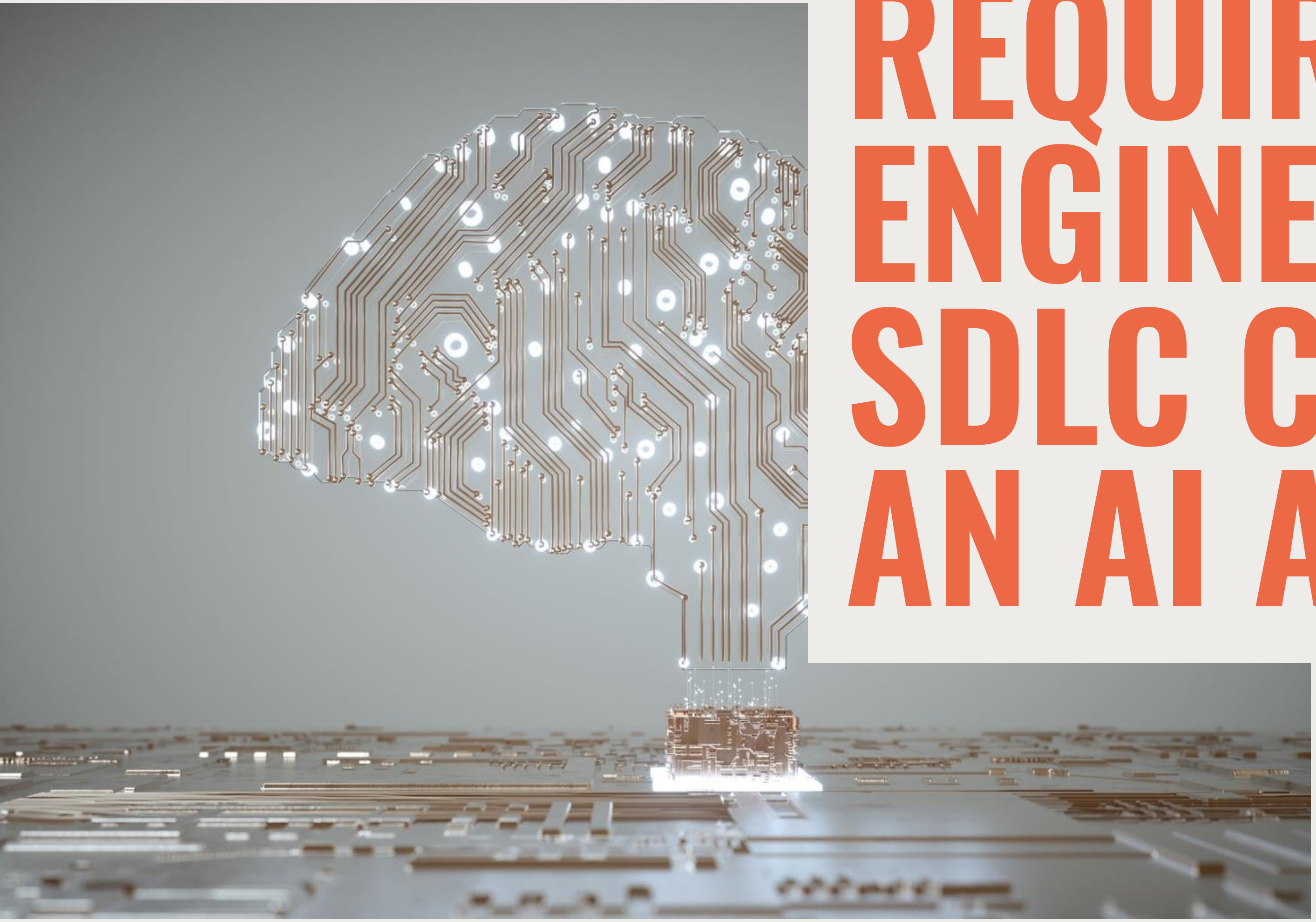
Dr Michael Glienecke

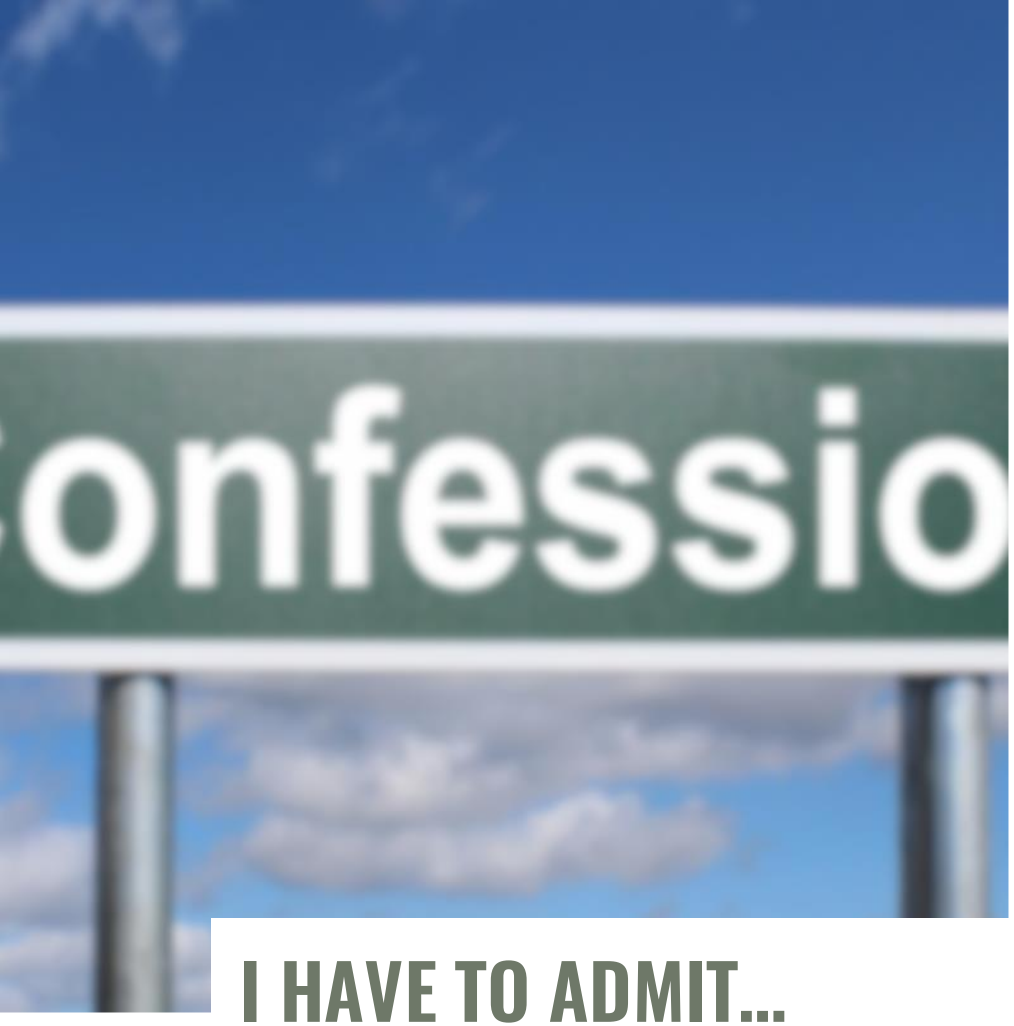
REFLECTIONS ON REQUIREMENTS ENGINEERING AND SDLC CHANGES IN AN AI AGE



TRANSFORMING DEVELOPMENT THROUGH INTELLIGENT AUTOMATION AND INNOVATION

FIRST THINGS FIRST

A CONFESSION…



I HAVE TO ADMIT…

I love software engineering

Always did, always will do. Going down, deep down, where nobody has been before…

I am no augur

I can read the trends, see the current patterns and add my own thoughts. These are mine. Don't have to be true…

AI in SE can be great (if used right)

AI is a wonderful tool. It's a tool – you need apply it correctly. Especially in SE, even more in restricted / controlled environments. Used in the wrong way it can harm and destruct; used properly it creates benefit and value.

Sometimes "out of the box" approaches (aka seemingly failures) brake new paths (Post-It, Viagra, …)

THE UNIVERSITY of EDINBURGH informatics

REQUIREMENTS ENGINEERING

Past Requirements Engineering

Current Requirements Engineering

AI Impacted Requirements Engineering

THE PAST (OR CURRENT…)

Document-Centric Approach
Heavy upfront documentation with detailed Software Requirements Specification (SRS) documents before development begins.

Waterfall Integration
Requirements phase completed entirely before design starts, following strict sequential lifecycle phases.

Formal Sign-offs
Stakeholder approval gates with change control boards managing any modifications to approved requirements.

Big Design Up Front (BDUF)
Comprehensive analysis attempts to capture all requirements before any implementation work begins.

Limited Customer Interaction
Customer involvement primarily at project start and end, with minimal feedback during development

MOSTLY NOW…

User Stories and Epics
Lightweight requirement artifacts capturing user needs from end-user perspective, enabling iterative refinement.

Continuous Discovery
Ongoing requirements elicitation throughout the project, adapting to emerging needs and customer feedback.

Product Backlog Management
Dynamic prioritization based on business value, technical dependencies, and stakeholder input.

Collaborative Workshops
Cross-functional sessions like story mapping, refinement meetings, and design thinking workshops.

Acceptance Criteria
Testable conditions defining requirement completion, enabling continuous validation and verification.

THE FUTURE?

AI-Assisted Elicitation
LLMs generate user stories, identify missing requirements, and suggest edge cases from minimal input.

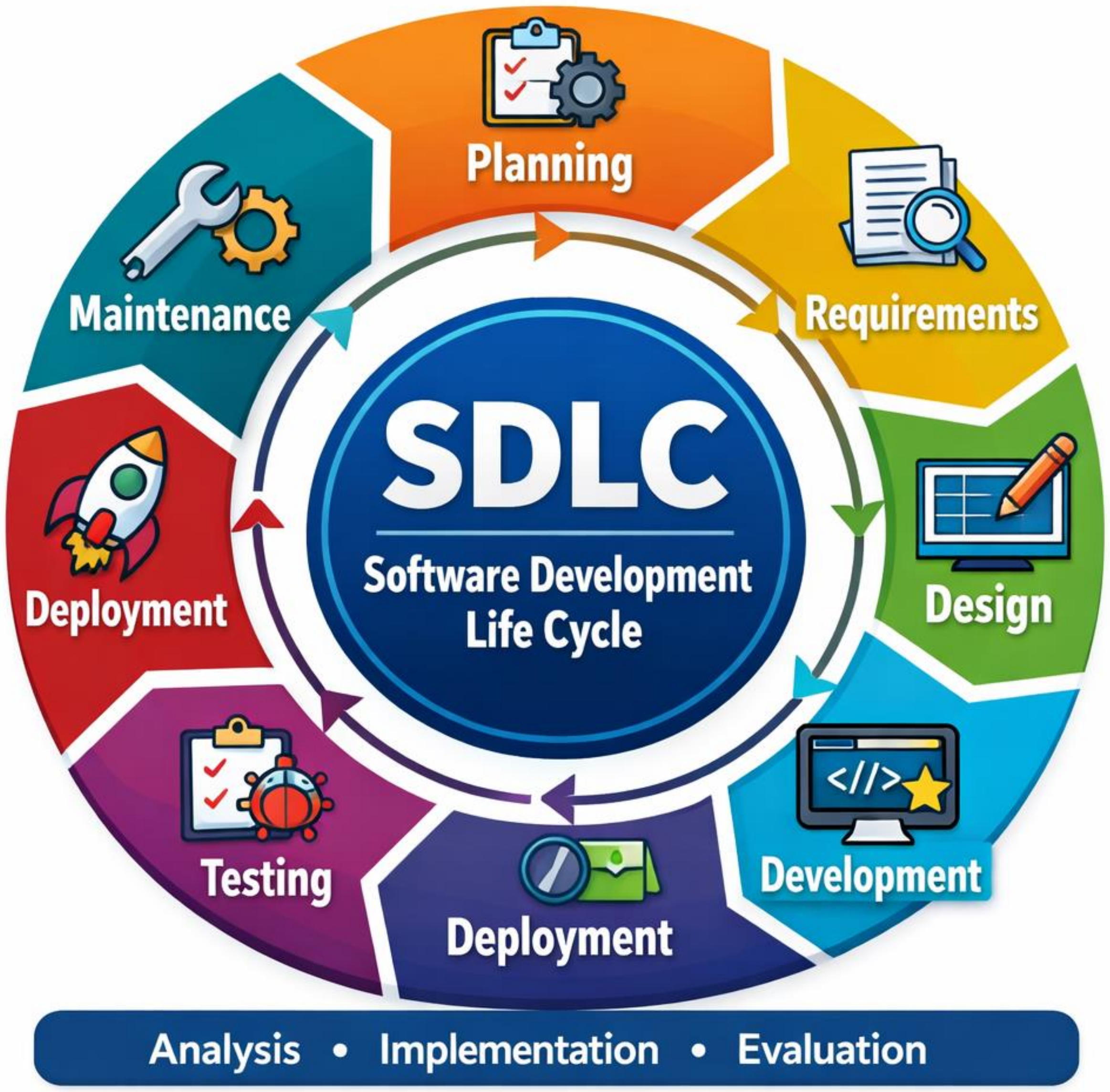
Automated Analysis
AI tools detect ambiguities, conflicts, and inconsistencies in requirement specifications automatically.

Natural Language Processing
Requirements extracted from documents, emails, meetings, and customer feedback at scale.

Intelligent Validation
AI-powered testing of requirements against design and implementation for continuous traceability.

Rapid Prototyping
AI generates mockups, prototypes, and initial code directly from requirement descriptions.

CURRENT SOFTWARE DEVELOPMENT LIFECYCLE (SDLC)


Planning
Requirements
Design
Development
Deployment
Testing
Deployment
Maintenance
SDLC
Software Development Life Cycle
Analysis • Implementation • Evaluation

OVERVIEW OF SDLC STAGES

Idea to Epic Formation

Business identifies needs, transformed into ideas and formalized as Epics by Product Management.

Story Breakdown and Estimation

Epics are broken into Stories, with developers estimating effort for effective ticket management.

Development and Code Review

Developers implement code and submit pull requests for peer review to maintain quality.

Execution and Control

Project Management controls the execution of the tickets / stories and emergencies. Compiles status reports and steering notes

Testing and Deployment

QA performs acceptance and penetration testing; DevOps manages deployment to production.

AI DISRUPTION IN SDLC

POINTS OF DISRUPTION AND CHANGE

AI Accelerates Ideation

AI tools generate prototypes rapidly, speeding up ideation and validation in product development.

AI Transforms Development

AI alters code writing, review, and maintenance, fundamentally changing software development workflows.

Project Management Challenges

AI-generated automation creates unpredictability, complicating project timeline estimation for managers. Sometimes project management becomes a team task and tickets are abandoned altogether

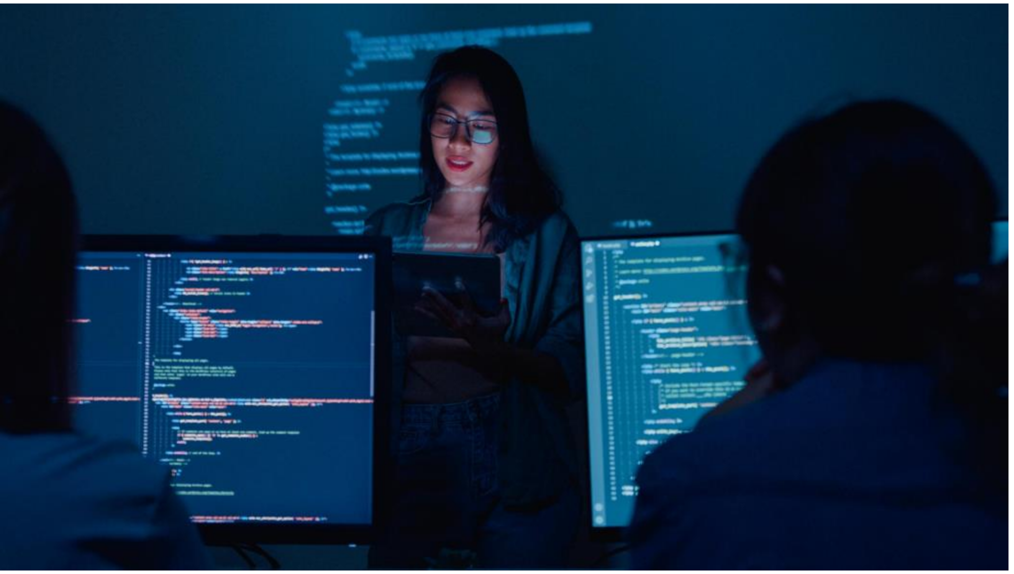
Business Investment Shifts

Businesses favor fast AI-driven solutions over foundational technologies, risking long-term sustainability.

Break early / break fast and “garbage engineering” getting ever more dominant.

THE UNIVERSITY *of* EDINBURGH informatics

DEVELOPMENT PARADIGM SHIFT



EVOLUTION OF DEVELOPMENT ROLES AND PRACTICES

Historical Development Roles

Earlier development separated system and application programmers, shaping structured roles in software creation.

Reactive AI-Driven Development

Modern development integrates AI tools like LLMs, fostering an exploratory and reactive coding approach.

Challenges with AI Integration

AI-generated outputs lack predictability and consistency, requiring new oversight and integration strategies.

Tool Support in Modern Development

The integration of AI into DEV tools (Claude / Cline, Github Copilot, JetBrains AI, etc.) adds benefit yet pose additional challenges as well.

INCIDENT AND PROBLEM MANAGEMENT

AI IN ISSUE RESOLUTION WORKFLOWS

Automation of Issue Tracking

AI enables automatic creation of development tickets from detected issues, speeding up the resolution process.

AI-Powered Development Assistance

AI development assistants suggest or implement fixes, reducing manual intervention and enhancing workflow efficiency.

Risks and Governance

Dependence on AI introduces risks that require validation and governance to ensure code quality and accountability.

AI IN QUALITY ASSURANCE

AUTOMATION AND GENERATION OF QA ARTIFACTS

Automated Unit Test Generation

AI automatically generates unit tests, increasing coverage but producing complex, hard-to-read code.

AI-Driven Component Testing

Cucumber scripts generated from Swagger endpoints enable scalable and consistent component testing.

On-the-Fly Testing Tools

AI creates testing tools dynamically, such as Python scripts to test REST services across multiple languages.

Acceptance Criteria Extraction

Acceptance criteria are extracted from screenshots, documents, or JIRA tickets to automate validation processes.

PR Review as 1st line of defense

Tools like PR Buddy help summarize pull requests, improving code review efficiency and collaboration.

UTILIZING AI TOOLS

STRATEGIES FOR EFFECTIVE AI INTEGRATION

Architectural Oversight

Architectural views help teams understand AI tool plans, providing necessary oversight during integration.

Enhanced Code Quality Checks

AI tools improve code quality by identifying issues and suggesting improvements during pull requests.

Precise Prompt Generation

Tools like Cline help generate precise prompts, reducing randomness when interacting with language models.

Structured AI Usage

Following 'don't shake the tree directly' principle ensures intentional AI use, avoiding wasteful trial and error.

THE UNIVERSITY *of* EDINBURGH
informatics

STRATEGIC TAKEAWAYS

CHALLENGES AND RISKS OF AI ADOPTION

AI supports a clearer vision and ideation

AI adoption pushes experimentation, prototyping and validation earlier in the development lifecycle creating better software (less misinterpretation).

AI promotes the Dunning-Kruger effect

AI empowers users currently outside the SDLC, often raising the stakes for those in more classical roles.

Code Quality and Maintenance

Maintaining AI-generated code raises questions on responsibility for updates and validation over time.

Complexity and Cost

Gen-AI especially with IaC tools increases complexity and operational costs, complicating AI adoption decisions.

Knowledge Retrieval Challenges

Difficulty in accessing relevant AI knowledge highlights potential for large language models as meta-retrieval layers.

THE UNIVERSITY of EDINBURGH informatics

CRITICAL THINKING ON AI TOOLS

EVALUATING APPLICABILITY AND LIMITATIONS

Critical Thinking with AI Tools

Approach AI tools thoughtfully, understanding their specific applications and inherent learning curves.

Challenges of Disruptive Tools

Disruptive AI tools require significant time and effort to master and integrate effectively.

Risks of Unstructured Experimentation

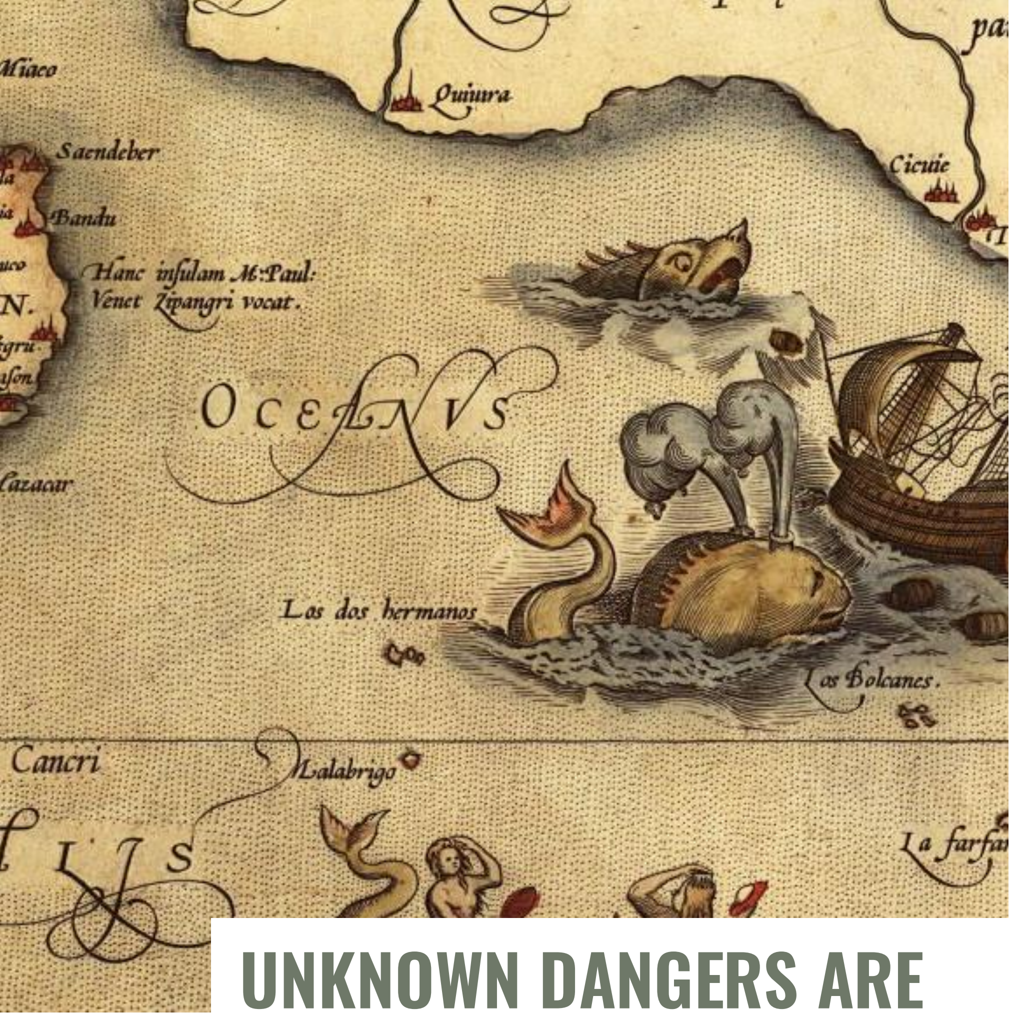
The 'Try & Pray' approach to AI often causes inefficiencies and frustration without clear goals.

Importance of Training and Evaluation

Teams must invest in training and structured evaluation to leverage AI tools effectively and avoid wasted resources.

THE UNIVERSITY of EDINBURGH informatics

HIC DRACONIS



UNKNOWN DANGERS ARE LURKING

Exposing your IP

IP – perhaps the most valuable asset – can be significantly exposed (and learned from) by utilizing external AI tools. Own hosting is, curated LLMs and offline mode (Llama, etc.) are options.

MCP servers

Many security issues are still pending and not everything which shines is gold. You can create quite an uncontrollable tools-landscape (aka Noah's arch) with too much and no proper constraint

I said "lunch" – not "launch"

Does my AI system really understand me? Can I express myself clear enough?

Think twice and measure seven times

AI needs questioning and interrogation. Especially as it is autonomous, feels omnipotent and acts like an emperor without constraint.

THE UNIVERSITY of EDINBURGH informatics