Nuke-M Methodology

Author: Julius Castillo, Jr. | AiA Systems | aiasystems.org

The First Software Development Methodology Under CEDA's ACE-EDSO

Pronounced: "nuke 'em"

Nuke-M is an AI-native software engineering and software development methodology operating under **ACE-EDSO: Ai-Assisted Continuous Execution using Event-Driven Software Orchestration**.

Where CEDA provides the governing philosophy and ACE-EDSO provides the operational method, Nuke-M defines the concrete lifecycle by which a software product is conceived, specified, designed, implemented, tested, released, operated, maintained, and completed.

CEDA establishes that the software product is the central artifact, that its real-time state is the authoritative source of truth, and that recurring human synchronization should be replaced by continuous AiA-maintained product-state awareness. Nuke-M turns those philosophical commitments into an executable development methodology.

---

1. Definition

Nuke-M is an Ai-Assisted, continuously executing, event-driven software development methodology in which work is prioritized, assigned, observed, verified, and advanced according to the real-time state of the software product rather than through scheduled meetings, fixed ceremonies, or periodic human status reporting.

Nuke-M creates work based on:

- continuously maintained product state,
- AiA-generated execution priorities,
- asynchronous human review,
- automated quality gates,
- targeted event notifications,
- minimum-necessary human synchronization.

Its objective is not to eliminate communication. Its objective is to eliminate communication that does not need to interrupt software production.

---

2. Position of Nuke-M Within CEDA

The relationship is:

CEDA
Continuous Engineering & Development Architecture
The governing software engineering philosophy

↓

ACE-EDSO
Ai-Assisted Continuous Execution using
Event-Driven Software Orchestration
The operational execution method

↓

Nuke-M
The software development methodology
used to take a product through its complete development lifecycle

Nuke-M is designed from the beginning around a different operating assumption:

> An AiA system can continuously maintain, analyze, prioritize, and publish the real-time state of the software product, even at onset or conception.

---

3. Fundamental Operating Rule

Nuke-M has one default state:

> **Continuous Execution**

An engineer who has an executable, unblocked, properly prioritized work item continues engineering.

The engineer does not stop because:

- the clock reached 9:00 AM,
- a scheduled calendar invitation exists,
- the rest of the team is providing status,
- a manager wants verbal confirmation of information already represented in the product state.

Execution stops only when:

1. the current task is completed;
2. a dependency prevents further execution;
3. a quality or security gate fails;
4. an event requires targeted human resolution;
5. authorized organizational maintenance is scheduled;
6. a higher-priority event legitimately preempts the work.

This implements CEDA's position that execution is the default and interruption is the exception.

---

4. The Nuke-M Product State

The central operating artifact of Nuke-M is the **Real-Time Product State**.

It is the continuously updated representation of everything presently known about the software product.

The Product State includes, but is not limited to:

- product objectives;
- approved requirements;
- unapproved requirements;
- completed capabilities;
- incomplete capabilities;
- work items;
- work-item owners;
- current priorities;
- dependencies;
- architectural decisions;
- source-code changes;
- branches and commits;
- pull or merge requests;
- build results;

- automated test results;
- manual test results;
- code-review status;
- security findings;
- deployment status;
- environmental state;
- defects;
- risks;
- blockers;
- operational telemetry;
- release readiness;
- evidence supporting completion.

The Product State is not merely a project-management dashboard.

A dashboard displays entered information.

The Nuke-M Product State is continuously derived, correlated, and validated from engineering evidence.

Its sources may include:

- Jira, Microsoft Project, Trello, ClickUp, Asana, Monday.com, Smartsheet, or another work-management system;
- source-control systems;
- CI/CD platforms;
- automated test systems;
- deployment systems;
- observability platforms;
- production telemetry;
- architecture records;
- product specifications;
- security tools;
- code-review systems;
- human decisions captured as durable records.

The project-management tool is therefore replaceable.

The Product State is not.

---

5. The Role of AiA

Within Nuke-M, **AiA means Ai-Assisted**.

AiA is the methodology's continuous execution-orchestration layer.

AiA performs six primary functions:

5.1 Observe

AiA monitors authorized product artifacts and engineering systems.

5.2 Analyze

AiA determines what changed, what remains incomplete, what is blocked, and what evidence exists.

5.3 Correlate

AiA connects requirements, tasks, code changes, tests, defects, deployments, dependencies, and responsible engineers.

5.4 Prioritize

AiA recommends the next executable work based on product value, dependencies, risk, urgency, effort, resource availability, and organizational objectives.

5.5 Orchestrate

AiA routes work, identifies affected participants, raises events, applies gates, and advances verified work through its lifecycle.

5.6 Publish

AiA makes the current Product State available to authorized participants in real time.

AiA may recommend and orchestrate. It does not automatically acquire unlimited authority. Human decision rights remain governed by the organization's authorization model.

---

6. Nuke-M Participants

6.1 Product Authority

The Product Authority determines:

- the product mission;
- business priorities;
- acceptable outcomes;
- scope boundaries;
- release objectives;
- value and risk tradeoffs;
- whether the product has satisfied its completion conditions.

This responsibility may belong to one person or an authorized group.

6.2 Engineering Authority

The Engineering Authority governs:

- technical direction;
- architecture;
- engineering standards;
- quality thresholds;
- security requirements;
- deployment readiness;
- technical exception approval.

6.3 Engineers

Engineers design, implement, test, review, debug, validate, deploy, and maintain the product.

An engineer may perform several disciplines, including:

- software development;
- quality engineering;
- DevOps;
- platform engineering;

- security engineering;
- database engineering;
- architecture;
- user-experience engineering.

6.4 Product and Domain Specialists

These participants contribute requirements, domain rules, acceptance evidence, regulatory knowledge, customer needs, and operational constraints.

6.5 AiA System Engineer

Ascertains that AiA maintains the Product State and coordinates execution under configured authority, rules, evidence requirements, and human governance.

6.6 Event Authority (Project Engineer)

An Event Authority is the person authorized to resolve a particular class of event when AiA cannot resolve it from existing evidence or policy.

Different events may have different authorities:

- product-scope event → Product Authority;
- architectural event → Engineering Authority;
- security event → Security Authority;
- production incident → Incident Authority;
- regulatory event → Compliance Authority.

There is no permanent meeting facilitator role because routine synchronization meetings are not part of the methodology.

---

7. Nuke-M Core Artifacts

7.1 Product Mission

A durable statement explaining:

- who the product serves;
- what problem it solves;

- what outcome it must produce;
- what boundaries constrain it.

7.2 Product Requirements

The approved functional, nonfunctional, operational, regulatory, security, accessibility, and quality expectations of the product.

7.3 Product Map

A structured representation of the product's:

- capabilities;
- systems;
- components;
- interfaces;
- data;
- dependencies;
- environments;
- release boundaries.

7.4 Work Inventory

The complete set of known unfinished work.

It replaces the idea of a backlog as a periodically groomed list. The inventory is continuously evaluated as product knowledge changes.

7.5 Execution Queue

The ordered set of work items that are presently:

- approved;
- sufficiently defined;
- dependency-ready;
- assigned or assignable;
- executable.

The Work Inventory may contain unresolved or future work. The Execution Queue contains work that can actually proceed.

7.6 Event Record

A structured record created when product execution requires intervention.

It contains:

- triggering condition;
- affected product area;
- affected work items;
- evidence;
- severity;
- required participants;
- decision authority;
- response deadline;
- resolution;
- resulting actions.

7.7 Evidence Record

The evidence supporting a claim about product state, such as:

- test output;
- commit;
- build result;
- screenshot;
- review approval;
- deployment response;
- performance measurement;
- security scan;
- user acceptance result.

7.8 Decision Record

A durable explanation of an important decision, including:

- decision;
- context;
- evidence;
- alternatives;
- authority;
- consequences;
- date;
- conditions under which the decision should be revisited.

7.9 Product-State Publication

The human-readable view of the current Product State published by AiA.

7.10 Completion Record

The accumulated evidence proving that a work item, release, capability, or product has met its defined completion conditions.

---

8. Work-Item Structure

Every executable Nuke-M work item must contain enough information to support independent execution and objective verification.

A work item should define:

- unique identifier;
- product and product area;
- business or technical purpose;
- requirement or problem;
- expected outcome;
- acceptance conditions;
- dependencies;
- priority;
- assigned engineer or eligible ownership group;
- affected components;
- known risks;
- required evidence;
- quality gates;
- current state;
- event history;
- completion evidence.

A task that lacks sufficient information to execute is not considered executable.

It remains in a clarification or preparation state until the deficiency is resolved.

---

9. Standard Work-Item States

Nuke-M uses an evidence-driven state model.

9.1 Discovered

A need, defect, risk, opportunity, or requirement has been identified.

9.2 Under Analysis

The item is being investigated to determine meaning, value, feasibility, scope, and dependencies.

9.3 Awaiting Decision

A human authority must resolve a product, business, technical, legal, security, or architectural question.

9.4 Approved

The item is authorized for eventual execution.

9.5 Ready for Execution

The item is sufficiently defined, dependencies are satisfied, and execution may begin.

9.6 In Execution

An engineer is actively producing or modifying the product.

9.7 Blocked

Execution cannot continue because a dependency or unresolved event prevents progress.

9.8 In Verification

Implementation is complete enough to undergo required reviews, tests, and validation.

9.9 Rework Required

Verification identified a deficiency that must be corrected.

9.10 Verified

All work-item-level acceptance conditions and quality gates have passed.

9.11 Integrated

The verified change has been incorporated into the appropriate product baseline.

9.12 Released

The change has been delivered to its intended operational environment or users.

9.13 Observed

Post-release operation has been examined for required stability, correctness, performance, and user impact.

9.14 Done

All defined completion conditions and evidence requirements are satisfied.

9.15 Reopened

New evidence demonstrates that the Done state is no longer accurate.

A state transition must be supported by evidence or authorized decision. Merely stating that work is complete does not make it complete.

---

10. Prioritization Under Nuke-M

AiA continuously evaluates unfinished work rather than waiting for a planning ceremony.

Prioritization considers:

- product value;
- user impact;
- safety;
- security;

- legal or regulatory obligation;
- production severity;
- dependency unlocking;
- architectural risk;
- defect impact;
- time sensitivity;
- effort;
- available expertise;
- release objectives;
- cost of delay;
- probability of failure;
- operational consequences.

AiA publishes both:

1. the recommended priority order; and
2. the evidence and reasoning supporting the recommendation.

The authorized Product or Engineering Authority may override the recommendation, but the override becomes a recorded decision rather than an undocumented verbal instruction.

---

11. Assignment Under Nuke-M

Assignments are based on:

- required expertise;
- current ownership;
- workload;
- dependency relationships;
- code or component familiarity;
- availability;
- conflict-of-interest restrictions;
- organizational authority;
- delivery risk.

AiA may recommend an assignee or ownership group.

Human authority may approve, modify, or override that recommendation.

Assignments remain visible in the Product State. Therefore, no meeting is required to determine who is working on which unfinished item.

---

12. The 9:00 AM Product-State Publication

Nuke-M may use scheduled publications without using scheduled synchronization meetings.

For example, at 9:00 AM, AiA publishes a Product-State Brief containing:

- product health;
- changes since the previous publication;
- completed work;
- unfinished work;
- current owners;
- blocked work;
- failed builds or tests;
- unresolved events;
- pending decisions;
- release risks;
- recommended priority changes;
- each engineer's recommended next executable work.

This publication does not begin a meeting.

Engineers review the portions relevant to their responsibilities asynchronously.

At 9:15 AM, ACE-EDSO continues. In practice, engineers may already have continued executing while the publication was generated and reviewed.

The 9:00 AM publication is a convenience, not the sole source of current information. The underlying Product State continues updating in real time.

---

13. The Complete Nuke-M Product Lifecycle

Phase 1: Product Discovery

Purpose

Establish whether a product should exist and what problem it must solve.

Activities

- identify users and stakeholders;
- define the problem;
- examine current conditions;
- collect objective evidence;
- identify constraints;
- identify potential value;
- identify major risks;
- determine whether software is an appropriate solution.

AiA Responsibilities

- collect and correlate available evidence;
- identify contradictions and missing information;
- compare proposed outcomes with known constraints;
- generate questions requiring human judgment;
- maintain the evolving Product Mission.

Human Responsibilities

- supply domain knowledge;
- resolve value judgments;
- approve the product mission;
- determine acceptable risk;
- authorize continuation or termination.

Exit Conditions

- the problem is sufficiently understood;
- intended users are identified;
- the expected outcome is defined;
- major constraints are documented;
- the Product Authority authorizes product definition.

No kickoff meeting is required. A targeted event may be raised when discovery evidence contains ambiguity that cannot be resolved asynchronously.

Phase 2: Product Definition

Purpose

Translate the Product Mission into verifiable product outcomes and requirements.

Activities

- define functional requirements;
- define nonfunctional requirements;
- define security requirements;
- define privacy and compliance requirements;
- define operational expectations;
- define accessibility expectations;
- establish acceptance conditions;
- identify exclusions and deferred scope.

AiA Responsibilities

- detect inconsistent or duplicate requirements;
- trace requirements to the Product Mission;
- identify unverifiable language;
- expose missing acceptance conditions;
- maintain requirement status;
- identify affected product areas.

Human Responsibilities

- make policy and scope decisions;
- validate domain correctness;
- approve requirements;
- resolve conflicting objectives.

Exit Conditions

- sufficient requirements exist to begin architecture and execution;
- acceptance conditions are measurable;
- unresolved decisions are represented as events;
- approved requirements are visible in the Product State.

CEDA does not require all future requirements to be known before execution begins. It requires that each executable work item be sufficiently grounded before it enters execution.

Phase 3: Product Architecture

Purpose

Define the product structure needed to satisfy approved requirements.

Activities

- select architectural patterns;
- define components and boundaries;
- define data models;
- define interfaces;
- define environments;
- define integration points;
- define security controls;
- define deployment approach;
- define observability;
- identify architectural risks.

AiA Responsibilities

- map requirements to architecture;
- identify dependency chains;
- detect architectural conflicts;
- compare design decisions against standards;
- preserve architectural decisions;
- raise events for unresolved ambiguity.

Human Responsibilities

- exercise engineering judgment;
- approve consequential architecture decisions;
- accept or reject risk;
- determine exceptions.

Exit Conditions

- enough architecture exists to support safe execution;
- affected components are known;
- critical dependencies are visible;
- architectural decisions are recorded;
- initial quality gates are defined.

Architecture evolves with the product. Nuke-M does not require a single architecture meeting or a frozen design phase.

Phase 4: Work Decomposition

Purpose

Convert approved requirements and architectural needs into executable work.

Activities

- identify capabilities;
- divide capabilities into deliverable changes;
- establish dependencies;
- define acceptance conditions;
- define completion evidence;
- identify suitable owners;
- estimate effort where useful;
- assign preliminary priorities.

AiA Responsibilities

- generate or recommend work decomposition;
- identify work that is too broad or ambiguous;
- reveal dependency order;
- detect duplicate work;
- distinguish executable from non-executable items;
- populate the Work Inventory and Execution Queue.

Human Responsibilities

- validate the decomposition;
- correct domain or technical errors;
- approve high-impact work;
- decide unresolved tradeoffs.

Exit Conditions

- at least one valuable item is Ready for Execution;
- required dependencies are visible;
- ownership can be established;
- completion conditions are defined.

There is no mandatory backlog-refinement meeting. Refinement occurs continuously as AiA and humans obtain additional ground truth.

Phase 5: Execution Planning

Nuke-M does not create a fixed sprint commitment.

Instead, it creates a continuously updated Execution Queue.

AiA Responsibilities

AiA evaluates:

- what can be executed now;
- what should be executed next;
- what work unlocks other work;
- what work carries the greatest risk;
- what engineer is best positioned to execute it;
- whether parallel execution is safe;
- whether current assignments create bottlenecks.

Human Responsibilities

Authorized humans review priority and assignment recommendations when required.

Exit Conditions

- executable work is prioritized;
- ownership is known;
- dependencies are satisfied or explicitly tracked;
- engineers can begin without a planning ceremony.

Phase 6: Engineering Execution

Purpose

Produce a verifiable change to the software product.

Activities

- implementation;
- configuration;
- data changes;
- automated test creation;
- documentation required for operation or maintenance;

- code review;
- local validation;
- integration preparation.

Engineer Responsibilities

The engineer:

1. consults the current Product State;
2. selects or receives the highest-priority authorized item;
3. verifies that its requirements remain current;
4. executes the work;
5. records or generates evidence;
6. updates engineering artifacts through normal tools;
7. responds only to relevant events;
8. advances the item when completion conditions are met.

AiA Responsibilities

AiA continuously:

- observes engineering changes;
- compares implementation against requirements;
- updates progress;
- identifies scope drift;
- detects dependency effects;
- evaluates test and build results;
- recommends corrective action;
- updates product risk;
- raises targeted events when necessary.

Exit Conditions

- implementation evidence exists;
- local or preliminary verification passes;
- the item is ready for formal verification.

Phase 7: Continuous Verification

Verification is not postponed until the end of a sprint.

It occurs as soon as verifiable product changes exist.

Verification May Include

- automated tests;
- code review;
- static analysis;
- security scanning;
- integration testing;
- performance testing;
- accessibility testing;
- data validation;
- human acceptance testing;
- regulatory validation;
- deployment simulation.

AiA Responsibilities

- select applicable gates;
- collect results;
- correlate failures to requirements and changes;
- prevent unsupported advancement;
- create rework items;
- update the Product State.

Human Responsibilities

- perform judgment-dependent review;
- validate user or domain outcomes;
- decide exceptions;
- approve risk where authorized.

Exit Conditions

The work is either:

- Verified;
- returned as Rework Required;
- Blocked;
- escalated through an event.

Phase 8: Integration

Purpose

Incorporate verified work into the appropriate product baseline.

Activities

- merge verified changes;
- resolve integration conflicts;
- run integration gates;
- update dependencies;
- verify shared environments;
- update product documentation where required.

AiA Responsibilities

- determine integration readiness;
- detect conflicts;
- evaluate combined effects;
- ensure traceability;
- prevent integration when required evidence is absent.

Exit Conditions

- the change is incorporated;
- integration gates pass;
- the Product State reflects the new baseline.

Phase 9: Release Readiness

A release occurs when a coherent product state satisfies defined release conditions. It does not occur merely because a sprint ended.

Release Readiness Includes

- required capabilities are Verified and Integrated;
- release-blocking defects are resolved or formally accepted;
- security gates pass;
- deployment procedures are validated;
- rollback capability exists where required;
- operational monitoring is ready;
- release notes and user documentation are sufficient;
- required authorities approve the release.

AiA Responsibilities

- continuously calculate release readiness;
- expose missing evidence;
- identify release risk;
- recommend release, delay, or limited deployment;
- identify affected participants.

Exit Conditions

The release is:

- authorized;
- deferred;
- rejected;
- divided into smaller release units.

No release-readiness meeting is mandatory. A targeted decision event is created only when human authorization or unresolved judgment is required.

Phase 10: Deployment

Purpose

Deliver the verified release to its intended environment.

Activities

- execute deployment;
- validate configuration;
- apply data migrations;
- verify service health;
- perform smoke tests;
- confirm observability;
- monitor failure conditions;
- execute rollback when required.

AiA Responsibilities

- monitor deployment evidence;
- compare results against expected state;
- raise immediate incident events;
- identify affected responders;
- update deployment and product health.

Exit Conditions

The release is:

- successfully deployed;
- rolled back;
- partially deployed under controlled conditions;
- blocked pending corrective action.

Phase 11: Operational Observation

Deployment does not by itself prove product success.

Purpose

Determine whether the released product behaves correctly under actual operating conditions.

Observation Includes

- availability;
- error rates;
- user behavior;
- performance;
- security;
- data integrity;
- support incidents;
- business outcomes;
- requirement satisfaction.

AiA Responsibilities

- correlate telemetry with requirements and released changes;
- detect abnormal conditions;
- identify possible defects;
- generate targeted events;
- recommend rollback, correction, or continued observation.

Exit Conditions

- required observation period is complete;
- operational acceptance conditions pass;
- discovered deficiencies enter the Work Inventory;

- the release becomes Done or requires corrective execution.

Phase 12: Continuous Product Development

Conception or after the first release, Nuke-M works through the same execution cycle:

Observe product state

↓

Discover requirement, defect, risk, or opportunity

↓

Analyze and authorize

↓

Place executable work in the Execution Queue

↓

Engineer executes

↓

Verify

↓

Integrate

↓

Release

↓

Observe

↓

Continue

There is no mandatory cycle reset.

There is no artificial pause between product versions.

Product development continues for as long as authorized unfinished work exists.

Phase 13: Product Completion

A software product may be considered development-complete when:

- all approved completion-scope requirements are satisfied;
- all required capabilities are released;
- completion evidence exists;
- no unresolved release-blocking defect remains;
- security and compliance obligations are met;
- operational readiness is verified;
- documentation is sufficient;
- support and ownership arrangements exist;
- the Product Authority formally accepts completion.

Completion does not necessarily mean the software will never change again.

It may mean:

- the initial product is complete;
- a contracted scope is complete;
- a release objective is complete;
- active development is complete;
- the product has entered maintenance mode.

AiA generates a Product Completion Record linking every completion claim to supporting evidence.

Phase 14: Maintenance Mode

When active development ends, the product may enter maintenance mode.

Work may include:

- defect correction;
- security updates;
- dependency updates;
- compatibility changes;
- infrastructure maintenance;
- performance remediation;
- compliance updates;
- limited enhancements.

The same Nuke-M work states, evidence requirements, quality gates, and event-driven communication continue to apply.

Phase 15: Product Retirement

A complete lifecycle should include retirement, not merely initial delivery.

Retirement Activities

- authorize decommissioning;
- identify affected users and systems;
- preserve required records;
- migrate or archive data;
- terminate integrations safely;
- revoke credentials;
- remove infrastructure;
- satisfy legal retention rules;
- confirm that no critical dependency remains;
- publish final product state.

Exit Condition

The product is considered Retired only when evidence confirms that:

- intended service has ended;
- dependencies are resolved;
- data obligations are satisfied;
- infrastructure and access are safely decommissioned;
- responsible authority approves closure.

---

14. Nuke-M Event Model

Events replace routine meetings as the trigger for synchronization.

14.1 Informational Event

No response is required. The Product State changes and affected participants are informed asynchronously.

14.2 Action Event

A specific person must perform a defined action.

14.3 Decision Event

An authorized human must choose among alternatives.

14.4 Dependency Event

One work item cannot proceed until another condition is satisfied.

14.5 Quality Event

A build, test, review, security, or acceptance gate fails.

14.6 Architecture Event

A change conflicts with architecture or requires a consequential design decision.

14.7 Production Event

Operational behavior requires investigation or intervention.

14.8 Maintenance Event

Planned organizational or technical maintenance temporarily interrupts normal execution.

14.9 Emergency Event

Immediate action is necessary to prevent or reduce material harm.

Every event must identify:

- why interruption is justified;
- who is actually required;
- what evidence triggered it;
- what resolution is needed;
- when it is resolved.

---

15. Targeted Communication Protocol

When communication becomes necessary, Nuke-M applies the following sequence:

Event detected

↓

AiA gathers relevant evidence

↓

AiA identifies affected work and participants

↓

AiA determines whether the event can be resolved
through existing policy or evidence

↓

If yes:
Apply resolution and update Product State

↓

If no:
Notify the minimum necessary participants

↓

Participants resolve the specific issue

↓

Decision and evidence are recorded

↓

Affected work resumes

↓

Everyone else remains in execution

A real-time conversation may occur, but it is an engineering response to a defined event—not a recurring meeting.

---

16. Organizational Maintenance Windows

Nuke-M adopts the production-server analogy established by CEDA.

Example: An email server's primary mission is to receive and deliver email. A backup may interrupt or constrain normal operation, but the backup is justified because it preserves the system's long-term ability to perform its mission.

Likewise, a software engineering organization's primary mission is to produce and sustain working software.

Necessary organizational maintenance may include:

- onboarding;
- team development;
- conflict resolution;
- disaster-recovery exercises;
- compliance training;
- security exercises;
- knowledge preservation;
- organizational restructuring.

A maintenance window must be:

- intentional;
- infrequent;
- bounded;
- measurable;
- justified;
- scheduled to minimize production impact.

A routine status meeting is not maintenance because it does not preserve organizational capability when the same information already exists in the Product State.

---

17. Quality Gates

Nuke-M quality gates may apply at work-item, integration, release, deployment, and operational levels.

Typical gates include:

- requirement completeness;
- architecture conformance;
- code review;
- automated testing;
- security scanning;
- dependency validation;
- performance thresholds;
- accessibility requirements;
- regulatory compliance;
- deployment validation;
- post-release health.

A gate must define:

- what is measured;
- required threshold;
- evidence source;
- authority for exceptions;
- consequences of failure.

AiA must not mark an item complete solely because activity stopped. Completion requires evidence.

---

18. Definition of Done Under Nuke-M

A work item is Done when:

1. the approved outcome exists;
2. acceptance conditions pass;
3. applicable quality gates pass;
4. implementation is integrated;
5. required deployment is complete;
6. required operational observation passes;
7. documentation obligations are satisfied;
8. no unresolved blocking event remains;
9. completion evidence is linked;
10. the Product State reflects the verified result.

“Developer says it is done” is not evidence.

“Task moved to Done” is not evidence.

Done is an evidence-supported product state.

---

19. Metrics

Nuke-M measures the behavior and health of software production rather than attendance at ceremonies.

Relevant metrics may include:

- lead time;
- execution time;
- blocked time;
- review latency;
- build success rate;
- test pass rate;
- defect escape rate;
- deployment frequency;
- rollback rate;
- change failure rate;
- mean time to recovery;
- interruption frequency;
- interruption duration;

- dependency wait time;
- rework rate;
- release predictability;
- requirement-to-release traceability;
- product-health indicators.

Metrics must not be used blindly as employee-surveillance scores.

They are evidence for improving product flow, identifying constraints, and validating whether Nuke-M is producing better outcomes.

---

20. Multi-Product Operation

Nuke-M supports organizations developing multiple products.

AiA maintains:

- a Product State for each product;
- cross-product dependencies;
- shared engineering resources;
- shared platforms;
- organizational priorities;
- release obligations;
- production incidents;
- resource conflicts.

AiA may recommend work across products based on:

- business priority;
- severity;
- dependency impact;
- production risk;
- customer obligation;
- available expertise.

Engineers do not require a multi-product status meeting to determine what matters most. The prioritization and its reasoning are published in the shared Product State.

---

21. What Nuke-M Deems Optional

Nuke-M does not reject established software development methodologies or the practices associated with them. Instead, it removes the assumption that recurring ceremonies are always required to achieve visibility, coordination, prioritization, feedback, or continuous improvement.

It does not prohibit humans from speaking.

It requires an engineering justification for interrupting execution.

When the same objectives can be achieved through real-time product-state visibility, asynchronous review, automated evidence, or targeted event-driven communication, Nuke-M uses those mechanisms instead.

Nuke-M therefore seeks to reduce:

- unnecessary calendar-driven synchronization;
- repeated verbal reconstruction of already-known product facts;
- broad interruption of engineers who are not directly affected by an issue;
- recurring coordination activities whose purpose has already been satisfied by AiA-maintained product state.

Human communication remains essential. Nuke-M simply requires that interruptions to execution be purposeful, relevant, and proportionate to the value they create.

---

22. What Nuke-M Preserves

Nuke-M preserves the valuable objectives that traditional methodologies attempted to achieve:

- visibility;
- prioritization;
- accountability;
- adaptability;
- quality;
- collaboration;
- feedback;
- risk management;
- incremental delivery;
- continuous improvement.

It changes the mechanism.

Instead of periodically synchronizing humans to reconstruct product state, AiA continuously maintains product state and invokes humans only when human judgment or action is required.

23. The Nuke-M Lifecycle in One View

PRODUCT DISCOVERY
Identify the problem and intended outcome

↓

PRODUCT DEFINITION
Establish verifiable requirements

↓

ARCHITECTURE
Define sufficient technical structure

↓

WORK DECOMPOSITION
Create evidence-based executable work

↓

CONTINUOUS PRIORITIZATION
AiA maintains the Execution Queue

↓

ASSIGNMENT
Work is assigned according to priority,
expertise, ownership, and availability

↓

CONTINUOUS EXECUTION
Engineers design, code, test, debug,
review, validate, and deploy

↓

CONTINUOUS PRODUCT-STATE UPDATE
AiA observes and publishes real-time state

↓

EVENT DETECTED?
No → Continue execution
Yes → Notify minimum necessary participants

↓

CONTINUOUS VERIFICATION
Apply quality, security, and acceptance gates

↓

INTEGRATION
Incorporate verified work

↓

RELEASE READINESS
Evaluate evidence, risk, and authorization

↓

DEPLOYMENT
Deliver the verified product state

↓

OPERATIONAL OBSERVATION
Validate actual behavior and outcomes

↓

MORE AUTHORIZED WORK?
Yes → Return to continuous prioritization
No → Evaluate product completion

↓

PRODUCT COMPLETION
Generate evidence-supported Completion Record

↓

MAINTENANCE MODE
Sustain the product through event-driven work

↓

PRODUCT RETIREMENT
Safely conclude service and preserve obligations

---

24. Nuke-M in One Sentence

> **Nuke-M is an Ai-Assisted software development methodology under ACE-EDSO that carries a software product from discovery through retirement by continuously maintaining its real-time state, prioritizing executable work, orchestrating evidence-driven quality gates, and replacing recurring human synchronization with targeted event-driven collaboration.**

25. Nuke-M’s Defining Rule

> **When executable software work exists and no product event prevents it, engineering continues.**

That rule distinguishes Nuke-M from methodologies whose rhythm is controlled by meetings, iterations, or calendar ceremonies. Nuke-M’s rhythm is controlled by the real-time state of the software product.

Trivia: The ‘M’ in “Nuke-M” stands for ‘Meeting’. Now, you know.