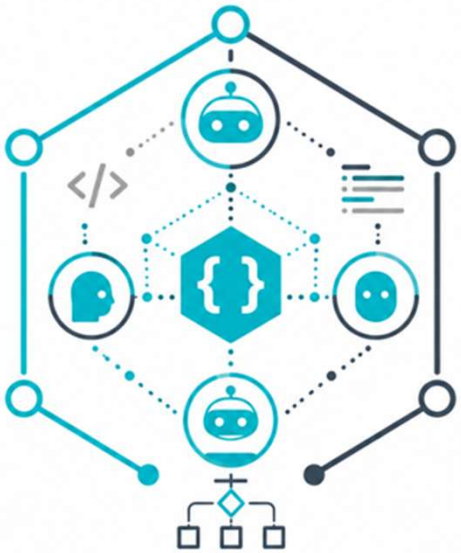


WORKSHOP

Engenharia de Software Apoiada por Agentes Artificiais

Universidade do Minho, Campus de Gualtar, Braga
2-3 de junho de 2026

accenture uphold F3M RETAILCONSULT Checkmarx VisionWare INESCTEC
keep. Cachapuz eurotux Kelvin ponto25

Engenharia de Software Assistida por Agentes Artificiais

Orlando Belo

Departamento de Informática, Escola de Engenharia, Universidade do Minho
PORTUGAL

2 de Junho de 2026

Resumo

O aumento da complexidade dos sistemas de software e a necessidade de ciclos de desenvolvimento mais curtos, mais eficientes, tem impulsionado a integração da Inteligência Artificial Generativa (IAG) no processo de programação. Inicialmente aplicada como ferramenta de apoio, a IAG evoluiu para desempenhar um papel mais ativo e estratégico no desenvolvimento de software. Os agentes de codificação resultaram dessa evolução. Um agente de codificação baseado em IAG é um sistema inteligente capaz de compreender objetivos de alto nível, planear etapas de implementação e gerar, modificar e otimizar código de forma semi-autónoma ou autónoma. A utilidade dos agentes de codificação assenta na sua capacidade de aumentar a produtividade, automatizar tarefas repetitivas e apoiar a manutenção de sistemas existentes. Desta forma, atuam como colaboradores inteligentes no processo de Engenharia de Software.

Tópicos

complexidade dos sistemas de software
ciclos de desenvolvimento mais curtos, mais eficientes
Inteligência Artificial Generativa
ferramenta de apoio
um papel mais ativo
agentes de codificação
sistema inteligente
compreender objetivos
planear etapas de implementação
gerar, modificar e otimizar código
capacidade de aumentar
a produtividade, automatizar tarefas repetitivas e apoiar a manutenção de sistemas existentes
colaboradores inteligentes
Engenharia de Software.

WordCloud

https://www.wordclouds.com/

Organização

- Engenharia de Software
- O Ciclo de Vida do Desenvolvimento de Software (SDLC)
- A Evolução do Processo de Software
- O Papel da IAG no SDLC
- A Evolução das Ferramentas
- Agentes de Codificação
- Comentários Finais

A Engenharia de Software

- A Engenharia de Software é a disciplina que aplica princípios científicos, métodos sistemáticos e boas práticas de gestão ao desenvolvimento, manutenção e evolução de sistemas de software.
- A sua definição emergiu como uma resposta à chamada "crise do software" nas décadas de 1960 e 1970, quando projetos se tornavam cada vez mais complexos, frequentemente ultrapassando prazos, orçamentos e falhando requisitos.

Duas Definições

IEEE (Institute of Electrical and Electronics Engineers)
"The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software."

IEEE Standard Glossary of Software Engineering Terminology," in IEEE Std 610.12-1990 , vol., no., pp.1-84, 31 Dec. 1990, doi: 10.1109/IEEESTD.1990.101064.

Ian Sommerville
"Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use."

Ian Sommerville, Software Engineering, Pearson, 2015.

A Engenharia de Software

- A engenharia de software envolve um conjunto estruturado de atividades:
  - definição de contexto;
  - análise de requisitos;
  - modelação;
  - projeto de arquitetura;
  - Implementação;
  - testes e validação;
  - documentação;
  - manutenção.

A Engenharia de Software

- Num contexto organizacional, a engenharia de software é fundamental para assegurar que as soluções desenvolvidas estão alinhadas com:
  - objetivos estratégicos;
  - normas técnicas;
  - necessidades dos utilizadores.

- A base para a construção de sistemas de software robustos, escaláveis e economicamente viáveis.

SDLC

- O Ciclo de Vida do Desenvolvimento de Software (SDLC) é um processo estruturado que define as diversas fases envolvidas no desenvolvimento de software, nomeadamente planear, desenvolver, testar, implementar e manter o sistema de software.
- O objetivo do SDLC é organizar o desenvolvimento de software de forma sistemática, garantindo qualidade, controlo de custos, cumprimento de prazos e alinhamento com os requisitos do utilizador ou da organização.

SDLC

- Tradicionalmente, o ciclo de vida de desenvolvimento de software (SDLC) inclui as seguintes fases:
  - Levantamento e análise de requisitos – identificação e caracterização das necessidades dos utilizadores.
  - Projeto (*design*) - definição da arquitetura e a especificação técnica do software a desenvolver.
  - Implementação – desenvolvimento do código-fonte do software.

SDLC

- Validação e testes - validação funcional e verificação da qualidade do software desenvolvido.
- Instalação (*deploy*), quando o sistema entra em produção.
- Manutenção – acompanhamento do software após instalação e arranque, assegurando correções, melhorias e adaptações ao longo do tempo.
- (…)

A Evolução do Desenvolvimento de Software

- Nas décadas de 1970 e 1980 consolidou-se a Engenharia de Software como uma:
  - disciplina formal, introduzindo modelos estruturados, com fases bem definidas e documentação rigorosa.

- A partir desta altura, a evolução do processo foi constante. Vejamos como, vendo um pouco da sua história.

Um Pouco de História

- (…)
- 1960–1970 – Crise do software e emergência da Engenharia de Software.
- 1970–1980 – Métodos estruturados e o modelo cascata.
- 1980–1990 – Orientação a objetos.
- 2000–2010 – Metodologias ágeis.
- 2010–2020 – DevOps, CI/CD e computação em nuvem.
- 2020–(hoje) – Inteligência Artificial, automação e desenvolvimento inteligente.

Um Pouco de História

1960—1970 - Projetos tornaram-se mais complexos e difíceis de controlar. Atrasos, derrapagens orçamentais e falhas frequentes nos projetos. Necessidade de processos de desenvolvimento mais estruturados. Nasceu a Engenharia de Software.

2020—(hoje) - Ferramentas de Inteligência Artificial auxiliam os processos de geração de código, elaboração de testes e produção de documentação. Desenvolvimento assistido por modelos de linguagem. Maior segurança, privacidade e qualidade de produto. Crescimento das práticas de engenharia de plataformas e automação inteligente.

A Evolução

Décadas de 1960–1970

Programação para mainframes

Linguagens especializadas

Poucas bibliotecas

Desenvolvimento manual

Proc

Equi

Hoje

Computação em nuvem

Linguagens multipropósito

Inúmeras bibliotecas disponíveis

Automação e Inteligência Artificial

Sistemas distribu

Equipas multidis

Linguagens (1960)
FORTRAN
COBOL
ALGOL
BASIC
PL/I

Linguagens (1970)
C
PASCAL
PROLOG
Smalltalk
SQL

Linguagens (Hoje)
Python
JavaScript
Java
C#
C++
Go
Rust

O Papel da IAG

- A Inteligência Artificial Generativa (IAG) tem vindo a assumir um papel transformador no desenvolvimento moderno de software, introduzindo:
  - capacidades de geração automática de código e de documentação
  - testes e até artefactos de design a partir de instruções em linguagem natural.
  - (…)

O Papel da IAG

- A IAG não se limita a executar comandos pré-definidos, disponibiliza-nos:
  - meios e instrumentos para interpretar contexto;
  - inferir intenções e produzir soluções adaptáveis às necessidades dos utilizadores;
  - meios de apoio ao longo das diversas etapas do ciclo de vida do software.

O Papel da IAG

- Atualmente, a IAG pode atuar como um copiloto inteligente, auxiliando engenheiros de software na:
  - definição, estudo e especificação de aplicações;
  - escrita e reescrita de código-fonte;
  - deteção de erros;
  - explicação de trechos complexos;
  - geração de testes automatizados;
  - (…).

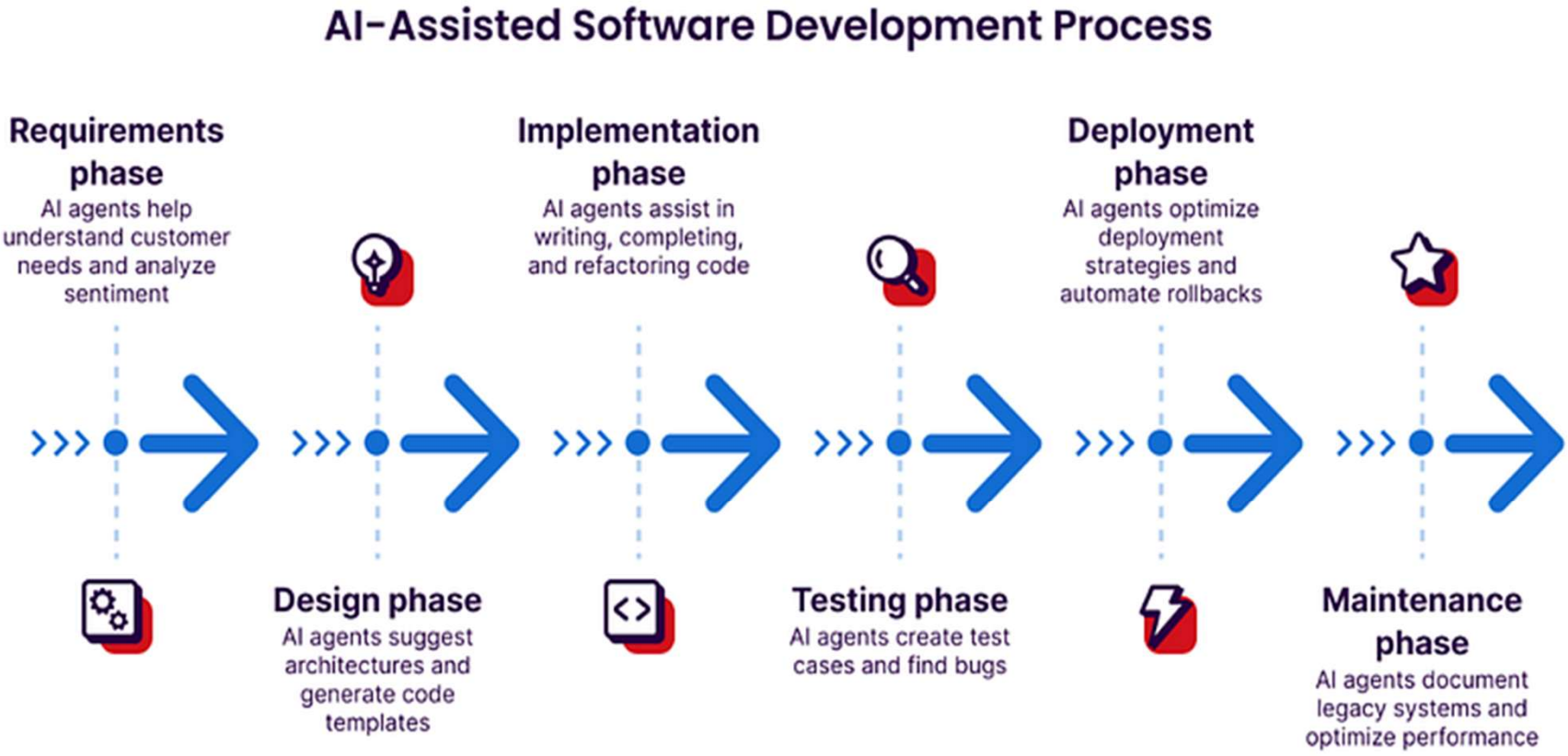
O Papel da IAG

- A IAG possibilitou o desenvolvimento de agentes de codificação (*coding agents*), capazes de planear, implementar e iterar soluções de forma semi-autónoma.
  - Uma transição do desenvolvimento puramente manual para um modelo colaborativo Homem-Inteligência Artifi ensamento crític acidades gene

A supervisão e o pensamento crítico humanos são complementados por capacidades generativas e analíticas das máquinas.

O Papel da IAG

- A IAG não substitui a engenharia de software, mas redefine práticas, competências e processos, tornando-se
  - um elemento estratégico na transformação digital e na construção de sistemas mais rápidos, inteligentes e adaptáveis;
  - um ator muito dinâmico em todas as fases do processo de desenvolvimento de software.



Fonte: https://medium.com/@joayrakesh/integrating-agentic-ai-into-the-software-development-lifecycle-sdlc-ff28ae9865da

IAG e SDLC

- Levantamento e Análise de Requisitos - Ajudar a interpretar e formalizar requisitos a partir de linguagem natural ou especificações vagas. Geração automática de modelos de casos de uso ou diagramas UML preliminares. Auxílio na validação de requisitos, identificando inconsistências ou lacunas.
- Projeto (Design) – Sugerir arquiteturas de software baseadas em padrões reconhecidos. Criação de protótipos rápidos de interfaces e fluxos. Apoio na documentação técnica.
- Implementação (Codificação) - Geração automática de trechos de código ou módulos completos. Reescrita inteligente de código legado. Codificação baseada em testes e requisitos.

IAG e SDLC

- Validação e Testes - Criação automática de testes unitários, de integração e end-to-end. Detecção e sugestão de correção de bugs. Simulação de cenários de teste complexos.
- Implementação (Deploy) - Auxílio na configuração de pipelines CI/CD. Geração de scripts de deploy ou automação de infraestrutura. Monitorização e detecção de falhas em produção.
- Manutenção - Análise contínua de logs e código para sugerir correções e otimizações. Atualização automática de documentação com base em mudanças no software. Detecção de vulnerabilidades e sugestões de mitigação.

A Evolução das Ferramentas

- Nos últimos anos, tem-se assistido à emergência dos agentes de codificação, sistemas de Inteligência Artificial e automação capazes de gerar, corrigir e otimizar código de forma autónoma ou semi-autónoma.
- Esta evolução alterou o papel do programador, que passou de escrever cada linha de código para supervisionar, orientar e validar o trabalho de agentes de codificação, enquanto surgem novos desafios relacionados com a ética, segurança e propriedade intelectual do código gerado por IA.

As Ferramentas

- Autocomplete - Sugere complementos de código enquanto o programador digita (variáveis, funções, métodos).
- Assistentes de Código - GitHub Copilot, Tabnine. Baseados em modelos de linguagem (LLMs), conseguem gerar trechos de código e ajudam na produção, documentação do software.
- Agentes de Codificação - Baseados em IAG, são apazes de agir de forma semi-autónoma, planeando, implementando, testando e iterando soluções completas.
- (...)

Agentes de Codificação

- Um agente de codificação é capaz de gerar, interpretar, modificar ou otimizar código de forma autónoma ou semi-autónoma.
  - Autonomia – executam tarefas de codificação sem intervenção contínua de um humano.
  - Adaptação – ajustam o código ou estratégias de programação com base em feedback, erros ou alterações no ambiente de desenvolvimento.
  - Colaboração – podem trabalhar junto com programadores humanos ou outros agentes, trocando informações e sugestões.
  - Eficiência e consistência – reduzem os erros repetitivos e aumentam a velocidade de produção de código.
  - Capacidade de aprendizagem – podem aprender padrões de código e estilos de programação ao longo do tempo.

Exemplos de Agentes

- GitHub Copilot (GitHub) - https://github.com/copilot - sugere código em tempo real dentro do editor, podendo gerar funções, documentação e testes.
- Claude Code (Anthropic) - https://claude.com/product/claude-code - opera diretamente sobre repositórios locais, podendo analisar código, modificar ficheiros, executar testes e responder a perguntas sobre código.
- CodexAgente (OpenAI) - https://openai.com/pt-PT/index/introducing-codex/ - capaz de implementar rotinas, corrigir bugs, executar testes e propor alterações em projetos de software.

Exemplos de Agentes

- Cursor (Anysphere) - https://cursor.com/pt-BR/home - permite modificar ficheiros através de instruções em linguagem natural, podendo compreender o contexto dos projetos
- Gemini Code Assist (Google Cloud) - https://codeassist.google/products/business - oferece geração de código, revisão e explicação de programas.
- (…)

Comentário Final

- A utilização de IAG na Engenharia de Software pode aumentar muito a produtividade, mas exige cuidados técnicos, éticos e estratégicos.
- Algumas recomendações:
  - Utilizar apenas como apoio, não como substituição do engenheiro.
  - Validar todo código gerado automaticamente.
  - Utilizar em tarefas para obter ganhos de produtividade.
  - Treinar as equipas com as devidas competências para utilizar em situações críticas.

Comentário Final

- Alguns cuidados:
  - Não enviar código sensível, credenciais ou dados proprietários para ferramentas externas.
  - O código gerado pode conter falhas.
  - Verificar possíveis conflitos de licenças.
  - Não usar de forma indiscriminado.
  - A IAG pode gerar funções inexistentes, bibliotecas incorretas ou soluções aparentemente válidas, mas erradas - alucinações.

Comentário Final

- Adotar a IAG e os agentes de codificação como copilotos técnicos, não como piloto automático.
- Apesar de serem meios expeditos que o desenvolvimento de software, a qualidade, a arquitetura, a segurança e as decisões críticas continuam sendo da responsabilidade humana.

Em 2035… Segundo o ChatGPT…

- “O papel do engenheiro estará menos centrado em "escrever código" e mais centrado em **conceber, supervisionar, validar e governar sistemas complexos**, mantendo a responsabilidade técnica e ética sobre o software produzido.”

Funções: descrever requisitos em linguagem natural, solicitar a agentes de IA a criação da solução, rever a arquitetura, segurança e conformidade do software, validar os resultados com utilizadores e stakeholders e coordenar vários agentes especializados.

Engenharia de Software Assistida por Agentes Artificiais

Orlando Belo
Departamento de Informática, Escola de Engenharia, Universidade do Minho
PORTUGAL

2 de Junho de 2026