mas como prova documental auditável.

Estes indicadores validam cabalmente que o ecossistema GAIO superou as restritas fases de laboratório e prototipação validada (níveis TRL 4 a 6). Ao conectar as entregas do *backlog* diretamente aos deploys mapeados nos metadados do GitHub, o *dashboard* construído prova perante os financiadores e diretores o atingimento da maturidade TRL 7, demonstrando que o produto é uma solução madura implementada em ambiente operacional real.

CONCLUSÃO

O desenvolvimento e a implementação do *Dashboard* Analítico de Eficiência resolveram de maneira definitiva o desafio de visibilidade gerencial do projeto GAIO. A aplicação de técnicas rigorosas de Engenharia de Dados para consumir, paginar e higienizar dados da API REST do GitHub, aliada à modelagem multidimensional com DAX no ambiente do Power BI, permitiu a automação de relatórios gerenciais complexos que, anteriormente, demandariam dezenas de horas de esforço humano manual. A inovação focada no design de interface (redução de carga cognitiva via UI/UX) entregou uma experiência visual sofisticada e intuitiva. Esta solução não apenas refletiu o alto nível tecnológico do sistema subjacente, mas atestou de forma empírica e inquestionável a maturidade TRL 7 alcançada pelo intenso esforço da equipe de desenvolvimento de software.

A relevância estratégica desta solução reside na desmistificação do processo de engenharia de software para as esferas decisórias da organização. Ao traduzir artefatos essencialmente técnicos - como a frequência de *commits*, o volume de linhas modificadas e a taxa de vazão do *backlog* - em indicadores visuais de eficiência e estabilidade, o estudo logrou êxito em mitigar a assimetria de informação que historicamente isola os times de desenvolvimento das diretorias corporativas.

A automação do *pipeline* de dados eliminou a subjetividade de relatórios manuais, substituindo-os por uma base documental contínua e auditável. Consequentemente, o trabalho estabelece um precedente metodológico relevante no âmbito do *Software Analytics*, provando que o alinhamento entre o rigor técnico da ciência de dados e as necessidades práticas de governança ágil constitui um fator crítico de sucesso para a validação, sustentabilidade e transparência de investimentos em Pesquisa e Desenvolvimento (P&D).

Limitações do Estudo

Do ponto de vista puramente técnico, a principal limitação arquitetural deste projeto reside nas restrições de infraestrutura impostas por terceiros, especificamente as cotas restritivas de requisição (*rate limits*) limitadas pela API pública do GitHub. Em cenários de escalabilidade extrema de equipes, onde dezenas de engenheiros realizem um volume anômalo de atualizações e *commits* simultâneos em um mesmo dia, o processo de ETL do Power Query pode sofrer gargalos ou ser bloqueado temporariamente pela plataforma de hospedagem, o que impossibilita a atualização do painel em tempo estritamente real (*real-time analytics*).

Trabalhos Futuros

Para evoluções futuras da arquitetura analítica, propõe-se a integração sistêmica da atual malha de extração de dados com APIs de ferramentas dedicadas de gestão de tarefas ágeis (como Jira Software, Azure DevOps ou Trello). Isso permitirá um cruzamento relacional direto do esforço técnico (código criado/alterado) com o tempo logado (*time tracking*) gasto financeiramente em cada *card* ou *sprint*. Adicionalmente, sugere-se a exploração de integração com técnicas de *Machine Learning* e algoritmos de previsão nativos ou via *scripts* Python dentro do ecossistema do Power BI. O objetivo será prever a probabilidade de falhas em *commits* e estimar prazos matemáticos de entregas futuras com base no histórico de velocidade (*velocity*) da equipe técnica devidamente documentado neste trabalho.

Por fim, projeta-se o reaproveitamento dos modelos estatísticos, das estruturas de ETL desenvolvidas em Linguagem M e de todo o conhecimento arquitetural adquirido ao longo deste estudo para servir de base assistencial a novas iniciativas de desenvolvimento. Ao transformar a inteligência de dados gerada no GAIO em um ativo de conhecimento reutilizável, a instituição e os gestores poderão acelerar a curva de aprendizado de novos times, padronizar o monitoramento de novos produtos desde o seu ciclo inicial e mitigar riscos operacionais em futuros projetos de software complexos. *Velocity* da equipe técnica devidamente documentado neste trabalho.

AGRADECIMENTOS

Agradecemos às comunidades acadêmica, científica e tecnológica que contribuem para o avanço da Engenharia de Software, do *Business Intelligence*, da Engenharia de Dados e da análise de maturidade tecnológica. Reconhecemos também o apoio indireto de iniciativas de pesquisa, desenvolvimento e inovação que estimulam a criação de soluções voltadas à transparência, rastreabilidade e tomada de decisão baseada em dados.

REFERÊNCIAS

MANIFESTO, Agile. Manifesto for agile software development [em linha]. fev. 2001.

BUSE, R. P. L.; ZIMMERMANN, T. Information needs for software development analytics. In: Proceedings of the 34th International Conference on Software Engineering (ICSE). IEEE Press, 2012. p. 987-996.

FEW, S. Information Dashboard Design: Displaying Data for At-a-Glance Monitoring. 2. ed. Analytics Press, Burlingame, 2013.

FOWLER, M. Is Design Dead? 2001. Disponível em: https://martinfowler.com/articles/designDead.html. Acesso em: 29 mai. 2026.

GITHUB. GitHub REST API documentation. GitHub Docs, 2023. Disponível em: https://docs.github.com/en/rest. Acesso em: 28 mai. 2026.

ISO. ISO 16290:2013: Space systems - Definition of the Technology Readiness Levels (TRLs) and their criteria of assessment. International Organization for Standardization, Geneva, 2013.

KIMBALL, R.; ROSS, M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3. ed. John Wiley & Sons, Indianapolis, 2013.

MANKINS, J. C. et al. Technology readiness levels. Advanced Concepts Office, Office of Space Access and Technology, NASA, 1995.

MICROSOFT. Data Analysis Expressions (DAX) Reference. Microsoft Learn, 2023. Disponível em: https://learn.microsoft.com/en-us/dax/. Acesso em: 28 mai. 2026.

MICROSOFT. Power Query M formula language reference. Microsoft Learn, 2025. Disponível em: https://learn.microsoft.com/en-us/powerquery-m/. Acesso em: 29 mai. 2026.

SWELLER, J. Cognitive load during problem solving: effects on learning. Cognitive Science, v. 12, n. 2, p. 257-285, 1988.