TrustX Agent Risk Classification Tool (ARC)

A Framework for Risk-Tiering Agentic AI Systems

Responsible AI Institute

Framework Version: ARC v1.1 | Tool Version: 1.0 (January-March 2026)

Based on Responsible AI Institute Standards

Abstract. The proliferation of agentic AI systems across enterprise and public-sector contexts has outpaced the capacity of general-purpose AI risk frameworks to classify and govern them. This paper introduces the **TrustX Risk Classification Tool**, a structured, repeatable instrument for risk-tiering agentic AI systems across seven agent types: Autonomous Agents, Coding Assistants, Decision Support Systems, AI Embedded/Physical Agents, Knowledge Assistants, Tool-Using Agents, and Transaction/Commerce Agents. The tool employs a **GPA+IAT classification model** (Goal, Perception, Action, Iteration, Adaptation, Termination) to determine agency level, a **twelve-dimension scoring rubric** to quantify risk, and a **five-level autonomy framework** (L1–L5) to characterise human–agent interaction. These inputs produce a **three-tier governance output** (Tier 1 Standard, Tier 2 Enhanced, Tier 3 Rigorous) with mapped control recommendations. A **specialised Coding Assistant extension** replaces the standard classification structure with a Capabilities Assessment, Deployment Model Classification (Models 1–4), a coding-specific threat taxonomy, and a 22-control security catalog, reflecting the categorically distinct risks of executable output, direct system access, and supply-chain position. The framework is grounded in NIST AI RMF, the EU AI Act, ISO/IEC 42001, OWASP, MITRE ATLAS, and SR 11-7. It is intended for AI governance practitioners, risk officers, developers, and regulators.

Keywords: Agentic AI, risk classification, AI governance, TrustX ARC, coding assistants, autonomy levels, GPA+IAT, NIST AI RMF, EU AI Act

1 Introduction

1.1 Motivation

The deployment of agentic AI systems, or applications capable of perceiving, planning, and acting with varying degrees of autonomy, has accelerated across enterprise operations, financial services, healthcare, software development, and public-sector administration. This proliferation creates governance challenges that existing, general-purpose AI risk frameworks are not designed to address. Regulatory pressure is intensifying: the EU AI Act (Regulation 2024/1689) [4] imposes risk-tier obligations, the NIST AI Risk Management Framework [3] provides voluntary but influential guidance, and sector-specific instruments such as SR 11-7 [8] mandate model risk management in banking.

Concurrently, a series of real-world incidents has demonstrated the concrete risks of under-governed agentic systems. In November 2025, Anthropic disclosed that a Chinese state-sponsored group (GTG-1002) had manipulated Claude Code to conduct espionage operations against roughly 30 global targets, with the AI executing 80–90% of the campaign autonomously [12]. In December 2025, the "IDEsaster" disclosure by security researcher Ari Marzouk revealed over 30 vulnerabilities across more than ten AI coding tools, including remote code execution via JSON schema attacks and IDE settings manipulation [13, 14]. In March 2025, Pillar Security disclosed the Rules File Backdoor attack, which exploited invisible Unicode characters in configuration files to manipulate Copilot and Cursor outputs [15]. These incidents underscore the need for a practical, defensible, and repeatable classification instrument.

1.2 Scope and Boundaries

This work addresses seven agentic AI system types (Section 3) and provides a complete intake and risk-tiering methodology. We adopt the operational definition of "agentic AI" from Bommarito, Bommarito, and Katz [1]: a system that exhibits goal-directed behaviour, environmental perception, and the capacity to act. In addition to these foundations, full agentic systems demonstrate iteration, adaptation, and termination properties. The tool is not a substitute for a comprehensive AI risk management plan. Instead, it serves as a structured intake instrument whose output informs downstream governance, audit, and compliance processes.

1.3 Key Contributions

1. **GPA+IAT agency classification model** that distinguishes Minimal Agency from Full Agentic Systems through six evaluation properties.

2. **Twelve-dimension risk scoring rubric** grounded in established frameworks, with a "critical dimension" approach that prevents high risks from being averaged away.
3. **Five-level autonomy classification** (L1–L5) adapted from Feng et al. [2].
4. **Three-tier governance output** (Tier 1, 2, 3) with mapped control recommendations aligned to NIST AI RMF, EU AI Act, and ISO/IEC 42001 [5].
5. A **specialised Coding Assistant extension** featuring a Capabilities Assessment, Deployment Model Classification (Models 1–4), a coding-specific risk taxonomy, and a 22-control security catalog.

1.4 Intended Audience

This tool is designed for AI governance and risk officers, compliance and legal teams, system developers and deployers, and regulators and policy researchers.

1.5 Paper Organization

Section 2 reviews related work. Section 3 presents the agent taxonomy. Sections 4 and 5 detail the standard and coding assistant methodologies respectively. Section 6 provides agent-type risk profiles. Section 7 offers cross-agent comparative analysis. Sections **??**–9 address validation, discussion, and conclusions.

2 Background and Related Work

2.1 Conceptual Foundations of Agentic AI

The concept of agency in artificial intelligence has evolved from early symbolic AI planners to today's foundation-model-powered agents capable of multi-step reasoning, tool use, and autonomous execution. Bommarito, Bommarito, and Katz [1] formalise this progression through the **GPA+IAT properties model**, which defines agency via six operational properties: *Goal* (the agent pursues an objective), *Perception* (it receives environmental information), *Action* (it can affect its environment), *Iteration* (it operates in repeated perceive-act cycles), *Adaptation* (it modifies behaviour based on feedback), and *Termination* (it has defined stopping conditions). Systems exhibiting all three GPA properties constitute *Minimal Agency*; those additionally demonstrating IAT properties constitute *Full Agentic Systems*. Systems lacking core GPA properties are classified as *Non-Agentic*.

2.2 Autonomy Levels in Agentic Systems

Feng et al. [2] propose a five-level Levels-of-Autonomy framework that characterises the human–agent relationship:

- **L1 (Operator):** The user directs all actions; the agent acts on command.
- **L2 (Collaborator):** User and agent collaboratively plan and execute.
- **L3 (Consultant):** The agent leads but consults the user for expertise and preferences.
- **L4 (Approver):** The agent engages the user only in pre-specified risk scenarios.
- **L5 (Observer):** Full autonomy under monitoring only.

Higher autonomy levels demand correspondingly more rigorous governance controls.

2.3 Established AI Risk and Governance Frameworks

The ARC tool builds upon several foundational frameworks:

- **NIST AI RMF 1.0** [3]: Provides the "Map, Measure, Manage, Govern" functions that structure organisational AI risk management.
- **EU AI Act** [4]: Establishes a four-tier risk classification (unacceptable, high, limited, minimal) with binding obligations for high-risk systems.
- **ISO/IEC 42001:2023** [5]: Specifies requirements for AI management systems.
- **OWASP Agentic AI Threats and Mitigations** [6]: Catalogues threat patterns specific to agentic architectures, complemented by the LLM Top 10.
- **MITRE ATLAS** [7]: Documents adversarial techniques targeting AI systems.
- **SR 11-7** [8]: Federal Reserve guidance on model risk management applicable to AI-driven financial models.

2.4 Agent-Specific and Coding-Specific Risk Research

The AURA Framework [9] addresses agent autonomy risk assessment. The Cloud Security Alliance's Capabilities-Based Risk Assessment (CBRA) offers a complementary capabilities-driven approach. Fu et al. [16] analyzed 733 AI-generated code snippets from GitHub projects and found insecure code generation rates of 29.5% in Python and 24.2% in JavaScript, spanning 43 CWE categories. GitGuardian [10] found that repositories using Copilot leak 40% more secrets than non-Copilot repositories.

2.5 What Makes AI Coding Assistants Categorically Different

AI coding assistants differ fundamentally from other agentic systems in several respects:

1. **Executable output.** They produce executable code that enters production systems directly.
2. **Direct system access.** They interact with file systems, terminals, networks, and databases.
3. **Persistent learning.** They learn from codebase patterns across sessions.
4. **Autonomous workflows.** They can plan, execute, test, commit, and deploy without human intervention.
5. **Supply-chain multiplier.** A single compromised assistant can inject vulnerabilities across dozens of projects.
6. **Attack timeline compression.** Reconnaissance-to-exfiltration is reduced from days to minutes.
7. **Trust boundary erosion.** Developers inherently over-trust AI-generated code.

2.6 Gaps Addressed by This Work

Existing frameworks lack agent-type-specific scoring rubrics, do not employ a "critical dimension" approach to prevent high risks from being averaged away, offer no integrated tool combining agency classification, risk scoring, autonomy level, and governance output, and provide no dedicated risk instrument for AI coding assistants grounded in real incident data.

3 Agentic AI Taxonomy

3.1 Classification Criteria

Agent types are differentiated along four axes: (i) primary function and task domain, (ii) degree of autonomy and human-in/on-the-loop characteristics, (iii) nature of operating environment (digital, physical, hybrid), and (iv) action authority and system reach.

3.2 The Seven Agent Types

1. **Autonomous Agents** — Self-directed, goal-pursuing systems with minimal human intervention.
2. **Coding Assistants** — Developer-facing code generation, analysis, and execution tools. Uses a fully differentiated assessment structure (Section 5).
3. **Decision Support Systems** — AI that informs but does not replace human decisions.
4. **AI Embedded/Physical Agents** — Agents operating in or as physical systems (vehicles, robots, industrial equipment).
5. **Knowledge Assistants** — Information retrieval, synthesis, and Q&A agents.
6. **Tool-Using Agents** — Agents that invoke external APIs, tools, or services.
7. **Transaction/Commerce Agents** — Agents that execute financial or commercial actions.

3.3 Rationale for the Coding Assistant Structural Exception

The GPA+IAT model does not capture the supply-chain, executable-output, and direct-system-access properties unique to coding tools. The Deployment Model classification (Models 1–4) more accurately characterises the risk profile than L1–L5 autonomy levels, and the real-world incident record justifies a dedicated threat taxonomy.

4 The TrustX Standard Risk Classification Methodology

This section describes the six-section assessment structure applied to all agent types *except* Coding Assistants.

4.1 Tool Structure Overview

The standard tool comprises six sections: (1) Agent Identification, (2) GPA+IAT Agent Properties Assessment, (3) Risk Tier Classification across twelve dimensions, (4) Autonomy Level Assessment (L1–L5), (5) Additional Risk Factors, and (6) Recommended Governance Controls.

4.2 Section 1 — Agent Identification

Metadata fields capture the Agent Name, Agent ID, Assessment Date, Agent Owner/Team, Business Unit, Deployment Status, and Agent Purpose/Description.

4.3 Section 2 — GPA+IAT Agency Classification

Six agentic properties are evaluated with a binary (Yes/No) assessment and supporting evidence. The classification output is automated:

- All six properties present ⇒ **Full Agentic System**
- GPA only ⇒ **Minimal Agency System**
- Core GPA absent ⇒ **Non-Agentic System**

Table 1 summarises the six properties with illustrative evidence.

Table 1: GPA+IAT Properties Assessment

Tier	Property	Description
GPA	Goal	Agent pursues a defined objective
	Perception	Receives information from environment
	Action	Can affect its environment
IAT	Iteration	Operates in repeated perceive-act cycles
	Adaptation	Modifies behaviour based on feedback
	Termination	Has defined stopping conditions

4.4 Section 3 — Twelve-Dimension Risk Scoring

Each dimension is scored on a three-point scale: 1 (Low), 2 (Medium), 3 (High). Table 2 presents the twelve dimensions with tier descriptors.

Three computed outputs are derived: Total Score (sum), Average Score, and Highest Individual Dimension score.

4.5 Section 4 — Autonomy Level Assessment (L1–L5)

The assessor identifies the single most appropriate autonomy level. Risk mapping proceeds as: L1–L2 ⇒ Low; L3–L4 ⇒ Medium; L5 ⇒ High.

4.6 Final Risk Tier Determination Logic

The "critical dimension" approach ensures that a single high-risk score cannot be averaged away:

- **Tier 3 (High Risk):** Any dimension scored 3, *or* Autonomy Level = L5.
- **Tier 2 (Medium Risk):** Maximum dimension = 2, *or* average ≥ 1.5.
- **Tier 1 (Low Risk):** All dimensions ≤ 1 and average < 1.5.

Special modifiers apply: financial services customer-facing agents require a Tier 2 minimum; multi-agent systems inherit the highest tier of any constituent agent; Tier 3 may require board-level approval and regulatory reporting; and re-assessment is triggered by any change in capabilities, tooling, or deployment context.

4.7 Section 5 — Additional Risk Factors

A contextual checklist evaluates: PII handling, financial transaction processing, credit/underwriting decisions, regulated data access, multi-agent coordination requirements, real-time decision-making, and external system integration. Each factor is assessed as Present/Not Present with supporting evidence.

4.8 Section 6 — Recommended Governance Controls

Controls are mapped to tiers:

- **Tier 1 — Standard:** Documentation, basic human-in-the-loop, audit logs.
- **Tier 2 — Enhanced:** Behavioural boundaries, kill switch, enhanced monitoring.
- **Tier 3 — Rigorous:** Third-party validation, continuous monitoring, regulatory reporting.

Controls are drawn from the TrustX ARC control catalog (RAI-GOV, RAI-SEC, RAI-SAFE series) and aligned to NIST AI RMF, EU AI Act, and ISO/IEC 42001.

5 The TrustX Coding Assistant Methodology

The Coding Assistant tool extends TrustX ARC v1.1 with a fully differentiated six-section structure that replaces the GPA+IAT and L1–L5 autonomy sections with coding-specific counterparts.

5.1 Section 1 — Coding Assistant Identification

Fields include: Tool Name, Tool Type (Claude Code, Copilot, Cursor, etc.), Deployment Mode (IDE, CLI, Cloud, On-prem), Model Version, Development Team/Owner, Assessment Date, Primary Use Case, and Repositories/Projects with Access.

5.2 Section 2 — Capabilities Assessment

Replacing GPA+IAT, this section evaluates 20 discrete capabilities across three groups:

Code Generation Capabilities: Line/function completion, multi-file/full feature generation, code refactoring, code review and security analysis, automated bug detection, test generation, and documentation generation.

System Interaction Capabilities: Terminal/shell command execution, file system read and write access, API/external tool integration (MCP, plugins), web browsing/information retrieval, database access, cloud resource management (AWS, Azure, GCP), and git operations (commit, push, PR creation).

Autonomy and Memory Capabilities: Multi-step autonomous workflows, local codebase learning, persistent memory across sessions, YOLO mode/skip permissions mode, and Docker/container integration.

The capability profile directly informs downstream risk scoring and deployment model classification.

Table 2: Twelve-Dimension Risk Scoring Rubric — Standard Tool

Dimension	Tier 1 (Low)	Tier 2 (Medium)	Tier 3 (High)	Scoring Criteria
Autonomy	Human-in-the-loop required for every action	Human-on-the-loop; agent acts but human can intervene	Fully autonomous operation with no real-time oversight	Level of human supervision
Decision Scope	Localised decisions for the specific task	Domain or function-level decisions	Enterprise-wide decisions	Influence on organisation
Temporal Coupling	Isolated, independent actions	Chained workflows with potential domino effects	Continuous autonomous feedback loops	Risk compounding over time
Action Authority	Read-only or advisory	Create/modify content	Full transaction execution authority	Scope of performable actions
System Reach	Single internal system	Multiple internal systems	Cross-domain or third-party systems	Degree of system access
Blast Radius	Single user or small scope	Team or department level	Enterprise-wide or public-facing	Scale of potential harm
Persistence	Stateless; no memory	Session-based memory	Long-term persistent memory	State retention capabilities
Reversibility	Fully reversible	Partially reversible	Irreversible or cascading	Ability to undo actions
Control Authority	Standalone agent	Supervises other agents	Orchestrates agent fleets	Control over other agents
Data Sensitivity	Public or non-sensitive	Internal or confidential	Regulated or crown-jewel data	Classification of accessed data
Aggregation Risk	No aggregation	Limited aggregation	Cross-session or inferential	Harm through accumulation
Data Egress Paths	Constrained outputs	Multiple controlled outputs	Multi-tool or external paths	Methods data can exit

5.3 Section 3 — Deployment Model Classification

Replacing L1–L5 autonomy levels, four deployment models are defined (Table 3):

Table 3: Coding Assistant Deployment Models

Model	Description	Typical Tier
1: IDE Autocomplete	Suggestions only; no autonomous execution or file modifications	Tier 1
2: File-Level Agent	Can modify files, run read commands; requires approval	Tier 2
3: Autonomous Multi-Step	Executes 30+ minute workflows; may skip approvals	Tier 2–3
4: Production-Connected	Access to production, cloud, or regulated systems	Tier 3

5.4 Section 4 — Twelve Coding-Specific Risk Dimensions

The same twelve dimension names are used, but tier descriptors are re-specified for the developer and coding context. Key adaptations include:

- **Autonomy:** Ranges from "every action requires approval" (Tier 1) to "fully autonomous 30+ minute workflows" (Tier 3).
- **Action Authority:** Ranges from "code suggestions only" to "shell write + API calls + deployments."
- **Reversibility:** Ranges from "all changes easily reversible via git" to "production deployments or data changes."
- **Persistence:** Ranges from "session-only memory" to "cross-project persistent learning."

Tier determination logic follows the same critical-dimension approach as the standard tool.

5.5 Section 5 — Coding-Specific Risk Factors

Three risk groups are assessed, each with Presence, Severity, and Mitigation Status:

5.5.1 Group A: Supply Chain and Artifact Integrity

- SAFE-CA-002 Package Hallucination/Squatting (HIGH)
- SAFE-CA-005 Training Data Bias/Poisoning (MEDIUM)
- OPS-CA-001 License Compliance Issues (MEDIUM)
- SAFE-CA-001 Insecure Code Generation—CWEs (HIGH)

5.5.2 Group B: Agent Manipulation and Workflow Hijacking

- SEC-CA-003 IDE Configuration Manipulation (HIGH)
- SEC-CA-008 Invisible Unicode/JSON Schema Remote Trigger (MEDIUM)
- SEC-CA-002 Rules File Backdoor—.cursorrules (HIGH)
- SEC-CA-009 Persona Manipulation Attack (HIGH)
- SEC-CA-010 Multi-Agent Coordination Attack (MEDIUM)
- SEC-CA-007 Multi-Root Workspace Exploitation (MEDIUM)
- SEC-CA-006 Data Exfiltration via Generated Code (CRITICAL)
- SAFE-CA-003 Autonomous Workflow Without Bounds (CRITICAL)
- SEC-CA-001 Prompt Injection Vulnerability (HIGH)

5.5.3 Group C: Access and Privilege Abuse

- OPS-CA-005 Production System Access Without Controls (CRITICAL)
- SAFE-CA-003 YOLO Mode/Skip Permissions (CRITICAL)
- SAFE-CA-004 Code Review Bypass (HIGH)
- SEC-CA-005 Secrets/Credentials Leakage Risk (CRITICAL)
- SEC-CA-004 Command Injection via Terminal (CRITICAL)

5.6 Section 6 — Required Security Controls

Twenty-two controls comprise the RAI-CA catalog (Table 4).

Table 4: RAI-CA Security Control Catalog (Summary)

ID	Category	Control
RAI-CA-001	Code Review	Mandatory review of AI code
RAI-CA-002	Secrets Mgmt	Pre-commit secrets scanning
RAI-CA-003	Licence	Licence compliance checking
RAI-CA-004	Attribution	Tag AI-generated code
RAI-CA-005	Privacy	No training on private code
RAI-CA-006	Sandboxing	Filesystem isolation
RAI-CA-007	Sandboxing	Network isolation
RAI-CA-008	Code Security	SAST on generated code
RAI-CA-009	Secrets Mgmt	Real-time secret redaction
RAI-CA-010	Config Security	Rule file validation
RAI-CA-011	Config Security	IDE settings protection
RAI-CA-012	Audit	Prompt/response logging
RAI-CA-013	Supply Chain	Package verification
RAI-CA-014	Code Security	DAST implementation
RAI-CA-015	Oversight	Senior dev approval (critical)
RAI-CA-016	Monitoring	Behavioural monitoring
RAI-CA-017	Incident Resp.	Red team testing
RAI-CA-018	Incident Resp.	Kill switch / emergency stop
RAI-CA-019	Container	Docker isolation
RAI-CA-020	Access Control	Production access restriction
RAI-CA-021	Governance	Third-party security audit
RAI-CA-022	Compliance	Regulatory documentation

6 Agent-Type Risk Profiles: An Illustrative Example

The following profiles apply the ARC tool to representative configurations of each agent type. Scores are illustrative, reflecting typical deployments rather than specific assessed systems, and are intended to demonstrate how the methodology produces differentiated risk tiers across agent types.

6.1 Autonomous Agents

Autonomous agents are self-directed systems that pursue goals with minimal human intervention. GPA+IAT assessment consistently yields a **Full Agentic System** classification, with all six properties present. Representative examples include operations optimisation agents that monitor deployed systems, customer service agents that independently resolve tickets, and strategic planning agents that execute multi-step business processes.

Risk scoring from the TrustX ARC tool reveals a Total Score of 18 with an Average of 1.5. The highest-risk dimensions are **Persistence** (3—long-term memory) and **Data Sensitivity** (3—

regulated data access), with **Decision Scope** and **Aggregation Risk** at medium (2). The identified autonomy level is **L5 (Observer)**, confirming full autonomous operation. Additional risk factors include PII handling, financial transaction processing, multi-agent coordination, and real-time decision-making. The final risk tier is **Tier 3 (High Risk)**, requiring rigorous controls including third-party validation, continuous monitoring, and regulatory reporting.

6.2 Coding Assistants

Coding assistants follow the differentiated methodology (Section 5). Representative tools include Claude Code, GitHub Copilot, and Cursor. Using the Capabilities Assessment, a baseline Model 1 deployment (IDE autocomplete only) scores a Total of 16.8 with Average 1.4, yielding **Tier 2 (Medium Risk)**. The highest-risk dimension is **Data Sensitivity** (3) when accessing confidential codebases. Expected risk tiers by deployment model: Model 1 = Tier 1; Model 2 = Tier 2; Model 3 = Tier 2–3; Model 4 = Tier 3.

Key risk factors include supply chain attacks (package hallucination, insecure code generation), agent manipulation (Rules File Backdoor, IDE configuration attacks), and access abuse (YOLO mode, credentials leakage).

6.3 Decision Support Systems

Decision support systems generate insights and recommendations for human decision-makers. GPA+IAT classification yields **Full Agentic System** when adaptation and iteration are present. Risk scoring shows a Total of 14, Average 1.17. The critical dimension is **Autonomy** (3), as some systems operate with full autonomy in generating recommendations. Identified autonomy level is L5 (Observer). Despite low scores across most dimensions—Action Authority (1, advisory only), System Reach (1), Blast Radius (1)—the single Autonomy score of 3 triggers **Tier 3 (High Risk)**, demonstrating the critical dimension principle.

6.4 AI Embedded/Physical Agents

Physical agents—autonomous vehicles, surgical robots, industrial automation—present unique risks due to **irreversibility** of physical actions and high **blast radius** in safety-critical environments. All six GPA+IAT properties are typically present (Full Agentic System). Risk scoring yields Total 18, Average 1.5, with Persistence (3) and Data Sensitivity (3) as highest dimensions. Autonomy level is L5 (Observer). Final tier: **Tier 3 (High Risk)**.

6.5 Knowledge Assistants

Knowledge assistants retrieve and synthesise information. GPA+IAT classification ranges from Minimal Agency to Full Agentic System depending on adaptation capabilities. Risk scoring yields Total 18, Average 1.5, with critical dimensions in **Persistence** (3—long-term memory) and **Data Sensitivity** (3—access to regulated data). Action Authority is typically low (1, advisory only). Autonomy level: L5. Final tier: **Tier 3 (High Risk)**, primarily driven by data sensitivity rather than action authority.

6.6 Tool-Using Agents

Tool-using agents invoke external APIs and services. Full Agentic System classification is typical. Risk scoring yields Total 18, Average 1.5, with highest dimensions in **Persistence** (3) and **Data Sensitivity** (3). Key risk drivers are System Reach and Data Egress Paths, as these agents bridge multiple systems. Autonomy level: L5. Final tier: **Tier 3 (High Risk)**.

6.7 Transaction/Commerce Agents

Transaction agents execute purchasing, payments, claims approval, and inventory management. GPA+IAT: Full Agentic System. Risk scoring yields Total 14, Average 1.17, with **Autonomy** (2) and **Persistence** (2) as highest dimensions. Autonomy level: **L3 (Consultant)**. Final tier: **Tier 2 (Medium Risk)**. Regulatory considerations include AML, KYC, SR 11-7, and consumer protection. Financial services deployments require Tier 2 minimum for customer-facing agents; enterprise-scope deployments may escalate to Tier 3.

7 Cross-Agent Comparative Analysis of Illustrative Scenarios

Table 5 presents the aggregate risk landscape across all seven agent types. The scores are adopted from the hypothetical deployments described in Section 6.

7.1 Key Findings

Shared high-risk dimensions. Data Sensitivity and Persistence are the most frequent drivers of Tier 3 classification across agent types. This reflects the fundamental risk that agentic systems

Table 5: Cross-Agent Risk Score Comparison (illustrative scores for representative deployments)

Agent Type	*Auton.*	*Dec. Scope*	*Temporal*	*Action*	*Sys. Reach*	*Blast Rad.*	*Persist.*	*Revers.*	*Control*	*Data Sens.*	*Aggreg.*	*Egress*	Total	Avg	Tier
Autonomous	1	2	1	1	1	1	3	1	1	3	2	1	18	1.50	**3**
Coding Asst. (M1)	1	2	1	2	1	1	2	1	1	3	2	1	18	1.50	**2**
Decision Support	3	1	1	1	1	1	1	1	1	1	1	1	14	1.17	**3**
Embedded/Physical	1	2	1	1	1	1	3	1	1	3	2	1	18	1.50	**3**
Knowledge Asst.	1	2	1	1	1	1	3	1	1	3	2	1	18	1.50	**3**
Tool-Using	1	2	1	1	1	1	3	1	1	3	2	1	18	1.50	**3**
Transaction	2	1	1	1	1	1	2	1	1	1	1	1	14	1.17	**2**

with access to regulated data and long-term memory present compounding governance challenges.

The critical dimension effect. Decision Support Systems illustrate a key design principle: despite 11 of 12 dimensions scoring 1, a single Autonomy score of 3 triggers Tier 3. This validates the "critical dimension" approach as essential for preventing risk dilution.

Transaction/Commerce agents as the moderate-risk archetype. With no dimension exceeding 2, Transaction agents represent the only type consistently classified at Tier 2, though financial services modifiers may escalate them.

Multi-agent cascade risks. The tier inheritance rule—where a system inherits the highest tier of any constituent agent—is particularly significant for heterogeneous deployments. A coding assistant used as a sub-agent within an autonomous workflow inherits the workflow's tier classification.

8 Discussion

8.1 Implications for AI Governance and Policy

The TrustX ARC tier structure maps directly to regulatory risk categories. The critical dimension approach offers a design principle transferable to policy: no amount of low-risk properties should offset a single genuinely high-risk property. For coding assistants specifically, regulatory considerations span GDPR (data in training), SOC 2 (access controls), intellectual property law (licence compliance), and emerging software liability frameworks.

8.2 Implications for Developers and Deployers

Organisations should embed TrustX ARC into procurement gates, development lifecycle reviews, and deployment approvals. Key triggers for reassessment include capability upgrades, new tool integrations, deployment model transitions, and changes in data access scope. YOLO mode represents an organisational policy question, not merely a technical configuration—its activation should require explicit governance approval.

8.3 Limitations

Scoring subjectivity is mitigated but not eliminated by detailed rubrics and scorer training. The tool provides a static snapshot and does not capture drift over the agent lifecycle. The coding assistant threat landscape is rapidly evolving and the tool will require regular updates. The seven agent types, while comprehensive, may not cover all emerging agentic configurations.

8.4 Future Work

Planned extensions include: dynamic and continuous risk monitoring, automated scoring via capability inspection (cf. Cihon et al.), dedicated extensions for other high-specificity agent types, integration with enterprise AI governance platforms, and expansion of the taxonomy to emerging agent configurations such as multi-modal agentic systems and agent-to-agent networks.

9 Conclusion

The TrustX ARC Agent Risk Tool provides a structured, defensible, and repeatable methodology for risk-tiering agentic AI systems. Its dual-track architecture addresses the full spectrum of agentic AI currently deployed in enterprise contexts. The GPA+IAT agency classification, twelve-dimension risk scoring, five-level autonomy framework, and three-tier governance output together deliver an integrated instrument that bridges the gap between theoretical AI risk frameworks and the operational requirements of governance practitioners, developers, and regulators.

The critical dimension approach ensures that genuinely high-risk properties are never diluted by averaging. The coding assistant extension, grounded in real-world incident data, provides the

first dedicated risk instrument for a class of agents whose supply-chain position, executable output, and direct system access warrant differentiated treatment.

We invite adoption, feedback, and contribution to future versions from the AI governance community.

A GPA+IAT Property Assessment — Worked Examples by Agent Type

Table 6 presents representative GPA+IAT evidence strings across the six standard agent types.

Table 6: GPA+IAT Worked Examples by Agent Type

Type	Property	Evidence (Goal)	Evidence (Iteration)
Autonomous	All 6: Yes	Improve website performance; identify competitors	Check for tickets every 5 min; monitor systems
Decision Support	All 6: Yes	Generate data summaries; provide recommendations	Execute analyses on different datasets; live statistics
Embedded/Physical	All 6: Yes	Provide navigation; map walking routes	Continuously check for incoming objects
Knowledge Asst.	All 6: Yes	Answer questions about contents; summarise documents	Converse with user; provide updated summaries
Tool-Using	All 6: Yes	Browse web for answers; scrape websites	Retry failed API calls; scrape sequentially
Transaction	All 6: Yes	Purchase within budget; execute payments	Execute repeat payments; check inventory

B Coding Assistant Capabilities Assessment — Worked Example

Table 7 illustrates a completed 20-capability assessment for a representative coding assistant deployment.

Table 7: Capabilities Assessment — Coding Assistant (20 Capabilities)

Group	Capability	Present?	Risk Implication
Code Gen.	Line/Function Completion	Yes	Insecure code generation
	Multi-File/Full Features	–	Broader blast radius
	Code Refactoring	–	Unintended logic changes
	Code Review & Security	–	False sense of security
	Automated Bug Detection	–	Missed vulnerabilities
	Test Generation	–	Inadequate test coverage
	Documentation Generation	–	Stale documentation
Sys. Inter.	Terminal/Shell Execution	–	Command injection risk
	File System Read	–	Data exposure
	File System Write	–	Unauthorised modifications
	API/MCP Integration	–	Supply chain attack surface
	Web Browsing	–	Prompt injection via web
	Database Access	–	Data exfiltration
	Cloud Resource Mgmt	–	Infrastructure compromise
	Git Operations	–	Unauthorised commits
Autonomy	Multi-step Workflows	–	Unbounded execution
	Local Codebase Learning	–	Pattern leakage
	Persistent Memory	–	Cross-session aggregation
	YOLO Mode	–	Bypassed permissions
	Docker Integration	–	Container escape

C Coding Assistant Threat Taxonomy

The coding-specific threat taxonomy comprises three series:

SAFE-CA (Safety): SAFE-CA-001 Insecure Code Generation (HIGH), SAFE-CA-002 Package Hallucination/Squatting (HIGH), SAFE-CA-003 Autonomous Workflow Without Bounds (CRITICAL), SAFE-CA-004 Code Review Bypass (HIGH), SAFE-CA-005 Training Data Bias/Poisoning (MEDIUM).

SEC-CA (Security): SEC-CA-001 Prompt Injection (HIGH), SEC-CA-002 Rules File Backdoor (HIGH), SEC-CA-003 IDE Configuration Manipulation (HIGH), SEC-CA-004 Command Injection via Terminal (CRITICAL), SEC-CA-005 Secrets/Credentials Leakage (CRITICAL), SEC-CA-006 Data Exfiltration via Generated Code (CRITICAL), SEC-CA-007 Multi-Root Workspace Exploitation (MEDIUM), SEC-CA-008 Invisible Unicode/JSON Schema (MEDIUM), SEC-CA-009 Persona Manipulation Attack (HIGH), SEC-CA-010 Multi-Agent Coordination Attack (MEDIUM).

OPS-CA (Operational/Compliance): OPS-CA-001 License Compliance Issues (MEDIUM), OPS-CA-005 Production System Access Without Controls (CRITICAL).

D User Guide Summary

Standard Tool: (1) Complete Section 1 identification fields. (2) Evaluate each GPA+IAT property with evidence. (3) Score all twelve dimensions 1–3. (4) Select the applicable autonomy level. (5) Complete the additional risk factors checklist. (6) Review the auto-generated governance controls.

Coding Assistant Tool: (1) Complete identification fields including deployment mode and repositories. (2) Assess all 20 capabilities. (3) Select the applicable deployment model. (4) Score twelve coding-specific dimensions. (5) Assess all three risk factor groups with severity. (6) Review the RAI-CA security controls.

Scoring Tips: When uncertain between adjacent scores, consider the "worst reasonable case" interpretation. Document rationale for each score. Involve subject-matter experts for dimensions outside the assessor's domain.

References

[1] M. J. Bommarito, J. Bommarito, and D. M. Katz, "What is an Agent? A Conceptual Primer and History of Agents and Agentic AI," SSRN, 2025. Available: https://ssrn.com/abstract=5806982

[2] Feng et al., "Levels of Autonomy for AI Agents," arXiv:2506.12469, 2025. Available: https://arxiv.org/abs/2506.12469

[3] National Institute of Standards and Technology, "AI Risk Management Framework (AI RMF 1.0)," 2023. Available: https://www.nist.gov/itl/ai-risk-management-framework

[4] European Union, "EU AI Act — Regulation (EU) 2024/1689 on Artificial Intelligence," 2024. Available: https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

[5] ISO/IEC 42001:2023, "AI Management System Standard," 2023. Available: https://www.iso.org/standard/42001

[6] OWASP, "Agentic AI: Threats and Mitigations; LLM Top 10," 2024. Available: https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/

[7] MITRE, "MITRE ATLAS: Adversarial Threat Landscape for AI Systems," 2024. Available: https://atlas.mitre.org/techniques

[8] Board of Governors of the Federal Reserve System, "SR 11-7: Guidance on Model Risk Management," 2011. Available: https://www.federalreserve.gov/supervisionreg/srletters/sr1107.htm

[9] Multiple Authors, "AURA Framework: Agent Autonomy Risk Assessment," arXiv:2510.15739, 2025. Available: https://arxiv.org/abs/2510.15739

[10] GitGuardian, "Copilot repositories leak 40% more secrets than non-Copilot repositories," GitGuardian Research, 2024.

[11] Multiple Authors, "IDEsaster: 24 CVEs Across 10+ AI Coding Tools," December 2025.

[12] Anthropic. (2025). Disrupting the first reported AI-orchestrated cyber espionage campaign. https://www.anthropic.com/news/disrupting-AI-espionage

[13] A. Marzouk (MaccariTA). (2025). IDEsaster: A Novel Vulnerability Class in AI IDEs. https://maccarita.com/posts/idesaster/

[14] R. Lakshmanan. (2025). Researcher Uncovers 30+ Flaws in AI Coding Tools Enabling Data Theft and RCE Attacks. *The Hacker News*, December 2025. https://thehackernews.com/2025/12/researchers-uncover-30-flaws-in-ai.html

[15] Z. Karliner and Pillar Security. (2025). New Vulnerability in GitHub Copilot and Cursor: How Hackers Can Weaponize Code Agents Through Compromised Rule Files. https://www.pillar.security/blog/new-vulnerability-in-github-copilot-and-cursor-how-hackers-can-weaponize-code-agents

[16] Y. Fu, P. Liang, A. Tahir, Z. Li, M. Shahin, J. Yu, and J. Chen, "Security Weaknesses of Copilot-Generated Code in GitHub Projects: An Empirical Study," *ACM Trans. Softw. Eng. Methodol.*, 2025. Available: https://dl.acm.org/doi/abs/10.1145/3716848