O artigo argumenta que **não existe uma "bala de prata"** (silver bullet) capaz de resolver de forma mágica os problemas fundamentais da engenharia de software, como atrasos, estouros de orçamento e falhas de qualidade.
Ele diferencia dois tipos de dificuldades:

- **Essenciais (essence):** inerentes à natureza do software – complexidade, conformidade, mutabilidade (changeability) e invisibilidade. Esses aspectos tornam o desenvolvimento de software intrinsecamente difícil e não podem ser eliminados por nenhuma técnica ou ferramenta.
- **Acidentais (accidents):** dificuldades ligadas às ferramentas e métodos usados (como programação em baixo nível ou lentidão no tempo de resposta). Estas já tiveram avanços

significativos com linguagens de alto nível, ambientes de programação integrados e timesharing.

Ele analisa várias propostas tidas como "salvadoras", como **linguagem Ada, programação orientada a objetos, inteligência artificial, sistemas especialistas, programação automática, programação gráfica, verificação formal, novos ambientes e estações de trabalho**.

Ele conclui que todas trazem ganhos incrementais, mas nenhuma revolução que elimine a essência da dificuldade.

Por outro lado, aponta caminhos promissores que atacam o **cerne conceitual** do problema:

- Comprar software pronto em vez de desenvolver tudo do zero.
- Refinamento iterativo de requisitos e prototipação rápida.
- Desenvolvimento incremental

("crescer" software em vez de "construir").
- Formação e valorização de **grandes designers**, capazes de criar sistemas melhores e mais simples.

A mensagem central é que **não há soluções mágicas**, apenas progresso contínuo, disciplinado e dependente do talento humano.

palavras aprendidas:

- **Accidental difficulties** – dificuldades não inerentes ao software, ligadas a ferramentas ou processos.
- **Essence** – essência; refere-se às dificuldades intrínsecas do software.
- **Changeability** – mutabilidade; tendência de o software ser constantemente modificado.
- **Conformity** – conformidade;

necessidade de o software se adaptar a sistemas e instituições externas.

- **Invisibility** – invisibilidade; dificuldade de visualizar software de forma clara, diferente de máquinas ou construções físicas.
- **Inference engine** – mecanismo de inferência; componente de sistemas especialistas que aplica regras para chegar a conclusões.