University of Essex Online

MSc Artificial Intelligence

Project Report:

Development Team Project

Submission date: 8 September 2025

Authors: Elias Medig, Mohamed Khaled Eissa Almail Alzaabi & Nikolaos Archontas

Contents

Table of Contents

Introduction

This report presents the design and architecture of a multi-agent system to automate and support business processes in the Digital Forensics domain of a fictional company. Its purpose is to define business requirements, architecture, methodology, and key challenges. Agent-based systems are well suited to this field as they combine autonomy, reactivity, and social ability (Wooldridge, 2009). Their application in corporate cybersecurity reflects the shift towards delegating repetitive monitoring tasks to intelligent systems (Russell and Norvig, 2021).

The company operates in a mixed IT environment with Windows and Linux endpoints. To support scalability and collaboration, it uses Microsoft Azure for storage and compute, integrated with the Microsoft ecosystem, including Power BI, Teams, and productivity tools.

Business Requirements

The agent must support secure file handling, metadata extraction and analysis, content inspection, and structured reporting to assist investigators. Users include law enforcement, corporate security, and compliance officers who need tools capable of scanning large file volumes, detecting forensic artefacts, and preserving evidential integrity with full auditability. In short, the agent should collect, process, and present digital forensic evidence such as log files, metadata, and network traces.

The agent should deliver:

- **File identification and collection:** Extract relevant files (Office documents, PDFs, executables).

- **Metadata analysis and integrity checking:** Capture timestamps, authorship, and apply cryptographic hashing.
- **Malware detection:** Scan for virus signatures, malware, or encrypted payloads.
- **PII detection:** Identify names, emails, phone numbers, or financial data for GDPR compliance.
- **Anonymisation and redaction:** Mask PII before storage in shared repositories.
- **Classification and tagging:** Categorise files by sensitivity or risk.
- **Audit logging:** Record all actions for full traceability.
- **Reporting and visualisation:** Generate structured reports and dashboards.
- **Secure storage:** Store outputs in encrypted repositories with access control.

These requirements reflect principles of reactivity and accountability in multi-agent systems (Brooks, 1991; Wooldridge, 2009).

Functional Requirements

The agent must identify, and extract specified file types, analyse metadata, verify integrity, detect malware, and locate and anonymise PII. Files should be classified by sensitivity, with all actions logged to ensure accountability. Processed outputs must be securely stored and presented with structured datasets and visual summaries for rapid interpretation. Automation ensures consistency and reliability, core attributes of agent-based systems (Wooldridge, 2009).

Non-Functional Requirements

The system should scale to large file volumes without loss of performance and remain accurate in detection. Security is critical: data must be encrypted, with access tightly controlled. Audit logs must be comprehensive, the system extensible for new formats or rules, and maintainable through modular updates. Compliance with GDPR and data protection policies is essential. These qualities align with layered architectures such as InteRRaP, which emphasise modularity and adaptability in dynamic environments (Russell and Norvig, 2021).

Expected Business Benefits

Automating file scanning and analysis reduces manual effort and improves speed. By detecting malware and ensuring data integrity, the agent strengthens cybersecurity and minimises breach risks. Automated PII handling enhances compliance, while classification and reporting increase transparency. Overall, the system delivers efficiency gains, cost savings, improved risk management, and greater organisational trust. Such benefits demonstrate the value of delegating operational monitoring tasks to intelligent agents (Russell and Norvig, 2021).

Category	Requirements / Benefits
Functional Requirements	• Fetch files from designated file systems • Analyse file metadata • Perform integrity checks according to digital forensics guidelines • Scan file content for sensitive data (PII) • Provide data classification • Generate structured reports • Support searching for specific file types • Log all actions to support audit trail
Non-Functional Requirements	• Scalable design • Results presented in a structured and consistent format • Compliance with privacy regulations (GDPR) • High accuracy maintained • Ensure auditability • Maintainable and extensible solution
Expected Business Benefits	• Process time reduction • Efficiency gains in identifying, collecting, and analysing artefacts • Increased consistency through repeatability and standardisation • Cost savings due to reduced manual labour

System Architecture

The proposed system uses a hybrid layered architecture modelled on InteRRaP, combining reactive, planning, and cooperative layers to balance responsiveness with reasoning (Wooldridge, 2009).

- **Reactive layer:** Provides immediate safeguards, such as malware detection or blocking suspicious access.
- **Planning layer:** Manages workflows including file collection, metadata analysis, verification, and classification.
- **Cooperative layer:** Enforces compliance rules, generates reports and dashboards, and ensures integration with external services.

Control flows upward when lower layers cannot manage an event (bottom-up activation) and downward when higher layers issue executable plans (top-down execution) (see Annex, *Activity Diagram, and High-Level Design*). This two-pass flow, central to InteRRaP, ensures responsiveness and structured reasoning (Brooks, 1991; Maes, 1991).

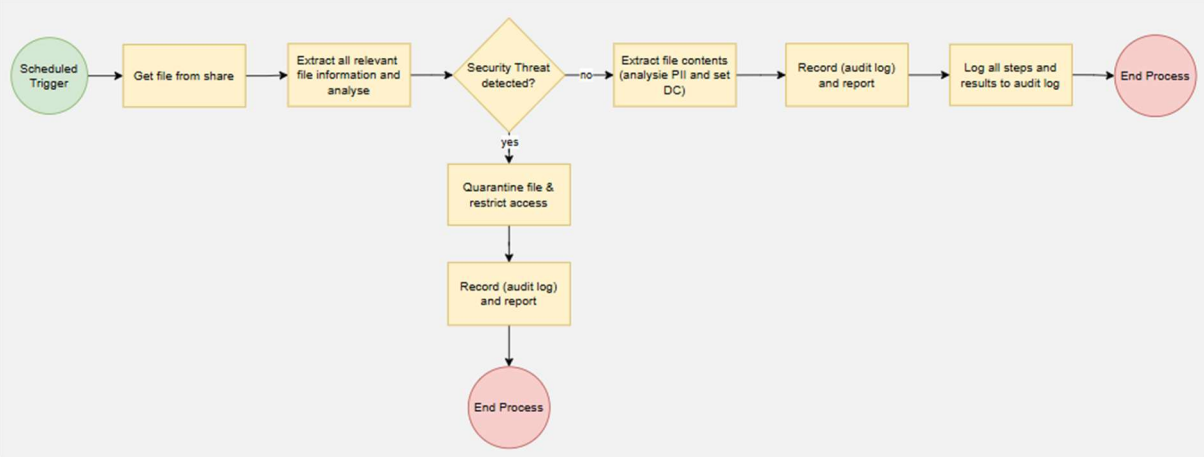

Figure 1: Activity Diagram

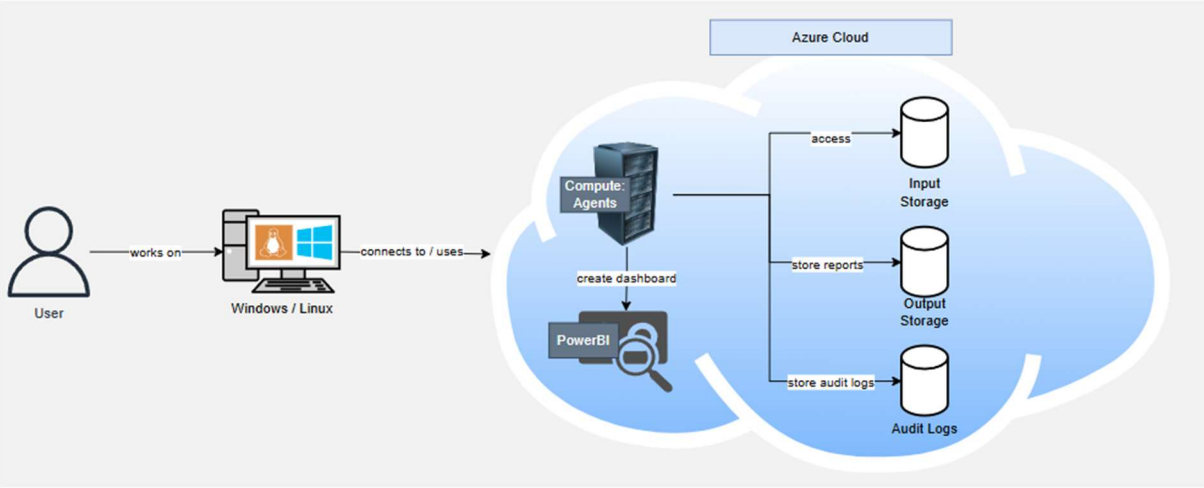

Figure 2: High Level Design

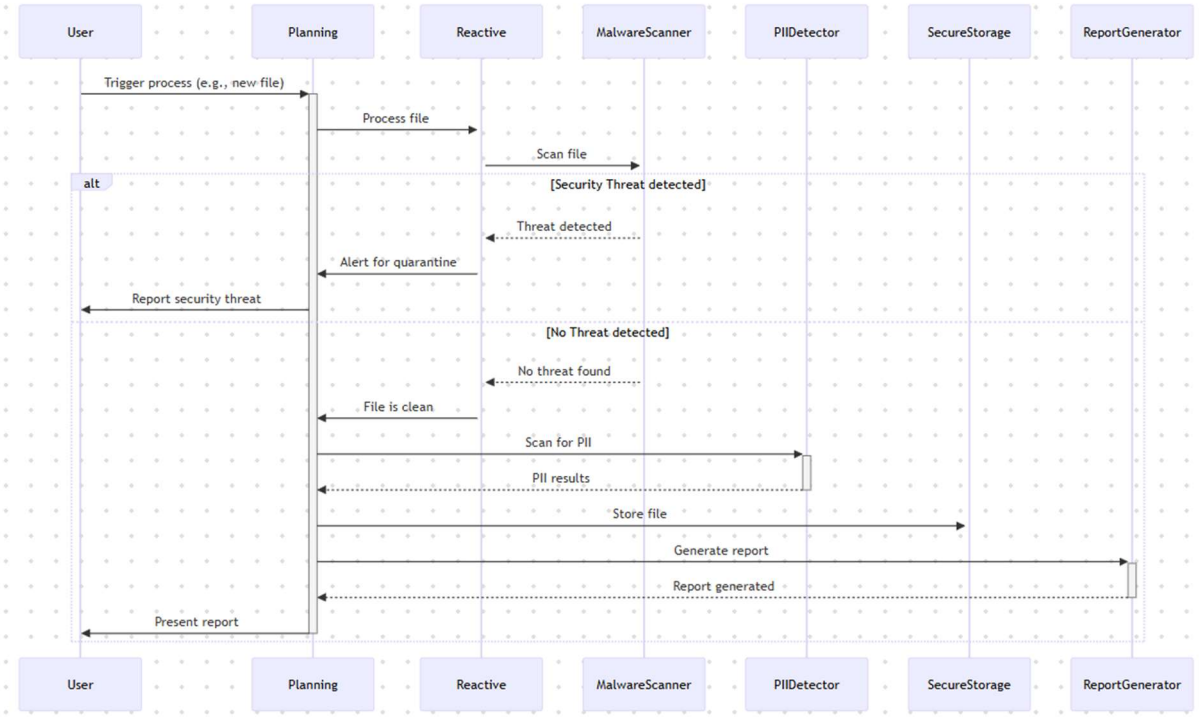

Figure 3: Sequence Diagram

Technology and Communication

Implementation will use Python, chosen for its strong ecosystem in security and data handling. Core libraries include *hashlib* for hashing, *pytsk3/dfVFS* for forensic file access, *pandas* for data processing, *matplotlib* for visualisation, *SQLite* for

lightweight encrypted storage. This stack ensures scalability, auditability, and compliance while reducing development risk (Wooldridge, 2009).

For communication, the design adopts KQML/Knowledge Query and Manipulation Language, which uses performatives such as *inform*, *request*, and *query* to express message intent (Finin et al., 1994; Searle, 1969). Although KQML lacks strict semantics and transport definitions, limiting interoperability, its performatives remain useful for clear agent interactions when paired with modern protocols.

Development Methodology

The project will follow an iterative, incremental approach inspired by agile principles. Development will begin with core functions such as file retrieval and hashing, before extending to malware detection, PII handling, and reporting. Each iteration delivers testable components validated against requirements and refined by feedback. This approach reduces complexity risks and fits the modular, layered architecture (Wooldridge, 2009).

Challenges & Justification

Technical challenges include managing large file volumes, addressed through efficient parsing and sampling. Ethical issues concern user privacy, requiring GDPR-compliant detection, anonymisation, and secure storage of PII. Legal challenges involve maintaining chain of custody for admissible evidence, addressed through audit logging and immutable action records. These reflect best practices in agent-based design, where modularity, accountability, and compliance are vital (Wooldridge, 2009).

Scholarly reviews highlight that AI-enabled digital forensics remains uneven, partly due to the absence of standardised datasets and reproducible benchmarks, which hinders comparability across tools (Ragho and Chaudhari, 2025). Although deep learning offers accuracy gains, its opacity and adversarial fragility undermine forensic reliability (Fattahi, 2024). These issues underscore the need for explainable AI, adversarial testing, and shared evaluation standards.

Critically, InteRRaP offers significant strengths by combining reactive safeguards with planning and cooperative reasoning, making it well suited to forensic tasks in dynamic environments (Wooldridge, 2009). Yet layered architectures can increase design complexity and lack the formal semantics of purely logic-based approaches (Russell and Norvig, 2021). Despite these trade-offs, InteRRaP remains a pragmatic choice for balancing responsiveness, planning, and accountability.

Conclusion

This project proposed the design of an autonomous agent to support digital forensics in a corporate cybersecurity setting. Business, functional, and non-functional requirements were defined, centred on secure file handling, metadata analysis, malware detection, PII anonymisation, and reporting. These were translated into a hybrid layered architecture inspired by InteRRaP, combining reactive safeguards, workflow planning, and cooperative compliance enforcement.

The Python-based technology stack with specialist forensic libraries makes the design both robust and feasible. KQML provides clarity in agent communication design, complemented by modern protocols for interoperability.

The system offers clear benefits: reduced manual effort, improved consistency, enhanced GDPR compliance, and strengthened organisational trust. Key challenges remain in data scalability, privacy protection, and evidential integrity, but these are mitigated through efficient data processing, encryption, and comprehensive audit logging. Literature further highlights the need for explainable and reproducible AI practices (Ragho and Chaudhari, 2025; Fattahi, 2024).

Overall, the agent demonstrates how intelligent systems can automate complex forensic tasks, offering a scalable and pragmatic solution that supports efficiency, compliance, and accountability (Russell and Norvig, 2021).

References

Brooks, R. A. (1991). “Intelligence Without Representation.” *Artificial Intelligence*, 47(1–3), 139–159.

Fattahi, J. (2024). *Machine Learning and Deep Learning Techniques Used in Cybersecurity and Digital Forensics: a Review*. arXiv:2501.03250 [cs.CR], submitted 24 December 2024. Available at: arXiv.org [Accessed on2 September 2025].

Finin, T., Fritzson, R., McKay, D., & McEntire, R. (1994). “KQML as an agent communication language.” *CIKM*.

Searle, J. R. (1969). *Speech Acts*. CUP.

Maes, P. (1991). “The Agent Network Architecture (ANA).” *SIGART Bulletin*, 2(4), 115–120.

Ragho, S. R. & Chaudhari, N. (2025). *Artificial Intelligence in Digital Forensics: A Review of Cyber-Attack Detection Models and Frameworks*. *Journal of Information Systems Engineering and Management*, 10(57s), published 19 July 2025. Available at: JISEM-Journal.com [Accessed on 2 September 2025].

Wooldridge, M. J. (2009). *An Introduction to Multiagent Systems*. Wiley.

Appendix

Activity Diagram — Digital Forensics Workflow

Activity diagram showing the end-to-end workflow of the forensic agent, from file discovery through integrity checks, malware scanning, PII detection, anonymisation, and secure storage to final reporting.

flowchart TD

Mermaid syntax:

A[Scheduled Trigger] --> B[Get file from share]

B --> C[Extract all relevant file information and analyse]

C --> D{Security Threat detected?}

D -- Yes --> E[Quarantine file & restrict access]

E --> F[Record (audit log) and report]

F --> G[End Process]

D -- No --> H[Extract file contents (analyse PII and set DC)]

H --> I[Record (audit log) and report]

I --> J[Log all steps and results to audit log]

J --> K[End Process]

High Level Diagram - Forensics Agent

High level diagram showing the infrastructure components of the Forensics Agent environment.

Mermaid syntax:

graph TD

```
    User[User] -->|works on| Endpoint[Windows / Linux]
    Endpoint -->|connects to / uses| ComputeAgents[Compute Agents]
    subgraph Azure Cloud
        direction TB
        ComputeAgents[Compute Agents] --> InputStorage[Input Storage]
        ComputeAgents[Compute Agents] --> OutputStorage[Output Storage]
        ComputeAgents[Compute Agents] --> AuditLogs[Audit Logs]
        ComputeAgents[Compute Agents] --> PowerBI[PowerBI]

        InputStorage -->|access| ComputeAgents
        OutputStorage -->|store reports| ComputeAgents
        AuditLogs -->|store audit logs| ComputeAgents
        PowerBI -->|create dashboard| ComputeAgents
    end
```

Sequence Diagram — Agent Layer Interactions

Sequence diagram illustrating interactions between the agent's layers (Reactive, Planning, Cooperative), external services (malware scanner, PII detector, storage), and the user, highlighting both suspicious and clean file scenarios.

Mermaid syntax:

```
sequenceDiagram
    participant User
    participant Planning
    participant Reactive
```

```
participant MalwareScanner
participant PIIDetector
participant SecureStorage
participant ReportGenerator

User->>+Planning: Trigger process (e.g., new file)
Planning->>+Reactive: Process file
Reactive->>MalwareScanner: Scan file
activate MalwareScanner
alt Security Threat detected
  MalwareScanner-->>-Reactive: Threat detected
  Reactive->>+Planning: Alert for quarantine
  Planning->>-User: Report security threat
else No Threat detected
  MalwareScanner-->>-Reactive: No threat found
  Reactive->>-Planning: File is clean
  Planning->>PIIDetector: Scan for PII
  activate PIIDetector
  PIIDetector-->>-Planning: PII results
  Planning->>+SecureStorage: Store file
  Planning->>ReportGenerator: Generate report
  activate ReportGenerator
  ReportGenerator-->>-Planning: Report generated
  Planning->>-User: Present report
end
```