Com base nos exemplos discutidos, pense sobre:

1. Quais são as características comuns a esses softwares?

Tanto os bons quanto os ruins compartilham algumas características fundamentais, como:

* Complexidade: Todos envolvem sistemas complexos, com grande responsabilidade funcional.
* Interação com usuários ou sistemas críticos: Muitos atuam em ambientes onde erros têm impacto direto na vida das pessoas ou em finanças.
* Dependência de confiabilidade: A maioria precisa operar com alto grau de precisão e segurança.
* Desenvolvimento sob pressão: Prazos, custo ou marketing influenciam decisões técnicas.

1. Quais aspectos definem as questões de qualidade?

Com base nas normas ISO (9126, 25010) e na literatura (Pressman, Hirama), destacam-se os seguintes aspectos:

* **Funcionalidade**: O sistema realiza corretamente as funções para as quais foi projetado?
* **Confiabilidade**: Quão consistente é o software frente a diferentes condições de uso?
* **Usabilidade**: É fácil de usar e entender?
* **Eficiência**: Faz uso adequado dos recursos (tempo, memória)?
* **Manutenibilidade**: É fácil corrigir, adaptar e evoluir?
* **Segurança**: Protege contra falhas e acessos indevidos?
* **Portabilidade**: Pode ser usado em diferentes plataformas?

Além disso, **expectativas dos stakeholders** (usuários, desenvolvedores, gestores, clientes) influenciam a percepção de qualidade.

1. O que poderia ser feito para melhorar?

Diversas melhorias podem ser aplicadas:

* Testes mais abrangentes e em camadas (unitários, integração, sistema, aceitação).
* Revisões independentes de código e projeto.
* Documentação clara e padronizada.
* Validação e verificação rigorosas, inclusive com usuários finais.
* Sistemas de redundância e *failover,* especialmente em sistemas críticos.
* Interfaces mais intuitivas e com comunicação clara de erros.
* Uso de padrões e frameworks reconhecidos (boas práticas da engenharia de software).

1. Onde os pontos falhos poderiam ser corrigidos?

Os pontos falhos geralmente se concentram em:

* Conversões e manipulações de dados sensíveis.
* Interfaces de operação mal projetadas.
* Ausência de testes em cenários críticos e raros.
* Excesso de confiança no software e negligência na redundância de hardware.
* Pressa na entrega sem garantir qualidade mínima.
* Falta de cultura de qualidade no ciclo de vida do software (desde os requisitos até a manutenção).