Engineering Trustworthy Intelligent Systems

ETIS Educational Ecosystem Guide

Teaching, Adoption, and Stewardship

Software Engineering, Governance, and Operational Trust in the AI Era

William T. O’Connell, Ph.D.

FIRST EDITION EDUCATIONAL PRODUCT

Contents

Copyright and Use

This educational product is part of the Engineering Trustworthy Intelligent Systems educational product suite.

The ETIS book, appendices, website content, downloadable materials, figures, educational products, and related publication assets are protected by applicable copyright laws.

Educators and institutions are encouraged to use ETIS concepts, practices, and educational models in their own courses and programs. Redistribution or reproduction of substantial ETIS content, educational products, figures, appendices, downloadable materials, or instructor resources remains subject to the applicable ETIS licensing terms unless separate permission has been granted.

ETIS educational products are intended for professional and educational use. They are not substitutes for institutional policy, legal advice, regulatory review, academic governance, or professional judgment.

ETIS Educational Product Series

This document is part of the **ETIS Educational Product Series**.

The educational product series transforms *Engineering Trustworthy Intelligent Systems* from a publication into a teachable, adoptable, and stewardable engineering framework.

The series is designed for instructors, students, departments, universities, professional educators, and institutions adopting ETIS in software engineering, AI governance, responsible AI, enterprise systems, capstone, project-based, or professional-practice environments.

Product Family

The ETIS Educational Product Series includes:

Product	Primary Purpose
ETIS Educational Ecosystem Guide	Explain the full ETIS educational architecture and public product model
ETIS Instructor Course Package	Provide the instructor operating system for course design and delivery
ETIS Classroom Facilitation Guide	Help instructors run ETIS classrooms as active engineering environments
ETIS Instructor Handbook	Preserve instructor guidance, teaching judgment, and long-term stewardship practices
ETIS Student Professional Engineering Guide	Help students understand and practice professional engineering behaviors

Relationship to the ETIS Book

The ETIS book remains the authoritative doctrine.

The educational product series translates that doctrine into teaching, learning, adoption, classroom operation, and stewardship resources.

Educational products teach ETIS.

Adoption examples prove ETIS.

Educational stewardship sustains ETIS over time.

Common Educational Premise

ETIS education is built on a simple premise:

> AI can produce artifacts. Engineers create trust.

Students should not merely complete assignments.

They should develop evidence of engineering maturity.

Instructors should not merely deliver content.

They should operate educational systems that help students practice trustworthy engineering.

Shared Educational Mission

ETIS educational products teach future engineers to:

- define intent
- engineer context
- bound authority
- verify behavior
- operate reality
- explain decisions
- own outcomes

Use of This Product

This product is intended to be used as a public educational resource.

It should be read together with the ETIS book, appendices, educational ecosystem pages, instructor resources, student resources, flagship implementation guidance, and institutional adoption guidance.

ETIS Educational Ecosystem Guide

Who This Is For

This guide is for:

- instructors
- departments
- universities
- curriculum designers
- professional educators
- educational leaders
- institutional adopters
- software engineering programs
- AI governance and responsible AI programs

It is also useful for students and professional readers who want to understand how ETIS becomes teachable and adoptable.

Purpose

The ETIS Educational Ecosystem Guide explains how ETIS moves from authoritative doctrine into public educational products.

It defines the educational product model, the major audiences, the instructional boundaries, and the adoption pathways that help institutions teach trustworthy intelligent systems in the AI era.

This guide is the umbrella product for ETIS education.

Relationship to ETIS

ETIS begins as a book and framework.

The educational ecosystem turns ETIS into a teachable, repeatable, and stewardable educational system.

ETIS Book ↓ Educational Ecosystem ↓ Instructor Products Student Products Implementation Products ↓ Institutional Adoption ↓ Stewardship

Educational Philosophy

Educational work should resemble professional engineering work.

Software engineering education is no longer solely about building working software.

It is about engineering systems that can be:

- understood
- reviewed
- governed
- operated
- improved
- trusted over time

AI changes what students can produce.

It does not remove what engineers must own.

Educational Mission

ETIS education teaches future engineers to:

- define intent
- engineer context
- bound authority
- verify behavior
- operate reality
- explain decisions
- own outcomes

These are not side effects of the course.

They are the educational outcomes.

Student Transformation Model

The ETIS student transformation model is:

Student ↓ Responsible Engineer ↓ Reviewer ↓ Architect ↓ Release Defender ↓ Operational Thinker ↓ Future Trustworthy Engineer

This transformation should be observable.

Engineering maturity leaves evidence.

Grades measure performance. Maturity signals measure transformation.

Educational Product Portfolio

The public ETIS educational product portfolio includes:

Product	Audience	Purpose
ETIS Educational Ecosystem Guide	Instructors, departments, institutions	Educational product architecture and adoption overview
ETIS Instructor Course Package	Instructors	Instructor operating system
ETIS Classroom Facilitation Guide	Instructors	Classroom operations and facilitation
ETIS Instructor Handbook	Instructors	Teaching judgment, continuity, and stewardship
ETIS Student Professional Engineering Guide	Students and instructors	Student professional engineering behaviors

Instructor Products

Instructor products help educators design, operate, evaluate, and steward ETIS learning environments.

They support:

- course design
- syllabus framing
- semester sequencing
- assignment progression

- assessment
- classroom facilitation
- instructor judgment
- multi-semester stewardship

Student Products

Student products help learners understand that ETIS is not merely a course.

It is a model of professional engineering behavior.

Students learn to produce evidence of engineering ability through:

- repository-centered engineering
- AI responsibility
- team accountability
- review participation
- release defense
- operational thinking
- portfolio evidence

Flagship Implementation

The Loyola University Chicago COMP330/474 implementation is the flagship ETIS educational implementation.

It demonstrates ETIS operating in a real software engineering course.

It is proof, not doctrine.

Institutions should inherit ETIS doctrine, not mechanically copy a single implementation.

Institutional Adoption

Institutions adopt ETIS by adapting the framework responsibly to their environment.

Adoption should consider:

- academic calendar
- course level
- class size
- student maturity
- institutional policy
- AI expectations
- assessment model
- tooling environment
- instructor capacity

The goal is trustworthy adaptation.

How To Use This Guide

Use this guide to:

- understand ETIS education
- orient faculty and departments

- explain the public product portfolio
- plan adoption
- distinguish doctrine from implementation
- prepare for instructor and student products
- align education with professional engineering practice

Relationship to Other ETIS Products

This guide sits above the other educational products.

Use it first when explaining ETIS education.

Then use:

- the Instructor Course Package to design the course
- the Classroom Facilitation Guide to run the classroom
- the Instructor Handbook to steward the course
- the Student Professional Engineering Guide to orient students

Bottom Line

The ETIS Educational Ecosystem transforms authoritative ETIS doctrine into durable educational products that can be adopted, operated, improved, and stewarded over time.

Educational products teach ETIS.

Educational laboratories prove ETIS.

Part I

Educational Architecture

ETIS Educational Architecture

Purpose

This document defines the educational architecture for the ETIS (Engineering Trustworthy Intelligent Systems) Educational Ecosystem.

The purpose of this architecture is to transform ETIS from a completed book into a teachable, reusable, governable, and sustainable educational system.

The ETIS book remains the authoritative knowledge source. The Educational Ecosystem defines how that knowledge is taught, sequenced, practiced, assessed, adapted, and preserved across educational implementations.

This document is the master educational architecture reference for Phase II.

How to Use This Document

This document should be read as the architectural guide for ETIS education.

It is intended for instructors, curriculum designers, educational adopters, and project stewards who need to understand how the ETIS book becomes an educational system.

This document does not replace detailed learning maps, audience pathways, shared asset documentation, course materials, or student starter kit guidance. Instead, it explains how those pieces relate to one another.

Readers should use this document to understand:

- The purpose of the ETIS Educational Ecosystem
- The relationship between the ETIS book and educational implementations
- The major educational architecture components
- The role of shared assets and starter kits
- The professional transformation model
- The ownership boundaries that prevent duplication and drift

Mission

The mission of the ETIS Educational Ecosystem is to prepare future engineers to build, govern, operate, review, and steward trustworthy intelligent systems.

Software engineering education must now extend beyond teaching students how to build working software.

Future engineers must learn to create systems that can be:

- Understood
- Reviewed
- Governed
- Operated
- Improved
- Defended
- Trusted over time

The Educational Ecosystem exists to teach those capabilities intentionally.

Core Educational Thesis

AI capability increases engineering responsibility.

As AI systems become more capable, engineers do not become less responsible.

They become more responsible for defining intent, engineering context, bounding authority, verifying behavior, preserving evidence, explaining decisions, and owning outcomes.

The ETIS Educational Ecosystem exists to teach that responsibility.

Relationship to the ETIS Book

The ETIS book is complete, frozen, and authoritative.

The Educational Ecosystem does not rewrite the book.

The Educational Ecosystem translates the book into educational structures.

Relationship model:

ETIS Book

↓

Educational Architecture

↓

Learning Models

↓

Shared Assets

↓

Educational Implementations

↓

Student Starter Kits

↓

Student Engineering Practice

The book defines the framework.

Educational architecture defines how the framework is taught.

Shared assets provide reusable educational building blocks.

Adoption examples demonstrate concrete implementation.

Student Starter Kits provide assembled engineering environments for learners.

Educational System Architecture

The ETIS Educational Ecosystem is organized around the following structure:

education/

|— README.md

|— educational_architecture/

|— learning_models/

|— shared ETIS educational assets

|— student starter kit

‘— adoption_examples/

Each directory represents a distinct architectural responsibility.

Directory Responsibilities

educational_architecture/ Defines how ETIS is taught.

Owns:

- Educational architecture
- Educational design principles
- Audience learning paths
- Chapter learning maps
- Educational ownership boundaries

This directory governs the educational system itself.

learning_models/ Defines models of learner development and professional progression.

Owns:

- Professional Transformation Model
- Engineering Maturity Model
- Two-Cycle Engineering Model
- Software Engineering Learning Progression

This directory explains how learners mature over time.

shared ETIS educational assets Contains reusable educational assets.

Owns:

- Playbooks
- Templates
- Workflows
- AI Engineering assets
- Assessment assets

This directory provides reusable educational building blocks.

student starter kit Contains assembled student engineering environments.

Owns:

- Starter kit structure
- Student-facing repository environments
- Practice scaffolding
- Engineering workspace organization

Student Starter Kits are products, not document collections.

adoption_examples/ Contains implementation-specific educational examples.

Owns:

- Course-specific materials

- Institution-specific implementations
- Syllabi
- Weekly schedules
- Assignments
- Rubrics
- Slides
- Implementation notes

The current flagship implementation is Loyola COMP330.

Flagship Implementation

Loyola University Chicago COMP 330 — Software Engineering is the flagship ETIS educational implementation.

COMP330 is important because it serves as the initial real-world educational implementation of ETIS Phase II.

However, COMP330 is not the Educational Ecosystem itself.

COMP330:

- Consumes educational architecture
- Consumes shared assets
- Uses learning models
- Uses the Student Starter Kit
- Contributes new educational assets
- Preserves Loyola provenance

Course-specific materials belong in:

adoption_examples/loyola_comp330/

Reusable assets belong outside COMP330, usually in:

shared ETIS educational assets

This boundary prevents reusable educational intellectual property from becoming trapped inside one course implementation.

Professional Transformation Architecture

The ETIS Educational Ecosystem is organized around professional transformation.

The target progression is:

Student

↓

Responsible Engineer

↓

Reviewer

↓

Architect

↓

Release Defender

↓

Operational Thinker

↓

Future Trustworthy Engineer

This progression is not merely aspirational.

It should influence:

- Course design
- Assignment sequencing
- Starter kit structure
- Assessment rubrics
- Classroom exercises
- Review practices
- Student reflection
- Instructor feedback

The educational goal is not only for students to complete software projects.

The educational goal is for students to mature into engineers capable of taking responsibility for trustworthy systems.

Educational Responsibility Model

ETIS education should teach learners to:

Define intent Students must learn to clarify what a system is supposed to accomplish and why it matters.

Engineer context Students must learn that context shapes system behavior, AI behavior, review quality, operational readiness, and governance.

Bound authority Students must learn to define what people, systems, tools, and AI agents are allowed to do.

Verify behavior Students must learn that claims require evidence.

Operate reality Students must learn that systems must survive use, failure, change, and operational constraints.

Explain decisions Students must learn to make decisions understandable to others.

Own outcomes Students must learn that engineering responsibility cannot be delegated away.

Educational Design Philosophy

ETIS education should teach systems rather than isolated tasks.

Traditional software engineering education often emphasizes deliverables.

ETIS emphasizes the relationships among deliverables, decisions, evidence, review, governance, operations, and accountability.

Students should understand not only what artifact they are creating, but why that artifact matters within the engineering system.

A requirements document is not paperwork.

It is intent preservation.

An architecture document is not a diagram.

It is decision structure.

A test plan is not a checklist.

It is evidence design.

A release readiness review is not a presentation.

It is an engineering defense.

A postmortem is not a blame document.

It is organizational learning.

Educational Asset Architecture

Educational assets are first-class ETIS artifacts.

They should be intentionally created, named, governed, normalized, versioned, and preserved.

Authoritative format:

Markdown (.md)

Distribution format:

PDF (.pdf)

Optional editable format:

Word (.docx)

Markdown is the source of truth.

Word and PDF files may exist during transition, distribution, or archival use, but they are not authoritative.

Shared Asset Architecture

Shared assets are reusable educational intellectual property.

They are organized into five categories:

shared ETIS educational assets

|— playbooks/

|— templates/

|— workflows/

|— ai_engineering/

‘— assessment/

Each category has a distinct educational purpose.

Playbooks Teach professional engineering behaviors.

Core question:

How should engineers behave?

Templates Teach engineering evidence creation.

Core question:

What evidence should engineers create?

Workflows Teach engineering flow and process.

Core question:

How does engineering work move through a system?

AI Engineering Teach responsible human-AI collaboration.

Core question:

How should engineers responsibly collaborate with AI?

Assessment Teach evaluation of engineering maturity.

Core question:

How do we evaluate engineering responsibility?

Student Starter Kit Architecture

Student Starter Kits are assembled engineering environments.

They are products.

They are not document folders.

A Student Starter Kit should provide learners with a structured environment for practicing repository-centered engineering.

The current flagship starter kit is:

student starter kitcomp330/

Starter Kits may contain:

- Repository structure
- Documentation folders
- Source folders
- Test folders
- Script folders
- Data folders
- Evidence folders
- README guidance
- Starter policies
- Engineering scaffolding

Student Starter Kits consume shared assets but should not permanently duplicate them.

The starter kit is where students practice.

Shared assets are where reusable educational intellectual property lives.

Repository-Centered Engineering

Repository-centered engineering is central to ETIS education.

The repository is not merely where code is stored.

The repository is where engineering evidence accumulates.

Students should learn to use repositories to:

- Define intent
- Preserve requirements
- Document architecture
- Record decisions
- Track risks
- Show testing evidence
- Capture AI usage
- Conduct reviews
- Prepare releases
- Preserve operational knowledge

The repository becomes the student's engineering memory system.

This is one of the most important educational shifts in ETIS.

Evidence-Centered Education

ETIS education should teach students that important engineering claims require evidence.

Students should not merely say:

"The system works."

They should be able to show:

- Requirements evidence
- Design evidence
- Test evidence
- Review evidence
- Traceability evidence
- AI verification evidence
- Release readiness evidence
- Operational readiness evidence

Evidence-centered education prepares students for professional engineering environments where decisions must be reviewed, defended, audited, and improved.

AI-Assisted Engineering Education

AI is a major educational concern within ETIS.

Students should learn to use AI responsibly, but they should not learn to outsource engineering responsibility.

AI may assist with:

- Brainstorming
- Drafting
- Code explanation
- Test generation
- Review preparation
- Documentation improvement
- Risk identification
- Alternative analysis

However, students remain responsible for:

- Understanding
- Verification
- Accuracy
- Decisions
- Integration
- Evidence
- Final outcomes

The governing principle is:

AI proposes; engineers verify.

AI should be taught as a bounded engineering collaborator, not an authority.

Two-Cycle Engineering Model

The Two-Cycle Engineering Model is foundational to the COMP330 implementation and may be reused in future implementations.

The model distinguishes between two major project learning phases.

Cycle 1 Cycle 1 asks:

Can it work?

Students focus on establishing core feasibility, basic functionality, team coordination, and initial engineering structure.

Cycle 2 Cycle 2 asks:

Can it survive?

Students focus on quality, reviewability, operational readiness, traceability, release readiness, risk, and sustainability.

This model teaches students that working software is not the finish line.

A system must also be maintainable, reviewable, governable, and operationally credible.

39-Chapter Consumption Architecture

The ETIS book contains 39 chapters.

Not every educational implementation should consume all 39 chapters equally.

Different audiences require different pathways.

For example, COMP330 primarily emphasizes Parts I and II while selectively introducing Parts III and IV.

Other implementations may emphasize:

- AI governance
- Operational readiness
- Architecture review
- Release governance
- Incident response
- Human oversight
- Stewardship

The purpose of educational architecture is to preserve the authority of the complete ETIS book while allowing responsible educational adaptation.

Audience Architecture

ETIS may serve multiple audiences over time.

Potential audiences include:

- Undergraduate software engineering students
- Graduate software engineering students
- Capstone teams
- Early-career software engineers
- Professional engineers
- Software architects
- Technical leads
- Engineering managers
- AI governance teams
- Review boards
- Organizational training programs

These audiences should not all receive identical learning paths.

Audience learning paths should be designed intentionally.

Assessment Architecture

Assessment in ETIS should evaluate engineering maturity, not merely task completion.

Students should be assessed on their ability to:

- Define intent
- Create evidence
- Explain decisions
- Work professionally
- Manage risk
- Use AI responsibly
- Review engineering work
- Demonstrate release readiness
- Think operationally
- Own outcomes

Assessment should mirror professional engineering evaluation.

The central question should not be only:

Did the software work?

The better question is:

Is the engineering work understandable, reviewable, governable, sustainable, and trustworthy?

Instructor Architecture

The ETIS Educational Ecosystem should eventually support instructors beyond the original COMP330 implementation.

Instructor-facing materials may include:

- Course setup guidance
- Weekly sequencing guidance
- Assignment guidance
- Rubric guidance
- Classroom exercises
- Starter kit usage guidance
- Assessment guidance
- Adoption notes
- Implementation patterns

Instructor materials should help adopters teach ETIS without requiring them to reconstruct the educational architecture from scratch.

Classroom Exercise Architecture

Classroom exercises should reinforce ETIS behaviors through practice.

Potential exercise areas include:

- Requirements review
- Architecture review
- AI usage review
- Pull request review
- Release readiness defense
- Risk review
- Incident response
- Postmortem analysis
- Operational readiness review

Exercises should help students practice engineering judgment, not merely recall concepts.

Ownership Boundaries

Clear ownership boundaries prevent educational drift.

Educational architecture owns system design. It defines how ETIS is taught.

Learning models own progression models. They explain how learners mature.

Shared assets own reusable artifacts. They contain reusable teaching assets.

Student Starter Kits own assembled practice environments. They give students structured engineering spaces.

Adoption examples own implementation-specific materials. They preserve course-specific and institution-specific context.

These boundaries should be preserved unless a significant architectural event justifies change.

Provenance

The current educational assets originate primarily from Loyola University Chicago COMP330.

This provenance should be preserved.

Loyola provenance matters because it records the implementation history of the Educational Ecosystem.

Educational assets should not be prematurely generalized.

Reusable assets may become more implementation-neutral over time, but their origin should remain traceable.

Related Documents

This document is part of the `educational_architecture` directory.

Related documents include:

- `educational_design_principles.md`
- `audience_learning_paths.md`
- `39_chapter_learning_map.md`

Those documents provide more detailed guidance on educational design rules, audience-specific pathways, and chapter-level learning use.

Core Thesis

The ETIS Educational Ecosystem exists because trustworthy engineering must be taught intentionally.

Future engineers will not be judged solely by whether they can produce software.

They will be judged by whether they can define intent, engineer context, bound authority, verify behavior, operate reality, explain decisions, and own outcomes.

This architecture exists to make that education possible, repeatable, governable, and sustainable.

Educational Design Principles

Purpose

This document defines the educational design principles that govern the ETIS (Engineering Trustworthy Intelligent Systems) Educational Ecosystem.

These principles guide the creation of educational architecture, learning models, shared assets, student starter kits, adoption examples, classroom exercises, assessment mechanisms, and instructor materials.

The purpose of this document is to preserve coherence as the Educational Ecosystem grows.

Educational design decisions should reinforce the ETIS mission rather than merely produce more instructional content.

Core Premise

Software engineering education must evolve.

Students no longer need to learn only how to build working software.

They must learn how to build systems that can be understood, reviewed, governed, operated, improved, and trusted over time.

AI increases this responsibility.

The educational system must therefore teach engineering judgment, evidence, accountability, governance, and operational thinking as central parts of software engineering practice.

Principle 1: Teach Systems, Not Isolated Tasks

ETIS education should help learners understand engineering work as a connected system.

Assignments, templates, workflows, reviews, tests, releases, and postmortems should not be treated as isolated activities.

They should be taught as related parts of an engineering lifecycle.

A requirement influences architecture.

Architecture influences implementation.

Implementation influences testing.

Testing influences release readiness.

Release readiness influences operations.

Operations influence future learning.

Students should see those relationships.

The goal is not to complete tasks.

The goal is to understand how engineering work moves through a system.

Principle 2: Teach Evidence, Not Paperwork

Engineering artifacts should not be presented as administrative paperwork.

They are evidence.

A requirements document preserves intent.

An architecture document explains decision structure.

A risk register exposes uncertainty.

A test plan defines verification strategy.

A release readiness review supports engineering defense.

A postmortem preserves learning.

Students should understand why artifacts matter.

The goal is not to produce documents.

The goal is to create durable engineering evidence.

Principle 3: Teach Engineering Judgment, Not Procedural Compliance

Students should not be trained merely to follow steps.

They should learn to make defensible engineering decisions.

Procedures are useful, but judgment determines whether procedures are applied intelligently.

ETIS education should ask students to explain assumptions, compare alternatives, identify risks, justify tradeoffs, and defend decisions.

The goal is not compliance.

The goal is responsible judgment.

Principle 4: Teach Responsibility, Not Activity

Educational work should not reward activity for its own sake.

Students should not confuse effort, volume, or tool usage with engineering quality.

A large amount of code does not prove good engineering.

A long document does not prove clear thinking.

Frequent AI use does not prove productivity.

Many commits do not prove progress.

ETIS education should evaluate whether students understand, verify, explain, and own the outcomes of their work.

The goal is not busyness.

The goal is responsibility.

Principle 5: Teach AI Collaboration, Not AI Dependency

AI should be taught as a bounded engineering collaborator.

Students may use AI to assist with brainstorming, explanation, drafting, testing, review preparation, and alternative analysis.

However, students remain responsible for understanding, verification, accuracy, integration, and final outcomes.

The governing principle is:

AI proposes; engineers verify.

Students should learn how to use AI without surrendering engineering ownership.

The goal is not to maximize AI usage.

The goal is to improve trustworthy engineering.

Principle 6: Teach Context Engineering

Context shapes engineering behavior.

Poor context produces poor decisions.

Students should learn to define and maintain context for people, teams, repositories, reviewers, AI systems, and future maintainers.

Context includes:

- Goals
- Constraints
- Assumptions
- Risks
- Requirements
- Architecture
- Decisions
- Evidence
- Operational expectations

ETIS education should make context visible and intentional.

The goal is not simply communication.

The goal is controlled understanding.

Principle 7: Teach Bounded Authority

Modern engineering systems involve people, tools, AI systems, automation, repositories, workflows, and operational processes.

Each participant requires boundaries.

Students should learn to define what different actors may do, what they may not do, and where human review is required.

This applies to:

- Team roles
- AI assistance
- Pull requests
- Release decisions
- Operational procedures
- Review responsibilities
- Automation workflows

The goal is not control for its own sake.

The goal is safe delegation.

Principle 8: Teach Repository-Centered Engineering

The repository should be taught as the center of engineering work.

It is not merely a code container.

The repository is where engineering memory accumulates.

Students should learn to use repositories to preserve:

- Intent
- Requirements
- Architecture
- Decisions
- Risks
- Tests
- Reviews
- AI usage
- Release evidence
- Operational knowledge

The goal is not repository mechanics.

The goal is repository-centered engineering discipline.

Principle 9: Teach Review as Engineering Work

Review is not a ceremonial checkpoint.

Review is engineering work.

Students should learn how to review requirements, architecture, code, tests, releases, AI usage, risks, and operational readiness.

They should also learn how to be reviewed.

Review teaches accountability, communication, evidence, judgment, and improvement.

The goal is not approval.

The goal is better engineering.

Principle 10: Teach Release Readiness, Not Demo Readiness

A demo can show that something appears to work.

It does not prove that the system is ready.

Students should learn the difference between demonstration and release readiness.

Release readiness requires evidence of:

- Requirements coverage
- Testing
- Traceability
- Known limitations
- Risk awareness
- Operational expectations
- Supportability
- Responsible AI usage
- Team understanding

The goal is not an impressive demo.

The goal is defensible release judgment.

Principle 11: Teach Operations Early

Operations should not be treated as something that happens after the course ends.

Students should learn early that systems exist in operational environments.

Even small student projects can teach operational thinking through:

- Runbooks
- Known limitations
- Incident response notes
- Observability notes
- Postmortems
- Release notes
- Support expectations

The goal is not production operations at full professional scale.

The goal is operational awareness.

Principle 12: Teach Professional Communication

Communication is engineering work.

Students should learn how to communicate status, risks, blockers, decisions, assumptions, requests, and evidence.

Professional communication supports trust.

Poor communication creates engineering risk.

ETIS education should treat communication as a core engineering capability rather than a soft skill.

The goal is not more communication.

The goal is useful communication.

Principle 13: Teach Team Accountability

Software engineering is a team discipline.

Students should learn that teams succeed through coordination, reliability, clarity, and shared responsibility.

ETIS education should reinforce:

- Role clarity
- Working agreements
- Meeting discipline
- Traceability
- Peer review
- Risk visibility
- Escalation
- Mutual support

The goal is not group work.

The goal is professional teamwork.

Principle 14: Teach Progressive Maturity

Students should not be expected to demonstrate full professional maturity immediately.

Educational design should support progression.

Students move from basic participation toward responsible engineering ownership.

The ecosystem should support growth across stages such as:

Student

↓

Responsible Engineer

↓

Reviewer

↓

Architect

↓

Release Defender

↓

Operational Thinker

↓

Future Trustworthy Engineer

The goal is not perfection at the beginning.

The goal is visible professional growth.

Principle 15: Teach Adaptation Without Losing Authority

Different audiences should consume ETIS differently.

An undergraduate software engineering course, graduate seminar, capstone program, professional workshop, and organizational training program should not all use the same pathway.

However, adaptation should not weaken the authority of the ETIS framework.

Educational implementations may adapt sequencing, emphasis, assignments, and depth.

They should not discard the core principles of trustworthy engineering.

The goal is not one-size-fits-all education.

The goal is governed adaptability.

Principle 16: Preserve Provenance

Educational assets should preserve their origin.

Current ETIS educational assets primarily originate from Loyola University Chicago COMP 330 — Software Engineering.

That provenance matters.

It records where the asset came from, what implementation shaped it, and how the ecosystem evolved.

Assets should not be prematurely generalized.

The goal is not generic content.

The goal is reusable content with memory.

Principle 17: Prefer Reuse Over Duplication

Reusable educational assets should live in shared asset locations.

Course-specific material should live in implementation-specific locations.

Duplicating assets across implementations creates drift.

When an asset has value beyond a single course, it should be normalized and placed where future implementations can consume it.

The goal is not centralization for its own sake.

The goal is maintainable reuse.

Principle 18: Preserve Ownership Boundaries

The Educational Ecosystem depends on clear ownership boundaries.

Educational architecture defines how ETIS is taught.

Learning models define learner progression.

Shared assets provide reusable educational materials.

Student Starter Kits provide assembled practice environments.

Adoption examples preserve implementation-specific materials.

Blurring these boundaries creates confusion and duplication.

The goal is not rigid bureaucracy.

The goal is sustainable architecture.

Principle 19: Optimize for Long-Term Stewardship

Educational assets should be designed for future instructors, students, maintainers, and adopters.

A useful asset should remain understandable after the original creator is no longer present.

This requires clear naming, durable structure, appropriate metadata, readable prose, and preserved rationale.

The goal is not short-term convenience.

The goal is educational stewardship.

Principle 20: Design for Trustworthy Engineering Outcomes

Every educational asset should ultimately support trustworthy engineering.

Before creating or revising an asset, ask:

- Does this help learners define intent?
- Does this help learners engineer context?
- Does this help learners bound authority?

- Does this help learners verify behavior?
- Does this help learners operate reality?
- Does this help learners explain decisions?
- Does this help learners own outcomes?

If the answer is no, the asset may not belong in the ETIS Educational Ecosystem.

Application Guidance

These principles should be applied when creating or revising:

- Educational architecture documents
- Learning models
- Shared assets
- Student Starter Kits
- Adoption examples
- Assignments
- Rubrics
- Classroom exercises
- Instructor materials
- Course implementation notes

The principles are intended to guide judgment.

They should not be used as a mechanical checklist.

Relationship to Other Educational Architecture Documents

This document defines the design principles for ETIS education.

Related documents include:

- `ETIS_Educational_Architecture.md`
- `audience_learning_paths.md`
- `39_chapter_learning_map.md`

The master architecture document explains the overall educational system.

The audience learning paths explain how different audiences consume ETIS.

The 39-chapter learning map explains how the ETIS book supports different educational pathways.

Core Thesis

Educational design is engineering design.

The way ETIS is taught should reflect the same principles ETIS teaches.

If ETIS teaches evidence, governance, accountability, context, review, operations, and stewardship, then ETIS educational assets must be designed with those same commitments.

The Educational Ecosystem should not merely describe trustworthy engineering.

It should model it.

Audience Learning Paths

Purpose

This document defines how different audiences consume the ETIS (Engineering Trustworthy Intelligent Systems) framework.

The ETIS book remains complete, frozen, and authoritative.

However, not every audience should consume all 39 chapters equally.

Different audiences have different educational goals, levels of engineering maturity, time constraints, responsibilities, and operational needs.

The purpose of this document is to provide intentional learning pathways without weakening the authority of the full ETIS framework.

This document is a consumption architecture, not a curriculum.

Core Philosophy

The ETIS book is a complete engineering framework.

Educational implementations should adapt emphasis, sequencing, depth, and examples while preserving the integrity of the framework itself.

The educational question is not:

Which chapters should we remove?

The educational question is:

Which chapters should we emphasize for a particular audience?

Every audience should understand that the complete framework exists beyond their immediate learning path.

Educational Consumption Model

Educational consumption occurs through layered exposure.

ETIS Book

↓

Audience Learning Path

↓

Educational Implementation

↓

Shared Assets

↓

Student Starter Kit

↓

Applied Engineering Practice

The learning path sits between the complete ETIS framework and the educational implementation.

It guides emphasis without changing the authority of the framework.

Educational Progression Philosophy

ETIS education is intentionally progressive.

Learners should progressively expand their engineering perspective over time.

Early exposure often emphasizes:

- Intent
- Requirements
- Architecture
- Team collaboration
- Repository discipline

Intermediate exposure often emphasizes:

- Reviews
- Traceability
- Quality
- Release readiness
- Operations

Advanced exposure often emphasizes:

- Governance
- Human oversight
- AI delegation
- Stewardship
- Organizational trust

The educational objective is to progressively widen engineering responsibility.

Audience Categories

The following audiences represent current and anticipated ETIS consumers.

Additional audiences may emerge over time.

Undergraduate Software Engineering Students

Primary Goal Build foundational software engineering discipline.

Educational Emphasis Students should learn that software engineering extends beyond implementation.

Key areas include:

- Requirements
- Architecture
- Teamwork
- Repository-centered engineering
- Traceability
- Testing
- Reviews
- AI responsibility
- Release readiness

- Operational awareness

Recommended Exposure Heavy emphasis:

- Part I
- Part II

Selective exposure:

- Part III
- Part IV

Educational Outcome Students should leave understanding that working software is only one component of trustworthy engineering.

Target transformation:

Student

↓

Responsible Engineer

Graduate Software Engineering Students

Primary Goal Develop deeper engineering judgment and systems thinking.

Educational Emphasis Graduate students should expand beyond implementation and focus more heavily on:

- Tradeoffs
- Governance
- Review systems
- AI collaboration
- Operational thinking
- Human oversight
- Organizational implications

Recommended Exposure Heavy emphasis:

- Parts I
- II
- III

Moderate emphasis:

- Part IV

Educational Outcome Students should begin thinking like reviewers and architects rather than individual contributors.

Target transformation:

Responsible Engineer

↓

Reviewer

↓

Architect

Capstone Teams

Primary Goal Integrate multiple engineering disciplines into a sustained project experience.

Educational Emphasis Capstone teams should focus heavily on engineering integration.

Areas of emphasis include:

- Team coordination
- Requirements
- Architecture
- Testing
- Traceability
- Risk management
- Release readiness
- Operations

Recommended Exposure Heavy emphasis:

- Parts I
- II
- III

Selective emphasis:

- Part IV

Educational Outcome Students should understand how engineering systems survive beyond implementation.

Target transformation:

Responsible Engineer

↓

Reviewer

↓

Release Defender

Early-Career Software Engineers

Primary Goal Transition from individual contributor to professional engineer.

Educational Emphasis Early-career engineers should expand their perspective beyond assigned tasks.

Areas of emphasis include:

- Communication
- Traceability
- Reviews
- Risk awareness
- AI usage governance

- Operational awareness
- Engineering accountability

Recommended Exposure Balanced emphasis:

- Parts I
- II
- III

Selective emphasis:

- Part IV

Educational Outcome Engineers should understand how their work affects larger systems.

Target transformation:

Responsible Engineer

↓

Reviewer

Professional Engineers

Primary Goal Expand engineering responsibility across larger systems.

Educational Emphasis Areas of emphasis include:

- Governance
- Operational thinking
- Human oversight
- AI collaboration
- Release readiness
- Incident response
- Stewardship

Recommended Exposure Balanced emphasis across all four parts.

Educational Outcome Engineers should begin thinking beyond implementation and into organizational responsibility.

Target transformation:

Reviewer

↓

Architect

↓

Operational Thinker

Software Architects

Primary Goal Strengthen system-wide engineering leadership.

Educational Emphasis Areas of emphasis include:

- Architecture
- Governance
- AI delegation
- Human oversight
- Explainability
- Release governance
- Stewardship

Recommended Exposure Heavy emphasis:

- Parts II
- III
- IV

Moderate emphasis:

- Part I

Educational Outcome Architects should strengthen their ability to create understandable and governable systems.

Target transformation:

Architect

↓

Release Defender

↓

Operational Thinker

Technical Leads and Engineering Managers

Primary Goal Lead engineering systems and people simultaneously.

Educational Emphasis Areas of emphasis include:

- Team accountability
- Governance
- AI policy
- Release governance
- Operational readiness
- Stewardship

Recommended Exposure Heavy emphasis:

- Parts III
- IV

Moderate emphasis:

- Parts I
- II

Educational Outcome Leaders should learn how to create trustworthy engineering environments.

Target transformation:

Operational Thinker

↓

Future Trustworthy Engineer

AI Governance Teams and Review Boards

Primary Goal Govern intelligent systems responsibly.

Educational Emphasis Areas of emphasis include:

- Human oversight
- AI delegation
- Explainability
- Governance
- Stewardship
- Operational trust

Recommended Exposure Heavy emphasis:

- Parts III
- IV

Selective emphasis:

- Parts I
- II

Educational Outcome Participants should learn how to govern intelligent systems without becoming disconnected from engineering realities.

Target transformation:

Reviewer

↓

Operational Thinker

↓

Future Trustworthy Engineer

Organizational Training Programs

Primary Goal Improve engineering culture at scale.

Educational Emphasis Organizations should focus on creating shared engineering language.

Areas of emphasis include:

- Governance
- Evidence
- AI responsibility
- Release readiness

- Operations
- Stewardship

Recommended Exposure Balanced emphasis across all four parts.

Educational Outcome Organizations should establish common engineering behaviors and expectations.

Target transformation:

Engineering Organization

↓

Trustworthy Engineering Organization

Learning Path Adaptation Rules

Learning paths may adapt:

- Emphasis
- Sequencing
- Examples
- Exercises
- Depth

Learning paths should not adapt:

- Core ETIS principles
- Professional transformation goals
- Engineering responsibility expectations
- The authority of the ETIS book

This distinction preserves framework integrity.

Relationship to Educational Implementations

Learning paths are consumed by educational implementations.

Current flagship implementation:

adoption_examples/ ‘— loyola_comp330/

Different implementations may use different pathways while preserving ETIS consistency.

Learning paths should guide implementations rather than constrain them.

Relationship to Shared Assets

Learning paths influence which shared assets are emphasized.

Examples:

Undergraduate implementations may emphasize:

- Playbooks
- Templates
- Workflows

Professional training implementations may emphasize:

- AI Engineering
- Assessment
- Governance workflows

Shared assets remain reusable regardless of audience.

Relationship to the 39-Chapter Learning Map

This document explains who ETIS is being taught to.

The `39_chapter_learning_map.md` document explains how the ETIS book supports those audiences.

The two documents should always be read together.

Future Evolution

New audiences will emerge over time.

Potential future audiences include:

- Data science programs
- Cybersecurity programs
- Information systems programs
- MBA technology programs
- Executive leadership programs
- Government training programs

New audiences should be added only when they represent meaningful educational differences.

Core Thesis

The ETIS book is one framework.

There are many legitimate ways to learn it.

Educational pathways should adapt to the learner while preserving the authority of trustworthy engineering itself.

ETIS 39-Chapter Learning Map

Purpose

The **ETIS 39-Chapter Learning Map** translates the complete structure of *Engineering Trustworthy Intelligent Systems* into educational use.

The ETIS book is intentionally comprehensive. It is not expected that every course, workshop, professional training program, or adoption model will use all 39 chapters with equal emphasis.

This map helps instructors, students, engineers, leaders, and organizations decide how to use the book based on learning goals, professional roles, available instructional time, and desired engineering outcomes.

This document is not a table of contents.

It is an educational consumption map.

Relationship to the ETIS Book

The ETIS book remains complete, frozen, and authoritative.

This learning map does not reduce, replace, or rewrite the book.

Instead, it helps educational adopters decide how to emphasize the book responsibly.

A course may emphasize selected chapters.

A workshop may use a chapter cluster.

A professional training program may follow a role-specific path.

A student project may use only the chapters needed for a specific engineering phase.

That selective use is acceptable when it remains connected to the larger ETIS framework.

Relationship to Audience Learning Paths

This document should be read together with:

audience_learning_paths.md

The audience learning paths explain **who** is consuming ETIS.

This document explains **how the 39 chapters support educational use**.

Together, the two documents help instructors and adopters make responsible decisions about emphasis, sequencing, and depth.

Core Educational Principle

ETIS is a cumulative professional transformation framework.

The full book moves the reader through a progression:

Foundation

↓

Engineering

↓

Construction

↓

Validation

↓

Operation

↓

Governance

↓

Stewardship

Not every learner will travel through the entire progression at the same depth.

However, every ETIS learning path should preserve the central discipline:

Trustworthy intelligent systems require evidence, review, governance, operational awareness, human accountability, and stewardship over time.

The goal is not to read chapters for coverage.

The goal is to develop trustworthy engineering capability.

Learning Emphasis Levels

This map uses four levels of educational emphasis.

Core A chapter should be taught directly and connected to assignments, exercises, reviews, or assessments.

Supporting A chapter should be introduced and referenced, but may not require full instructional depth.

Selective A chapter should be used when relevant to the audience, project, or implementation.

Capstone A chapter should be used to frame professional identity, future practice, synthesis, or reflection.

These levels help instructors avoid the false choice between teaching everything equally or omitting important material entirely.

The Four-Part Learning Architecture

ETIS is organized into four major educational movements.

Book Part	Chapters	Educational Role	Core Learner Transformation
Part I — Foundations of Trustworthy Engineering	1-7	Establishes the engineering worldview	From programmer to responsible engineer
Part II — Engineering Construction	8-22	Teaches disciplined construction, evidence, review, and release defense	From builder to reviewer and release defender

Book Part	Chapters	Educational Role	Core Learner Transformation
Part III — Operational Trust	23-32	Extends engineering into operation, runtime evidence, governance, and organizational confidence	From release defender to operational thinker
Part IV — Trustworthy Intelligent Systems	33-39	Addresses intelligent systems, oversight, context, complexity, and stewardship	From operational thinker to future trustworthy engineer

The four parts should be understood as a professional development arc.

Part I establishes why trustworthy engineering matters.

Part II teaches how disciplined engineering work is created and defended.

Part III teaches what happens when systems enter operational reality.

Part IV teaches how engineers steward intelligent systems in an AI-shaped future.

Part I Learning Role: Foundations of Trustworthy Engineering

Part I establishes the worldview required for ETIS.

Learners encounter software engineering as a sociotechnical, evidence-centered, AI-aware, accountability-driven discipline.

Educational Function Part I teaches learners why software engineering must be more than coding.

It introduces failure, complexity, uncertainty, AI disruption, judgment, oversight, communication, review, and accountability.

Primary Learner Shift

Programmer

↓

Responsible Engineer

Typical Educational Use Part I should usually be introduced early in a course, workshop, or training path.

It provides the vocabulary needed for the rest of ETIS.

Chapters

Ch.	Chapter Title	Educational Role	Learner Capability
1	Engineering Trustworthy Intelligent Systems	Establishes the ETIS worldview	Understand trustworthiness as an engineering responsibility
2	Why Software Projects Fail	Explains why discipline is necessary	Recognize failure patterns in complexity, coordination, quality, and governance

Ch.	Chapter Title	Educational Role	Learner Capability
3	Complexity, Coordination, and Sociotechnical Systems	Frames systems as people-process-technology environments	Analyze sociotechnical risk and coordination complexity
4	Lifecycle Models and Engineering Under Uncertainty	Connects lifecycle choice to uncertainty and risk	Select lifecycle approaches based on consequence and uncertainty
5	AI Changes the Software Lifecycle	Introduces AI-era lifecycle disruption	Understand how AI affects requirements, design, coding, testing, and operations
6	Engineering Judgment and Human Oversight	Establishes accountability in AI-assisted work	Apply human judgment to generated or automated outputs
7	Teams, Communication, Review, and Accountability	Builds team engineering discipline	Coordinate work through roles, communication, review, and accountability

Evidence Learners Should Produce Learners should be able to produce evidence such as:

- Engineering responsibility reflections
- Failure-pattern analysis
- Team working agreements
- Communication expectations
- AI-use expectations
- Review participation evidence
- Initial project standards

Part II Learning Role: Engineering Construction

Part II teaches learners how to move from project launch through requirements, planning, architecture, implementation, review, testing, and release defense.

This part is central to undergraduate software engineering education and the COMP330 implementation.

Educational Function Part II teaches disciplined construction.

It connects intent, artifacts, repositories, decisions, AI-assisted work, testing, traceability, and release evidence.

Primary Learner Shift

Builder
↓
Reviewer
↓
Release Defender

Typical Educational Use Part II should usually carry the heaviest instructional load in a software engineering course.

It directly supports project checkpoints, repository work, review practices, testing, and final release defense.

Chapters

Ch.	Chapter Title	Educational Role	Learner Capability
8	Project Launch and Engineering Standards	Begins controlled engineering work	Establish team standards, workflow, and engineering expectations
9	Repository-Centered Engineering	Defines the repository as engineering memory	Use repositories as evidence systems, not just code storage
10	Requirements, Stakeholders, and Engineering the Right Problem	Teaches requirements discipline	Translate stakeholder needs into reviewable requirements
11	AI-Assisted Requirements Engineering	Shows responsible AI use in requirements	Use AI to critique and improve requirements without surrendering judgment
12	Planning, Estimation, Risk, and Tradeoffs	Connects planning to reality	Make work, estimates, risks, and tradeoffs visible
13	Architecture Fundamentals	Establishes structural design thinking	Define components, interfaces, responsibilities, and boundaries
14	Intelligent Systems Architecture	Extends architecture into AI-enabled systems	Design context, tools, permissions, orchestration, and governance boundaries
15	Architecture Reviews and ADRs	Makes decisions reviewable	Create and evaluate ADRs, review findings, and design tradeoffs
16	AI-Assisted Design and Coding	Governs AI-assisted construction	Use AI for implementation while preserving ownership and verification
17	Pull Requests, Reviews, and CI/CD	Teaches controlled change	Use branches, PRs, reviews, tests, and CI/CD as engineering controls
18	Reviewing AI-Generated Systems	Makes generated work inspectable	Review AI-generated code, artifacts, workflows, and risks
19	Testing and Verification Fundamentals	Establishes test evidence	Connect requirements to tests, defects, and verification evidence
20	Testing Intelligent and AI-Assisted Systems	Extends testing to AI-era uncertainty	Validate nondeterministic, AI-assisted, or context-sensitive behavior
21	Release Readiness and Engineering Evidence	Turns completion into defensible release judgment	Build release evidence, known limitations, and risk disposition
22	Engineering Presentations and Release Defense	Teaches evidence-backed communication	Defend engineering claims with repository evidence

Evidence Learners Should Produce Learners should be able to produce evidence such as:

- Repository structure
- Requirements
- Assumptions and open questions
- Risk register
- Architecture overview

- ADRs
- Pull requests
- Review evidence
- AI-use log
- Test plan
- Test evidence
- Defect log
- Release notes
- Release readiness review
- Engineering presentation or release defense

Part III Learning Role: Operational Trust

Part III extends engineering beyond construction and release.

It teaches learners that engineering responsibility continues after a system appears to work.

Educational Function Part III introduces operational evidence, stabilization, observability, runbooks, security governance, reliability, incident response, release authority, and organizational trust.

Primary Learner Shift

Release Defender

↓

Operational Thinker

Typical Educational Use Part III may be taught deeply in graduate courses, professional training, operations-focused courses, capstones, and advanced software engineering sequences.

In undergraduate project courses, Part III is often used selectively during Cycle 2, release readiness, postmortem work, and final reflection.

Chapters

Ch.	Chapter Title	Educational Role	Learner Capability
23	Postmortems and Engineering Learning	Converts problems into learning	Conduct postmortems and preserve lessons
24	Defect Reduction and Stabilization	Focuses on maturity after defects	Reduce defect patterns and stabilize systems
25	Observability and Runtime Evidence	Connects systems to runtime reality	Use logs, metrics, traces, and runtime evidence
26	Operational Readiness and Runbooks	Prepares systems for operation	Create runbooks, support paths, and readiness evidence
27	Security Engineering and Governance	Treats security as lifecycle responsibility	Identify permissions, controls, risks, and governance obligations
28	AI Governance and Controlled Delegation	Defines AI authority boundaries	Bound, review, audit, and control AI delegation
29	Reliability Engineering and Failure Analysis	Teaches failure-oriented engineering	Analyze reliability, failure modes, and recovery assumptions
30	Operational Incident Response	Teaches response under stress	Manage incidents, evidence, roles, communication, and recovery

Ch.	Chapter Title	Educational Role	Learner Capability
31	Release Governance and Approval Authority	Connects release to authority	Define who can approve, defer, reject, or escalate release decisions
32	Trust, Transparency, and Organizational Confidence	Connects evidence to institutional trust	Communicate trustworthiness through transparency and accountability

Evidence Learners Should Produce Learners should be able to produce evidence such as:

- Postmortem
- Defect trend analysis
- Observability notes
- Runtime evidence
- Runbook
- Security governance checklist
- Delegation boundary analysis
- Reliability assumptions
- Incident response notes
- Release approval rationale
- Transparency or trust summary

Part IV Learning Role: Trustworthy Intelligent Systems

Part IV addresses the future-facing responsibilities of engineers working with intelligent systems.

It focuses on agentic systems, context engineering, human oversight, understandability, complexity, stewardship, and professional identity.

Educational Function Part IV helps learners understand how trustworthy engineering evolves as systems become more intelligent, distributed, automated, and organizationally consequential.

Primary Learner Shift

Operational Thinker

↓

Future Trustworthy Engineer

Typical Educational Use Part IV may be used deeply in AI governance, architecture, graduate, enterprise, or professional training contexts.

In undergraduate courses, Part IV often works best as capstone framing, professional reflection, or future-practice orientation.

Chapters

Ch.	Chapter Title	Educational Role	Learner Capability
33	Agentic Systems and Workflow Orchestration	Introduces systems that act	Understand agents, workflows, orchestration, and control
34	Enterprise AI Architecture and Context Engineering	Defines enterprise AI context architecture	Engineer authoritative context, systems of record, and control planes

Ch.	Chapter Title	Educational Role	Learner Capability
35	Human Oversight in Intelligent Systems	Deepens oversight as system design	Design human review, approval, intervention, and accountability
36	Understandability and Operational Transparency	Makes systems explainable enough to govern	Improve transparency, interpretability, and operational understanding
37	Complexity, Cognitive Load, and Understandability	Connects complexity to human limits	Reduce cognitive load and design for maintainable understanding
38	Engineering Stewardship in the AI Era	Defines long-term responsibility	Preserve trust through maintenance, governance, and organizational memory
39	The Future Trustworthy Engineer	Culminates in professional identity	Integrate ETIS responsibilities into future engineering practice

Evidence Learners Should Produce Learners should be able to produce evidence such as:

- Agent boundary analysis
- Workflow orchestration review
- Context engineering map
- Human oversight plan
- Understandability review
- Cognitive load analysis
- Stewardship reflection
- Future engineer professional identity statement

COMP330 Selective Use Model

COMP330 should not attempt to teach all 39 chapters equally.

Instead, COMP330 should use ETIS as the governing framework for a semester-long software engineering experience.

COMP330 primarily emphasizes Parts I and II while selectively introducing Parts III and IV.

This reflects the practical reality of a one-semester undergraduate software engineering course that must combine reading, project work, teamwork, repository practice, reviews, testing, demonstrations, and release defense.

Primary COMP330 Coverage

COMP330 should strongly emphasize:

1. Engineering Trustworthy Intelligent Systems
2. Why Software Projects Fail
3. Lifecycle Models and Engineering Under Uncertainty
4. AI Changes the Software Lifecycle
5. Engineering Judgment and Human Oversight
6. Teams, Communication, Review, and Accountability
7. Project Launch and Engineering Standards
8. Repository-Centered Engineering
9. Requirements, Stakeholders, and Engineering the Right Problem
10. AI-Assisted Requirements Engineering
11. Planning, Estimation, Risk, and Tradeoffs

12. Architecture Fundamentals
13. Intelligent Systems Architecture
14. Architecture Reviews and ADRs
15. AI-Assisted Design and Coding
16. Pull Requests, Reviews, and CI/CD
17. Reviewing AI-Generated Systems
18. Testing and Verification Fundamentals
19. Testing Intelligent and AI-Assisted Systems
20. Release Readiness and Engineering Evidence
21. Engineering Presentations and Release Defense

These chapters align most directly with a team-based undergraduate software engineering course.

Chapter 3 may also be introduced selectively when discussing complexity, team coordination, sociotechnical systems, or project failure.

Secondary COMP330 Coverage

COMP330 should selectively introduce:

23. Postmortems and Engineering Learning
24. Defect Reduction and Stabilization
25. Observability and Runtime Evidence
26. Operational Readiness and Runbooks
27. Security Engineering and Governance
28. AI Governance and Controlled Delegation
29. Reliability Engineering and Failure Analysis
30. Operational Incident Response
31. Release Governance and Approval Authority
32. Trust, Transparency, and Organizational Confidence

These chapters are especially useful for Cycle 2, final presentations, operational maturity, release readiness, and professional reflection.

Capstone COMP330 Coverage

COMP330 should use the final part as a capstone framing device rather than full-depth coverage.

33. Agentic Systems and Workflow Orchestration
34. Enterprise AI Architecture and Context Engineering
35. Human Oversight in Intelligent Systems
36. Understandability and Operational Transparency
37. Complexity, Cognitive Load, and Understandability
38. Engineering Stewardship in the AI Era
39. The Future Trustworthy Engineer

These chapters help students understand where the profession is going and why their project evidence, AI-use discipline, engineering reviews, and release defense matter.

Chapter-by-Chapter Learning Map

Ch.	Chapter Title	Learning Role	Core Competency	Common Educational Use
1	Engineering Trustworthy Intelligent Systems	Establishes the ETIS worldview	Understand trustworthiness as an engineering responsibility	Core
2	Why Software Projects Fail	Explains why discipline is necessary	Recognize failure patterns in complexity, coordination, quality, and governance	Core
3	Complexity, Coordination, and Sociotechnical Systems	Frames systems as people-process-technology environments	Analyze sociotechnical risk and coordination complexity	Supporting
4	Lifecycle Models and Engineering Under Uncertainty	Connects lifecycle choice to uncertainty and risk	Select lifecycle approaches based on consequence and uncertainty	Core
5	AI Changes the Software Lifecycle	Introduces AI-era lifecycle disruption	Understand how AI affects requirements, design, coding, testing, and operations	Core
6	Engineering Judgment and Human Oversight	Establishes accountability in AI-assisted work	Apply human judgment to generated or automated outputs	Core
7	Teams, Communication, Review, and Accountability	Builds team engineering discipline	Coordinate work through roles, communication, review, and accountability	Core
8	Project Launch and Engineering Standards	Begins controlled engineering work	Establish team standards, workflow, and engineering expectations	Core
9	Repository-Centered Engineering	Defines the repository as engineering memory	Use repositories as evidence systems, not just code storage	Core
10	Requirements, Stakeholders, and Engineering the Right Problem	Teaches requirements discipline	Translate stakeholder needs into reviewable requirements	Core
11	AI-Assisted Requirements Engineering	Shows responsible AI use in requirements	Use AI to critique and improve requirements without surrendering judgment	Core
12	Planning, Estimation, Risk, and Tradeoffs	Connects planning to reality	Make work, estimates, risks, and tradeoffs visible	Core
13	Architecture Fundamentals	Establishes structural design thinking	Define components, interfaces, responsibilities, and boundaries	Core
14	Intelligent Systems Architecture	Extends architecture into AI-enabled systems	Design context, tools, permissions, orchestration, and governance boundaries	Core
15	Architecture Reviews and ADRs	Makes decisions reviewable	Create and evaluate ADRs, review findings, and design tradeoffs	Core
16	AI-Assisted Design and Coding	Governs AI-assisted construction	Use AI for implementation while preserving ownership and verification	Core

Ch.	Chapter Title	Learning Role	Core Competency	Common Educational Use
17	Pull Requests, Reviews, and CI/CD	Teaches controlled change	Use branches, PRs, reviews, tests, and CI/CD as engineering controls	Core
18	Reviewing AI-Generated Systems	Makes generated work inspectable	Review AI-generated code, artifacts, workflows, and risks	Core
19	Testing and Verification Fundamentals	Establishes test evidence	Connect requirements to tests, defects, and verification evidence	Core
20	Testing Intelligent and AI-Assisted Systems	Extends testing to AI-era uncertainty	Validate nondeterministic, AI-assisted, or context-sensitive behavior	Core
21	Release Readiness and Engineering Evidence	Turns completion into defensible release judgment	Build release evidence, known limitations, and risk disposition	Core
22	Engineering Presentations and Release Defense	Teaches evidence-backed communication	Defend engineering claims with repository evidence	Core
23	Postmortems and Engineering Learning	Converts problems into learning	Conduct postmortems and preserve lessons	Supporting
24	Defect Reduction and Stabilization	Focuses on maturity after defects	Reduce defect patterns and stabilize systems	Selective
25	Observability and Runtime Evidence	Connects systems to runtime reality	Use logs, metrics, traces, and runtime evidence	Supporting
26	Operational Readiness and Runbooks	Prepares systems for operation	Create runbooks, support paths, and readiness evidence	Supporting
27	Security Engineering and Governance	Treats security as lifecycle responsibility	Identify permissions, controls, risks, and governance obligations	Supporting
28	AI Governance and Controlled Delegation	Defines AI authority boundaries	Bound, review, audit, and control AI delegation	Supporting
29	Reliability Engineering and Failure Analysis	Teaches failure-oriented engineering	Analyze reliability, failure modes, and recovery assumptions	Selective
30	Operational Incident Response	Teaches response under stress	Manage incidents, evidence, roles, communication, and recovery	Selective
31	Release Governance and Approval Authority	Connects release to authority	Define who can approve, defer, reject, or escalate release decisions	Selective
32	Trust, Transparency, and Organizational Confidence	Connects evidence to institutional trust	Communicate trustworthiness through transparency and accountability	Supporting
33	Agentic Systems and Workflow Orchestration	Introduces systems that act	Understand agents, workflows, orchestration, and control	Capstone
34	Enterprise AI Architecture and Context Engineering	Defines enterprise AI context architecture	Engineer authoritative context, systems of record, and control planes	Capstone

Ch.	Chapter Title	Learning Role	Core Competency	Common Educational Use
35	Human Oversight in Intelligent Systems	Deepens oversight as system design	Design human review, approval, intervention, and accountability	Capstone
36	Understandability and Operational Transparency	Makes systems explainable enough to govern	Improve transparency, interpretability, and operational understanding	Capstone
37	Complexity, Cognitive Load, and Understandability	Connects complexity to human limits	Reduce cognitive load and design for maintainable understanding	Capstone
38	Engineering Stewardship in the AI Era	Defines long-term responsibility	Preserve trust through maintenance, governance, and organizational memory	Capstone
39	The Future Trustworthy Engineer	Culminates in professional identity	Integrate ETIS responsibilities into future engineering practice	Capstone

Competency Mapping

ETIS chapters can also be grouped by competency.

These groupings help instructors create modules, assignments, classroom exercises, workshops, and professional learning paths.

Engineering Mindset 1, 2, 3, 4, 5, 6, 7

Learners develop the worldview needed to treat software as a sociotechnical, evidence-based, governed engineering discipline.

Repository and Evidence Practice 8, 9, 15, 17, 21, 22, 23, 26, 38

Learners learn how to preserve engineering memory, support review, defend release claims, and steward knowledge over time.

Requirements, Planning, and Tradeoffs 10, 11, 12

Learners learn how to turn vague needs into bounded, reviewable, and manageable engineering commitments.

Architecture and Design 13, 14, 15, 34, 36, 37

Learners learn how to design boundaries, decisions, context, understandability, and governance structures.

AI-Assisted Engineering 5, 6, 11, 14, 16, 18, 20, 28, 33, 34, 35, 36

Learners learn how to use AI responsibly while preserving human accountability, verification, and control.

Testing, Verification, and Release 18, 19, 20, 21, 22, 31

Learners learn how to defend quality claims with evidence rather than confidence.

Operations and Runtime Trust 23, 24, 25, 26, 29, 30, 32

Learners learn how systems behave after release and how organizations observe, recover, learn, and improve.

Governance, Security, and Authority 6, 14, 18, 27, 28, 31, 33, 35, 36

Learners learn how authority, approval, security, auditability, oversight, and escalation must be engineered.

Stewardship and Professional Identity 32, 35, 36, 37, 38, 39

Learners learn how trustworthy engineering becomes a durable professional identity.

Recommended Chapter Bundles

Chapter bundles support selective adoption.

They should be treated as starting points, not rigid prescriptions.

Essential Undergraduate Bundle 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 21, 22, 23, 25, 26, 27, 32, 39

Use when the course has limited time but needs broad ETIS exposure.

This bundle prioritizes professional formation, project discipline, repository-centered engineering, testing, release defense, operational awareness, and future professional identity.

COMP330 Project Bundle 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 28, 34, 39

Use for a team-based software engineering course with AI-era emphasis.

This bundle supports requirements, architecture, AI-assisted engineering, review, testing, release defense, and selective operational trust.

AI Governance Bundle 5, 6, 11, 14, 16, 18, 20, 27, 28, 31, 33, 34, 35, 36, 39

Use for AI governance, risk, oversight, or review-board training.

This bundle emphasizes human accountability, AI authority boundaries, verification, oversight, context, transparency, and future professional responsibility.

Operations and Reliability Bundle 21, 23, 24, 25, 26, 27, 29, 30, 31, 32, 38

Use for DevOps, SRE, operational readiness, incident response, and reliability training.

This bundle emphasizes release evidence, runtime behavior, failure analysis, incident response, authority, trust, and stewardship.

Architecture and Stewardship Bundle 3, 6, 9, 12, 13, 14, 15, 21, 26, 28, 34, 35, 36, 37, 38, 39

Use for architects, technical leads, graduate students, and organizational adoption.

This bundle emphasizes complexity, judgment, repositories, planning, architecture, AI delegation, context, oversight, understandability, cognitive load, stewardship, and professional identity.

Using This Map

This map should be used when designing:

- Course syllabi
- Weekly schedules
- Reading assignments
- Project checkpoints
- Classroom activities
- Review-board exercises
- Rubrics
- Professional workshops
- Student starter kits
- Instructor packages
- Adoption examples

The map should not be treated as a rigid requirement.

Its purpose is alignment.

A course may use fewer chapters.

A workshop may use only one chapter cluster.

A professional team may use a single bundle.

That is acceptable.

What matters is that each educational use of ETIS remains connected to the larger framework of evidence, governance, operations, accountability, and stewardship.

Guiding Rule

Every selective adoption of ETIS should be able to answer:

What part of the trustworthy engineering journey are we teaching, and what evidence will learners produce to prove they practiced it?

If the answer is clear, the adoption is likely aligned with ETIS.

If the answer is unclear, the educational design should be revised.

Core Thesis

The ETIS book is one complete framework.

There are many legitimate ways to teach it.

A responsible learning map does not weaken the framework by allowing selective use.

It strengthens the framework by helping each audience enter the trustworthy engineering journey at the right depth, with the right emphasis, and with clear expectations for evidence, judgment, and accountability.

Part II

Learning and Transformation Models

ETIS Learning Models

Purpose

This document defines the learning model architecture for the ETIS (Engineering Trustworthy Intelligent Systems) Educational Ecosystem.

Learning models explain how learners develop from students completing academic work into engineers capable of responsible professional practice.

The purpose of this document is to connect the individual learning models in this directory into one coherent developmental system.

ETIS education is not only about teaching concepts.

ETIS education is about transforming how learners think, practice, review, verify, communicate, and assume responsibility.

Relationship to Educational Architecture

The `educational_architecture` directory defines how ETIS is taught at the ecosystem level.

The `learning_models` directory defines how learners mature inside that educational system.

Relationship model:

ETIS Book

↓

Educational Architecture

↓

Learning Models

↓

Shared Assets

↓

Student Starter Kits

↓

Educational Implementations

↓

Learner Practice

↓

Professional Transformation

The ETIS book defines the framework.

Educational architecture defines the teaching system.

Learning models define the developmental arc.

Shared assets provide reusable educational building blocks.

Starter kits provide structured practice environments.

Adoption examples show implementation-specific use.

Core Learning Thesis

ETIS education should change learner behavior.

A successful learner should not merely know more at the end of an ETIS learning experience.

A successful learner should practice differently.

They should:

- Define intent more clearly
- Preserve context more deliberately
- Bound authority more carefully
- Verify behavior more rigorously
- Explain decisions more professionally
- Operate systems more realistically
- Own outcomes more fully

The learning models exist to make that transformation visible.

Learning Model System

The ETIS learning model system currently contains four major model areas.

learning_models/

|— professional_transformation_model.md

|— engineering_maturity_model.md

|— software_engineering_learning_progression.md

‘— two-cycle_engineering_model/

Each model serves a distinct purpose.

Model Responsibilities

Professional Transformation Model The Professional Transformation Model defines the identity progression ETIS is designed to produce.

It answers:

Who is the learner becoming?

The current progression is:

Student

↓

Responsible Engineer

↓

Reviewer

↓

Architect

↓

Release Defender

↓

Operational Thinker

↓

Future Trustworthy Engineer

This is the central developmental arc of ETIS education.

Engineering Maturity Model The Engineering Maturity Model defines observable growth in engineering behavior.

It answers:

How does engineering behavior improve over time?

This model should describe maturity across dimensions such as:

- Intent definition
- Context quality
- Evidence creation
- Repository discipline
- Review participation
- AI responsibility
- Testing and verification
- Release readiness
- Operational thinking
- Accountability

The maturity model helps instructors, reviewers, and learners recognize growth.

Software Engineering Learning Progression The Software Engineering Learning Progression defines how students move from concept awareness to applied engineering practice.

It answers:

How does a learner move from knowing concepts to practicing trustworthy engineering?

This model should connect educational activities to progressive engineering capability.

It should help explain how readings, assignments, repository work, reviews, templates, workflows, and reflections become a coherent learning journey.

Two-Cycle Engineering Model The Two-Cycle Engineering Model defines a project-based learning progression.

It answers:

How does project work mature across time?

The model has two major cycles:

Cycle 1: Can it work?

Cycle 2: Can it survive?

Cycle 1 emphasizes feasibility, basic functionality, team formation, repository setup, requirements, architecture, and initial implementation.

Cycle 2 emphasizes quality, traceability, testing, reviewability, operational readiness, release evidence, known limitations, risk, and sustainability.

This model is especially important for COMP330 and other project-centered implementations.

How the Models Work Together

The models are complementary.

They should not be treated as competing frameworks.

The Professional Transformation Model defines the learner identity arc.

The Engineering Maturity Model defines observable behavior.

The Software Engineering Learning Progression defines educational sequencing.

The Two-Cycle Engineering Model defines project-phase progression.

Together, they form a layered learning architecture.

Professional Transformation Model

↓

Engineering Maturity Model

↓

Software Engineering Learning Progression

↓

Two-Cycle Engineering Model

↓

Assignments, Reviews, Artifacts, and Practice

This relationship helps preserve coherence between high-level educational goals and day-to-day student work.

Developmental Dimensions

ETIS learning models should reinforce several developmental dimensions.

Intent Learners should improve in their ability to define what a system is supposed to accomplish and why it matters.

Context Learners should improve in their ability to preserve relevant information for teammates, reviewers, AI tools, instructors, and future maintainers.

Authority Learners should improve in their ability to define who or what is allowed to act, decide, approve, modify, or release.

Evidence Learners should improve in their ability to support claims with artifacts, tests, reviews, logs, decisions, and traceability.

Review Learners should improve in their ability to review work and be reviewed by others.

AI Responsibility Learners should improve in their ability to use AI without surrendering judgment, understanding, verification, or accountability.

Operations Learners should improve in their ability to think beyond implementation toward runtime behavior, support, failure, recovery, and improvement.

Stewardship Learners should improve in their ability to preserve trust over time.

Relationship to Shared Assets

Shared assets provide the educational materials through which learning models become practice.

For example:

- Playbooks help learners develop professional behaviors.
- Templates help learners create engineering evidence.
- Workflows help learners understand engineering flow.
- AI Engineering assets help learners practice responsible AI collaboration.
- Assessment assets help evaluate engineering maturity.

Learning models explain why those assets matter.

Shared assets provide the mechanisms that make learner growth observable.

Relationship to Student Starter Kits

Student Starter Kits provide structured environments where learners practice ETIS.

Learning models explain what growth should occur inside those environments.

A starter kit should not merely organize files.

It should support progressive engineering maturity.

For example, a starter kit should help learners move from:

empty repository

to:

repository as engineering memory system

and from:

working code

to:

reviewable, testable, explainable, and defensible engineering evidence

The learning models define the intended developmental meaning of that movement.

Relationship to Educational Implementations

Educational implementations consume learning models.

The current flagship implementation is:

adoption_examples/ ‘— loyola_comp330/

COMP330 uses learning models to support:

- Team-based project work

- Cycle 1 and Cycle 2 expectations
- Repository-centered engineering
- Review practices
- Release readiness
- AI-use accountability
- Professional communication
- Engineering reflection

Future implementations may adapt sequencing and emphasis while preserving the professional transformation architecture.

Relationship to Assessment

Learning models should inform assessment.

Assessment should not merely evaluate whether students completed assigned tasks.

Assessment should evaluate whether learners are maturing as engineers.

Evidence of maturity may include:

- Better requirements
- Clearer architecture
- Stronger traceability
- More meaningful reviews
- Improved AI-use disclosure
- Better test evidence
- More defensible release claims
- More realistic operational thinking
- Greater accountability

The learning models give assessment a developmental foundation.

Relationship to Reflection

Reflection is important because professional transformation requires self-awareness.

Students should be encouraged to ask:

- How has my understanding of engineering changed?
- What evidence shows I improved?
- Where did I rely too heavily on assumptions?
- How did I verify AI-assisted work?
- How did I contribute to team trust?
- What would I do differently in Cycle 2?
- What does release readiness mean now compared to the beginning of the course?

Reflection should not be treated as sentiment.

Reflection should help learners recognize their own professional development.

Instructor Use

Instructors should use the learning models to:

- Sequence learning experiences
- Explain why assignments exist
- Design project checkpoints
- Provide developmental feedback

- Evaluate engineering maturity
- Help students understand professional growth
- Connect course activities to ETIS principles

The models should make the course feel like a progression rather than a sequence of disconnected deliverables.

Student Use

Students should use the learning models to understand what they are becoming.

They should understand that the goal is not simply to finish assignments.

The goal is to develop engineering responsibility.

Students should use these models to interpret feedback, prepare for reviews, improve team practices, and reflect on their growth.

Future Evolution

The ETIS learning model system may expand over time.

Potential future models include:

- Instructor development model
- Reviewer maturity model
- AI engineering maturity model
- Release defense maturity model
- Engineering portfolio maturity model
- Organizational learning model
- Capstone maturity pathway

Future models should only be added when they clarify learner development in a way not already covered by the existing models.

Core Thesis

ETIS learning models exist because trustworthy engineering is learned through progressive practice.

A learner does not become a trustworthy engineer by reading principles alone.

A learner becomes a trustworthy engineer by repeatedly practicing intent definition, context engineering, authority bounding, verification, review, operational thinking, and accountability.

The learning models make that progression visible, teachable, assessable, and repeatable.

Professional Transformation Model

Purpose

This document defines the ETIS (Engineering Trustworthy Intelligent Systems) Professional Transformation Model.

The purpose of this model is to describe the identity progression ETIS education is designed to produce.

ETIS is not solely a knowledge-transfer framework.

It is a professional development framework.

Students should progressively transform from learners completing assignments into engineers capable of designing, reviewing, governing, operating, and stewarding trustworthy intelligent systems.

This model makes that progression explicit.

Core Philosophy

Software engineering education should not end with technical competence.

Technical competence is necessary but insufficient.

Future engineers will increasingly be asked to:

- Define intent
- Engineer context
- Bound authority
- Verify behavior
- Explain decisions
- Operate reality
- Own outcomes

The Professional Transformation Model exists to teach those responsibilities intentionally.

The Transformation Path

The ETIS Educational Ecosystem is organized around the following progression.

Student

↓

Responsible Engineer

↓

Reviewer

↓

Architect

↓

Release Defender

↓

Operational Thinker

↓

This progression is developmental.

It is not a hierarchy.

It is not a career ladder.

It is not a set of job titles.

Each stage expands engineering perspective and responsibility.

The stages are cumulative.

Learners do not abandon previous stages as they advance.

They build upon them.

Stage 1: Student

Primary Identity Learner.

Primary Responsibility Acquire foundational engineering habits.

Common Perspective Students often initially focus on:

My assignment

My code

My grade

This perspective is natural at the beginning of the learning journey.

ETIS intentionally expands this perspective.

Educational Focus Students begin learning:

- Team participation
- Repository organization
- Engineering terminology
- AI responsibility
- Engineering communication
- Review participation

Guiding Question What am I being asked to build?

Stage 2: Responsible Engineer

Primary Identity Engineer responsible for outcomes.

Primary Responsibility Own engineering decisions.

Perspective Shift The learner moves from:

I completed my task.

to:

I understand why the work matters.

Educational Focus Learners develop:

- Intent definition
- Requirements discipline
- Repository-centered engineering
- Basic risk awareness
- Evidence creation
- Team accountability

Guiding Question What problem are we solving, and how do we know?

Stage 3: Reviewer

Primary Identity Engineer capable of evaluating engineering work.

Primary Responsibility Improve engineering quality through review.

Perspective Shift The learner moves from:

I produced this.

to:

Can someone else understand, verify, and improve this?

Educational Focus Learners develop:

- Review skills
- Traceability thinking
- AI verification
- Assumption identification
- Risk awareness
- Constructive feedback

Guiding Question How do we know this is trustworthy?

Stage 4: Architect

Primary Identity Engineer who thinks structurally.

Primary Responsibility Design understandable systems.

Perspective Shift The learner moves from:

How do I build this?

to:

How should this system be organized?

Educational Focus Learners develop:

- Boundary thinking
- Tradeoff analysis
- Context engineering

- System decomposition
- Interface design
- Governance awareness

Guiding Question How should this system be structured so others can understand and evolve it?

Stage 5: Release Defender

Primary Identity Engineer capable of defending engineering claims.

Primary Responsibility Determine whether engineering work is ready.

Perspective Shift The learner moves from:

It works.

to:

Can we responsibly stand behind this?

Educational Focus Learners develop:

- Release readiness
- Evidence gathering
- Risk communication
- Testing discipline
- Defect awareness
- Engineering presentations

Guiding Question What evidence supports our claims?

Stage 6: Operational Thinker

Primary Identity Engineer who thinks beyond implementation.

Primary Responsibility Prepare systems for operational reality.

Perspective Shift The learner moves from:

The project is finished.

to:

What happens when this system is used, changed, or fails?

Educational Focus Learners develop:

- Operational awareness
- Incident thinking
- Reliability thinking
- Observability awareness
- Runbook thinking
- Continuous improvement

Guiding Question Can this system survive operation?

Stage 7: Future Trustworthy Engineer

Primary Identity Engineer who stewards intelligent systems over time.

Primary Responsibility Preserve trust.

Perspective Shift The learner moves from:

How do we build systems?

to:

How do we sustain trustworthy systems in an AI-shaped future?

Educational Focus Learners develop:

- Stewardship
- Human oversight
- AI governance
- Explainability
- Organizational trust
- Long-term accountability

Guiding Question How do we preserve trust over time?

Transformation Characteristics

The progression expands responsibility across multiple dimensions.

Dimension	Early Stage	Mature Stage
Intent	Receive assignments	Define problems
Context	Consume context	Engineer context
Authority	Follow instructions	Bound authority
Evidence	Produce artifacts	Defend claims
Review	Receive feedback	Conduct reviews
AI Usage	Use AI	Govern AI
Operations	Build systems	Sustain systems
Accountability	Complete tasks	Own outcomes

These dimensions should be reinforced throughout ETIS implementations.

Relationship to the ETIS Book

The ETIS book supports this transformation.

Each part contributes to the progression.

Book Part	Primary Transformation
Part I	Student → Responsible Engineer
Part II	Responsible Engineer → Reviewer → Release Defender
Part III	Release Defender → Operational Thinker
Part IV	Operational Thinker → Future Trustworthy Engineer

This relationship is approximate rather than rigid.

Transformation is cumulative.

Relationship to Educational Implementations

Educational implementations should reinforce this progression.

The current flagship implementation is:

adoption_examples/ ‘— loyola_comp330/

COMP330 primarily focuses on:

Student

↓

Responsible Engineer

↓

Reviewer

↓

Release Defender

while selectively exposing learners to:

Operational Thinker

Future Trustworthy Engineer

This reflects the practical constraints of a one-semester undergraduate software engineering course.

Relationship to the Engineering Maturity Model

This document defines **who learners become**.

The Engineering Maturity Model defines **how engineering behavior improves**.

The two models are complementary.

Relationship to Assessment

Assessment should reinforce transformation.

Students should increasingly demonstrate:

- Better judgment
- Better evidence
- Better reviews
- Better communication
- Better AI usage
- Better operational thinking
- Better accountability

The objective is not merely higher scores.

The objective is visible professional growth.

Instructor Use

Instructors should use this model to explain why assignments exist.

Assignments should never feel isolated.

Students should understand:

This activity exists because it is helping you become a reviewer.

This activity exists because it is helping you become a release defender.

This activity exists because it is helping you become an operational thinker.

The model helps connect day-to-day activities to long-term professional identity.

Student Use

Students should use this model to understand their own development.

At various points they should ask:

- What engineering responsibilities do I now understand better?
- How has my engineering perspective changed?
- What evidence demonstrates my growth?
- Which stage am I currently strengthening?

The model should make professional growth visible.

Future Evolution

The Professional Transformation Model is intended to remain stable.

Future refinements may expand the descriptions, examples, or assessment mechanisms associated with each stage.

However, changes should be made cautiously because this model serves as one of the foundational pillars of the ETIS Educational Ecosystem.

Core Thesis

The goal of ETIS education is not to create students who can produce software.

The goal is to develop engineers who can define intent, engineer context, bound authority, verify behavior, operate reality, explain decisions, and own outcomes.

The Professional Transformation Model exists to make that evolution intentional.

Engineering Maturity Model

Purpose

This document defines the ETIS (Engineering Trustworthy Intelligent Systems) Engineering Maturity Model.

The purpose of this model is to describe how engineering behavior improves over time.

While the Professional Transformation Model explains who the learner is becoming, the Engineering Maturity Model explains how that growth becomes visible in practice.

This model helps instructors, learners, reviewers, and adopters recognize the difference between immature task completion and mature engineering responsibility.

Core Philosophy

Engineering maturity is not measured by confidence, activity, volume, or tool usage.

Engineering maturity is demonstrated through behavior.

A mature engineer defines intent clearly, preserves context, creates evidence, reviews work, verifies claims, communicates risks, uses AI responsibly, prepares for operation, and owns outcomes.

The purpose of this model is to make those behaviors observable.

Relationship to Professional Transformation

The Professional Transformation Model answers:

Who is the learner becoming?

The Engineering Maturity Model answers:

How does mature engineering behavior appear?

The models are complementary.

A learner may begin as a student, but their maturity grows as they demonstrate more disciplined engineering behavior across specific dimensions.

Maturity Levels

The ETIS Engineering Maturity Model uses five maturity levels.

Level 1: Task Completion

Level 2: Structured Participation

Level 3: Evidence-Based Engineering

Level 4: Reviewable Engineering Judgment

Level 5: Stewardship-Oriented Engineering

These levels are developmental.

They are not grades.

They are not labels to attach permanently to a learner.

They describe observable patterns of engineering behavior.

Level 1: Task Completion

Description At Level 1, the learner focuses primarily on completing assigned work.

The learner may produce code, documents, or project artifacts, but the work is often disconnected from broader engineering context.

Common Behaviors The learner may:

- Complete assigned tasks
- Produce required files
- Submit work on time
- Follow explicit instructions
- Use tools when directed

Limitations The learner may not yet:

- Understand why artifacts matter
- Connect work to requirements or risks
- Preserve context for others
- Explain decisions clearly
- Verify AI-assisted work deeply
- Think beyond immediate completion

Typical Mindset I did what was assigned.

Development Need The learner needs to move from task completion toward structured participation in an engineering system.

Level 2: Structured Participation

Description At Level 2, the learner begins participating in engineering practices with more consistency.

The learner understands that engineering work requires structure, coordination, communication, and artifacts.

Common Behaviors The learner may:

- Use the repository more consistently
- Participate in team communication
- Follow agreed workflows
- Create basic engineering artifacts
- Attend to requirements, risks, and roles
- Begin documenting AI usage
- Participate in reviews

Limitations The learner may still:

- Treat artifacts as forms rather than evidence
- Rely heavily on templates without understanding
- Produce shallow explanations
- Verify work inconsistently
- Participate in review passively

- Struggle to connect artifacts across the lifecycle

Typical Mindset I am following the engineering process.

Development Need The learner needs to move from following process toward creating evidence that supports engineering claims.

Level 3: Evidence-Based Engineering

Description At Level 3, the learner begins using artifacts, repositories, tests, reviews, and documentation as engineering evidence.

The learner understands that claims require support.

Common Behaviors The learner may:

- Connect requirements to design and testing
- Use the repository as an engineering memory system
- Preserve assumptions and decisions
- Create useful test evidence
- Document risks and limitations
- Explain AI-assisted contributions
- Prepare evidence for reviews or presentations

Limitations The learner may still:

- Need help evaluating evidence quality
- Miss deeper tradeoffs
- Treat review as confirmation rather than improvement
- Underestimate operational consequences
- Struggle to defend decisions under questioning

Typical Mindset We need evidence to support our claims.

Development Need The learner needs to move from evidence creation toward reviewable engineering judgment.

Level 4: Reviewable Engineering Judgment

Description At Level 4, the learner demonstrates the ability to make, explain, and defend engineering decisions.

The learner can evaluate alternatives, recognize risks, participate meaningfully in review, and revise work based on evidence.

Common Behaviors The learner may:

- Explain tradeoffs clearly
- Defend architectural and implementation choices
- Conduct meaningful reviews
- Challenge assumptions constructively
- Evaluate AI-generated outputs critically
- Connect release readiness to evidence
- Identify operational risks

- Communicate limitations honestly

Limitations The learner may still:

- Need more experience with real operational consequences
- Require support in complex governance situations
- Have limited exposure to long-term maintenance and stewardship
- Struggle with ambiguity under organizational pressure

Typical Mindset Can this decision be reviewed, defended, and improved?

Development Need The learner needs to move from reviewable judgment toward long-term stewardship.

Level 5: Stewardship-Oriented Engineering

Description At Level 5, the learner thinks beyond immediate delivery.

The learner understands that engineering work must remain trustworthy over time.

This level emphasizes operational reality, governance, transparency, human oversight, organizational memory, and sustained accountability.

Common Behaviors The learner may:

- Think in terms of lifecycle consequences
- Design for maintainability and understandability
- Preserve organizational learning
- Anticipate failure and recovery
- Bound AI and automation authority
- Support governance and oversight
- Communicate trustworthiness to stakeholders
- Own outcomes beyond initial delivery

Limitations Level 5 is aspirational in most undergraduate settings.

Students may be introduced to this level without fully mastering it.

Typical Mindset How do we preserve trust over time?

Development Need The learner continues developing professional depth through experience, reflection, and responsibility.

Maturity Dimensions

Engineering maturity should be observed across multiple dimensions.

The following dimensions help instructors and reviewers evaluate growth.

Dimension 1: Intent Definition

Immature Pattern The learner accepts tasks without clarifying purpose.

Mature Pattern The learner clarifies goals, stakeholders, constraints, and success criteria.

Evidence Possible evidence includes:

- Requirements
- Acceptance criteria
- Assumptions and open questions
- Stakeholder summaries
- Scope statements

Dimension 2: Context Engineering

Immature Pattern The learner assumes others already understand the work.

Mature Pattern The learner preserves context so teammates, reviewers, AI systems, instructors, and future maintainers can understand the work.

Evidence Possible evidence includes:

- README files
- Architecture notes
- Decision records
- Traceability documents
- Planning artifacts
- Review summaries

Dimension 3: Repository Discipline

Immature Pattern The repository is treated as a file dump or code container.

Mature Pattern The repository becomes an engineering memory system.

Evidence Possible evidence includes:

- Organized repository structure
- Meaningful commits
- Pull requests
- Documentation
- Test evidence
- Release evidence
- Operational notes

Dimension 4: Evidence Creation

Immature Pattern The learner makes claims without support.

Mature Pattern The learner supports claims with artifacts, tests, reviews, decisions, and traceability.

Evidence Possible evidence includes:

- Test plans
- Test results
- Traceability summaries
- Defect logs
- Release readiness reviews
- Known limitations

- AI verification notes

Dimension 5: Review Participation

Immature Pattern The learner treats review as approval or criticism.

Mature Pattern The learner treats review as engineering improvement.

Evidence Possible evidence includes:

- Pull request reviews
- Architecture review notes
- Requirements review findings
- Code review comments
- Release readiness feedback
- Updated artifacts after review

Dimension 6: AI Responsibility

Immature Pattern The learner uses AI output without sufficient understanding or verification.

Mature Pattern The learner uses AI as a bounded collaborator and remains accountable for results.

Evidence Possible evidence includes:

- AI-use logs
- AI verification notes
- Disclosures
- Human review comments
- Test evidence for AI-assisted work
- Explanation of accepted and rejected AI suggestions

Dimension 7: Testing and Verification

Immature Pattern The learner treats testing as a final task or demonstration support.

Mature Pattern The learner designs verification evidence throughout the lifecycle.

Evidence Possible evidence includes:

- Test strategy
- Test cases
- CI evidence
- Defect logs
- Regression notes
- Requirement-to-test traceability

Dimension 8: Release Readiness

Immature Pattern The learner equates working software with readiness.

Mature Pattern The learner distinguishes demo readiness from release readiness.

Evidence Possible evidence includes:

- Release notes
- Known limitations
- Risk disposition
- Traceability summary
- Release readiness review
- Engineering defense presentation

Dimension 9: Operational Thinking

Immature Pattern The learner assumes engineering ends at submission or demo.

Mature Pattern The learner considers runtime behavior, failure, support, monitoring, recovery, and improvement.

Evidence Possible evidence includes:

- Runbooks
- Observability notes
- Incident response notes
- Postmortems
- Operational readiness reviews
- Support assumptions

Dimension 10: Accountability

Immature Pattern The learner views responsibility as assigned work completion.

Mature Pattern The learner owns decisions, evidence, limitations, risks, and outcomes.

Evidence Possible evidence includes:

- Decision explanations
- Risk communication
- Review responses
- Reflection artifacts
- Release defense
- Team responsibility records

Relationship to Assessment

The Engineering Maturity Model should inform assessment.

Assessment should evaluate whether learners are becoming more mature engineers, not merely whether they completed assigned artifacts.

A learner who produces many files but cannot explain their decisions has not demonstrated strong maturity.

A learner who uses AI extensively but cannot verify the result has not demonstrated strong maturity.

A learner who completes a demo but cannot explain release risks has not demonstrated strong maturity.

Assessment should reward visible growth in engineering responsibility.

Relationship to Instructor Analysis

Instructor analysis tools may help evaluate aspects of engineering maturity.

For example, repository analysis scripts may inspect:

- Repository structure
- Commit activity
- Pull request usage
- Documentation completeness
- Test evidence
- Traceability
- AI-use records
- Release readiness artifacts

However, automated analysis should not replace human judgment.

Analysis tools can surface evidence.

Instructors and reviewers must interpret that evidence.

The maturity model defines what the evidence means.

Relationship to Student Reflection

Students should use this model to understand their own growth.

Reflection prompts may include:

- Where did I move beyond task completion?
- What evidence did I create that supports engineering claims?
- How did I use review to improve the work?
- How did I verify AI-assisted contributions?
- What operational risks did I identify?
- What would I improve if this system continued beyond the course?

Reflection should connect student experience to maturity growth.

Relationship to COMP330

In COMP330, most students will begin between Level 1 and Level 2.

The course should help students move toward Level 3 and introduce Level 4 behaviors.

Cycle 1 primarily moves students from task completion toward structured participation.

Cycle 2 primarily moves students from structured participation toward evidence-based engineering and release defense.

Some students may demonstrate elements of Level 4.

Level 5 is introduced as a professional horizon rather than expected full mastery.

Use Guidance

This model should be used to guide:

- Assignment design
- Rubric development
- Repository review
- Instructor feedback

- Student reflection
- Project checkpoints
- Release readiness reviews
- Professional portfolio development

The model should not be used mechanically.

It should support judgment.

Core Thesis

Engineering maturity is visible in behavior.

Students mature when they move from task completion toward evidence, review, judgment, operational thinking, and stewardship.

The ETIS Engineering Maturity Model exists to make that growth visible, teachable, assessable, and improvable.

Software Engineering Learning Progression

Purpose

This document defines the ETIS (Engineering Trustworthy Intelligent Systems) Software Engineering Learning Progression.

The purpose of this model is to explain how learners move from understanding software engineering concepts to practicing trustworthy engineering through structured educational experiences.

This progression connects readings, assignments, repositories, reviews, artifacts, AI use, testing, release readiness, operations, and reflection into a coherent learning journey.

Core Philosophy

Software engineering is learned through progressive practice.

Students do not become engineers simply by reading about engineering.

They become engineers by repeatedly practicing the behaviors of engineering:

- Defining intent
- Creating evidence
- Communicating decisions
- Reviewing work
- Managing uncertainty
- Verifying behavior
- Using AI responsibly
- Preparing for release
- Thinking operationally
- Owning outcomes

The learning progression explains how those behaviors develop across time.

Relationship to Other Learning Models

This document answers:

How does a learner move from knowing concepts to practicing trustworthy engineering?

It complements the other ETIS learning models.

The Professional Transformation Model explains who the learner is becoming.

The Engineering Maturity Model explains how mature engineering behavior appears.

The Two-Cycle Engineering Model explains how project work matures across a semester or implementation cycle.

This document connects those models to the educational journey.

Learning Progression Overview

The ETIS Software Engineering Learning Progression contains seven stages.

1. Orientation
2. Intent Formation
3. Engineering Structure
4. Construction and Review

5. Verification and Release Defense
6. Operational Awareness
7. Professional Reflection and Stewardship

These stages are developmental.

They may align with weeks, modules, project checkpoints, or course phases, but they are not strictly tied to calendar time.

Different educational implementations may compress, expand, repeat, or reorder parts of the progression.

Stage 1: Orientation

Learning Focus Learners begin by understanding what software engineering means in the ETIS framework.

They encounter the idea that software engineering is not merely coding.

It is disciplined, evidence-centered, AI-aware, reviewable, operationally conscious, and accountable work.

Learner Questions Students begin asking:

What does software engineering require beyond implementation?

Why do software projects fail?

What responsibilities do engineers carry?

How does AI change engineering work?

Educational Activities Possible activities include:

- Reading foundational ETIS chapters
- Discussing software failure patterns
- Reviewing examples of poor engineering evidence
- Establishing team expectations
- Introducing AI-use expectations
- Introducing repository-centered engineering

Expected Evidence Learners may produce:

- Initial engineering reflection
- Team working agreements
- AI-use expectations
- Repository setup evidence
- Communication expectations

Developmental Shift

Student completing assignments

↓

Student entering an engineering system

Stage 2: Intent Formation

Learning Focus Learners begin defining what the system is supposed to accomplish and why.

They learn that unclear intent creates downstream engineering failure.

Intent formation includes stakeholders, goals, constraints, assumptions, requirements, risks, and success criteria.

Learner Questions Students begin asking:

What problem are we solving?

Who is affected?

What does success mean?

What assumptions are we making?

What risks are already visible?

Educational Activities Possible activities include:

- Stakeholder analysis
- Requirements drafting
- AI-assisted requirements critique
- Assumption identification
- Scope definition
- Risk register creation
- Initial planning and estimation

Expected Evidence Learners may produce:

- Requirements document
- Acceptance criteria
- Assumptions and open questions
- Scope statement
- Risk register
- Initial estimates
- Traceability notes

Developmental Shift

Building what seems requested

↓

Defining what should responsibly be built

Stage 3: Engineering Structure

Learning Focus Learners organize the system and the work.

They learn that engineering requires structure before and during construction.

Structure includes architecture, responsibilities, boundaries, repository organization, roles, workflows, and decision records.

Learner Questions Students begin asking:

How should the system be organized?

What are the major components?

What decisions need to be recorded?

How will the team coordinate work?

How will future reviewers understand our design?

Educational Activities Possible activities include:

- Architecture overview creation
- Component responsibility definition
- ADR creation
- Repository structure review
- Team role clarification
- Workflow setup
- Pull request expectation review

Expected Evidence Learners may produce:

- Architecture overview
- Component responsibility notes
- ADRs
- Repository structure
- Role matrix
- Workflow documentation
- Planning updates

Developmental Shift

Writing code

↓

Engineering a system others can understand

Stage 4: Construction and Review

Learning Focus Learners implement while preserving accountability, reviewability, and evidence.

They learn that implementation is not private activity.

Engineering construction should move through visible, reviewable, and controlled workflows.

AI may assist, but students remain responsible for understanding and verification.

Learner Questions Students begin asking:

How do we make changes safely?

How do we review work?

How do we verify AI-assisted output?

How do we connect implementation to requirements?

How do we preserve context for teammates and reviewers?

Educational Activities Possible activities include:

- Branch-based development
- Pull request creation
- Code review
- AI-assisted implementation with disclosure
- CI/CD use
- Defect tracking
- Review response and revision
- Traceability updates

Expected Evidence Learners may produce:

- Pull requests
- Review comments
- Commit history
- AI-use logs
- AI verification notes
- Defect log
- CI evidence
- Updated traceability

Developmental Shift

Making changes

↓

Making reviewable engineering changes

Stage 5: Verification and Release Defense

Learning Focus Learners learn that working software is not enough.

They must verify behavior, preserve evidence, identify limitations, communicate risk, and defend readiness.

This stage teaches the difference between demo readiness and release readiness.

Learner Questions Students begin asking:

What evidence supports our claims?

What has been tested?

What remains uncertain?

What are the known limitations?

Can we responsibly stand behind this release?

Educational Activities Possible activities include:

- Test planning
- Test execution

- Defect review
- Requirements-to-test traceability
- Release notes creation
- Known limitations documentation
- Release readiness review
- Engineering presentation or release defense

Expected Evidence Learners may produce:

- Test plan
- Test cases
- Test evidence
- Defect review
- Traceability summary
- Release notes
- Known limitations
- Release readiness review
- Engineering defense presentation

Developmental Shift

It works

↓

We can defend what we claim

Stage 6: Operational Awareness

Learning Focus Learners extend thinking beyond completion and demonstration.

They begin considering what happens when a system is used, changed, monitored, supported, or fails.

This stage introduces operational thinking without requiring full-scale production operations.

Learner Questions Students begin asking:

What happens after release?

How would someone operate this?

How would failures be noticed?

What should be done if something goes wrong?

What would future maintainers need to know?

Educational Activities Possible activities include:

- Runbook drafting
- Observability planning
- Incident response scenario review
- Postmortem exercise
- Security governance review
- Reliability assumption review
- Operational readiness discussion

Expected Evidence Learners may produce:

- Runbook
- Observability notes
- Runtime evidence plan
- Incident response notes
- Postmortem
- Security checklist
- Reliability assumptions
- Operational readiness summary

Developmental Shift

The project is done

↓

The system must survive use, change, and failure

Stage 7: Professional Reflection and Stewardship

Learning Focus Learners integrate their experience into a broader professional identity.

They reflect on engineering responsibility, AI use, evidence, teamwork, review, release readiness, operations, and future practice.

This stage connects student work to long-term trustworthy engineering.

Learner Questions Students begin asking:

How has my understanding of engineering changed?

What evidence shows my growth?

How did I use AI responsibly?

Where did my team create trust?

What would I improve next time?

What kind of engineer am I becoming?

Educational Activities Possible activities include:

- Final engineering reflection
- Portfolio evidence review
- Team retrospective
- Postmortem review
- Future engineer reflection
- Professional identity discussion

Expected Evidence Learners may produce:

- Final reflection
- Portfolio summary
- Team retrospective
- Postmortem
- Lessons learned

- Future trustworthy engineer statement

Developmental Shift

Completing a course

↓

Owning a professional engineering identity

Relationship to the Two-Cycle Engineering Model

The learning progression can be mapped onto the Two-Cycle Engineering Model.

Cycle 1: Can it work?

Orientation Intent Formation Engineering Structure Initial Construction and Review

Cycle 2: Can it survive?

Improved Construction and Review Verification and Release Defense Operational Awareness Professional Reflection and Stewardship

Cycle 1 establishes feasibility and basic engineering structure.

Cycle 2 deepens evidence, reviewability, operational awareness, and professional accountability.

The progression helps explain why Cycle 2 should not merely be more implementation.

Cycle 2 should represent more mature engineering.

Relationship to the ETIS Book

The learning progression aligns with the 39-chapter ETIS book.

Approximate chapter alignment:

Learning Stage	Supporting Chapters
Orientation	1-7
Intent Formation	10-12
Engineering Structure	8-9, 13-15
Construction and Review	16-18
Verification and Release Defense	19-22
Operational Awareness	23-32
Professional Reflection and Stewardship	33-39

This alignment is approximate.

Educational implementations may adapt sequencing and emphasis.

Relationship to Shared Assets

Shared assets support the learning progression.

Learning Stage	Supporting Shared Assets
Orientation	Playbooks, AI Engineering
Intent Formation	Templates, Workflows

Learning Stage	Supporting Shared Assets
Engineering Structure	Templates, Workflows
Construction and Review	Workflows, AI Engineering
Verification and Release Defense	Templates, Assessment
Operational Awareness	Templates, Workflows, Assessment
Professional Reflection and Stewardship	Playbooks, Assessment

Shared assets should help learners practice each stage rather than simply read about it.

Relationship to Assessment

Assessment should align with the learning progression.

Early assessment may focus on:

- Participation
- Repository setup
- Team expectations
- Initial requirements
- Basic planning

Middle assessment may focus on:

- Architecture
- Reviews
- Traceability
- AI-use verification
- Testing

Later assessment may focus on:

- Release readiness
- Evidence quality
- Operational awareness
- Reflection
- Professional growth

Assessment should reinforce progression rather than treating every deliverable as isolated.

Instructor Use

Instructors should use this progression to design coherent learning experiences.

The progression helps instructors decide:

- What students should encounter first
- When to introduce engineering artifacts
- When to introduce review
- When to deepen AI accountability
- When to shift from implementation to release readiness
- When to introduce operational thinking
- How to frame final reflection

The progression should make a course feel like a deliberate professional journey.

Student Use

Students should use this progression to understand why the course changes over time.

Early in the course, the focus may feel like setup and structure.

Middle work may feel like engineering production and review.

Later work may feel like defense, operation, and reflection.

That progression is intentional.

Students should understand that the course is not simply asking for more artifacts.

It is asking them to practice increasingly mature engineering responsibilities.

Relationship to COMP330

COMP330 can use this progression as the backbone for a semester-long software engineering experience.

A typical COMP330 use may look like:

Early Course: Orientation Intent Formation Engineering Structure

Mid Course: Construction and Review Initial Verification

Late Course: Verification and Release Defense Operational Awareness Professional Reflection

This progression supports the two-cycle model and helps ensure that Cycle 2 raises the maturity level rather than simply extending Cycle 1.

Future Evolution

This learning progression may be adapted for different educational implementations.

A graduate course may spend more time on architecture, governance, and operations.

A professional workshop may compress the progression into modules.

A capstone sequence may stretch the progression across multiple terms.

A short AI governance workshop may use only selected stages.

Future adaptations should preserve the underlying movement from concept awareness to trustworthy engineering practice.

Core Thesis

Software engineering learning is a progression from awareness to responsibility.

Learners first understand what engineering requires.

Then they practice creating evidence, making decisions, reviewing work, verifying claims, preparing releases, thinking operationally, and reflecting professionally.

The ETIS Software Engineering Learning Progression exists to make that journey visible, teachable, and repeatable.

COMP 330 Two-Cycle Engineering Project Model

Software Engineering in the AI Era

Loyola University Chicago

An ETIS Educational Ecosystem Flagship Implementation

Provenance

This document originated from Loyola University Chicago COMP 330 — Software Engineering.

It serves as the flagship implementation of the ETIS Two-Cycle Engineering Model.

While many concepts may eventually become reusable educational assets, this document intentionally preserves COMP330 provenance because it reflects a specific educational implementation.

Future ETIS educational implementations may adapt the model while preserving the underlying principles.

Core Idea

The COMP 330 project is not two coding deadlines.

Cycle 1 proves that the team can build a controlled vertical slice with evidence.

Cycle 2 proves that the team can learn from Cycle 1, improve the system, and make a defensible final release judgment.

1. Purpose of the Two-Cycle Project Model

The COMP 330 project is organized as two development cycles because professional software engineering is not a one-shot coding exercise.

Teams must learn to define the problem, make commitments, build a controlled first release, review evidence, learn from defects and feedback, and then mature the system.

The two-cycle model intentionally separates first-release capability from engineering maturity.

A working Cycle 1 demo matters, but it is not the end of the project. The stronger professional question is what the team learned from Cycle 1 and how that evidence changed Cycle 2 decisions.

Cycle 1: Controlled Vertical Slice Cycle 1 asks:

Can it work?

The team demonstrates that it can organize, plan, architect, construct, review, test, and release a small working system with evidence.

Cycle 2: Evidence-Based Maturity Cycle 2 asks:

Can it survive?

The team uses postmortem evidence, defects, risks, estimates, testing gaps, review feedback, and presentation feedback to improve quality, security, governance, observability, and release readiness.

2. Team Size, Team Formation, and Team Engineering Expectations

Teams will be formed by Week 2.

The expected team size is usually four to five students.

A three-person team can work, but it is fragile.

A six-person team is possible, but it should be treated as the upper limit because coordination overhead and hidden work increase quickly.

Software engineering is a team discipline.

Everyone on the team is a developer and engineering contributor first. Every student is expected to contribute to implementation, review, testing, documentation, and repository evidence.

Additional operational roles exist to make ownership visible, not to excuse anyone from technical contribution.

Every team member must own visible repository work.

Repository-visible workflow evidence should include issues, branches, pull requests, reviews, commits, and traceability links as the project matures.

Every team member must participate in review and evidence creation.

Every team member must understand the system well enough to explain the team's main engineering claims.

Every role must have a primary owner and a backup so the team does not fail when one person is unavailable.

> **Professional warning**
> Uneven contribution is common in student teams and in industry teams. COMP 330 does not assume every team works perfectly. It requires visible ownership, weekly status checks, GitHub evidence, pull requests, reviews, peer feedback, and role accountability so contribution problems become visible early instead of appearing at the deadline.

3. Professional Responsibility Roles

The following five operational roles should be assigned to students.

Smaller teams may combine roles.

Larger teams may split responsibilities if approved by the instructor.

The role model is intentionally lightweight. It creates accountability without turning the project into bureaucracy.

Role	Primary Responsibility	Typical GitHub Evidence Owned
Team Lead	Meeting cadence, team coordination, decision visibility, milestone readiness, team accountability	`/docs/team/team-charter.md`, `/docs/team/working-agreements.md`, meeting notes, milestone checklists
Planning & Process Lead	Scope, task plan, estimates, risks, schedule, issue hygiene, progress tracking	`/docs/planning/`, GitHub Issues, `/docs/planning/risk-register.md`, `/docs/planning/traceability.md`
Architecture & Development Lead	Architecture coherence, repository structure, implementation coordination, technical decision records	`/src`, `/docs/architecture/`, `/docs/decisions/`, `README.md`

Role	Primary Responsibility	Typical GitHub Evidence Owned
Quality & Review Lead	Testing, pull request discipline, review evidence, CI/CD checks, defect tracking, validation evidence	`/tests`, `/docs/testing/`, `/docs/reviews/`, `/.github/workflows/`, `/docs/quality/`
Operations & Evidence Lead	Release evidence, observability, known limitations, AI-use evidence, final evidence index, presentation evidence readiness	`/docs/release/`, `/docs/observability/`, `/docs/ai/`, `/docs/security/`, security-governance checklist

Role ownership means responsibility for correctness, completeness, and readiness of the evidence at the proper point in the cycle.

It does not mean the owner is the only person allowed to update the artifact.

Any team member may update any evidence artifact when doing legitimate project work.

The owner is responsible for making sure the artifact is accurate, current, reviewable, and linked to the rest of the project evidence when the milestone arrives.

Ownership Rules

Ownership Level	Responsibility
Primary owner	Accountable for keeping the artifact correct and ready
Backup owner	Able to step in if the primary owner is unavailable
All team members	Allowed and expected to update evidence when their work affects it
Team	Responsible for reviewing important evidence before submission or presentation

4. Minimum Weekly Cadence Meetings

Each team must run a minimum weekly cadence meeting.

This can be on Zoom or in person, on any day and time the team chooses.

The purpose is not to hold a long engineering design session.

The purpose is to keep status, ownership, blockers, commitments, and evidence visible.

When done correctly, the weekly cadence meeting should usually take 15 to 30 minutes, with a target closer to 15 minutes.

Longer design discussions, debugging sessions, architecture debates, or pair-programming work should be scheduled separately.

Recommended Cadence Agenda Each weekly cadence meeting should answer:

1. What did each person complete since the last checkpoint?
2. What is each person doing next?

3. What is blocked, unclear, risky, or behind schedule?
4. What GitHub evidence changed?
5. What agreement or decision did the team make?
6. What repository evidence, issue updates, pull requests, or documentation changes are required before the next checkpoint?
7. What must be ready before the next class, milestone, or submission?

GitHub evidence may include:

- Issues
- Branches
- Pull requests
- Tests
- Documentation
- AI-use log entries
- Architecture decisions
- Release evidence

Cadence rule
A cadence meeting is a status and accountability checkpoint. It is not the place to design the entire architecture, debug a hard failure, or debate every implementation option. Those are real engineering meetings, but they should be scheduled separately.

5. Cycle 1 Expectations: Controlled Vertical Slice

Cycle 1 is intentionally longer than the raw implementation size of the default project would require in industry.

A professional team assigned a small version of this project might implement much of the basic functionality quickly.

COMP 330 slows the work down on purpose because students are learning the full engineering discipline around the work, not just the code.

Cycle 1 is the crawl phase.

The team is learning how to form, define scope, create evidence, make estimates, document architecture, use GitHub professionally, review AI-assisted work, test the system, and present a release.

Weekly progress is still required.

Teams that wait until the deadline will struggle because late work leaves no time for review, testing, integration, defect correction, or evidence cleanup.

Cycle 1 Minimum Expectations By the end of Cycle 1, the team should have:

- Formed the team and assigned roles, with backups identified
- Created and used the GitHub repository as the authoritative engineering record
- Established the repository `README.md` as the professional front door to the project and major engineering evidence
- Defined a Cycle 1 vertical slice and kept it intentionally small
- Documented requirements, acceptance criteria, assumptions, risks, and out-of-scope decisions
- Maintained task plan, estimates, schedule, traceability, and team commitments
- Made architecture, interfaces, data/context ownership, and governance boundaries reviewable
- Completed implementation through issues, branches, pull requests, reviews, and tests
- Disclosed, reviewed, verified, and owned AI use
- Presented the Cycle 1 release with evidence, known limitations, risks, and lessons learned

Cycle 1 Professional Standard Cycle 1 does not prove that the system is mature.

Cycle 1 proves that the team can create a controlled, reviewable, evidence-backed first release.

6. Cycle 2 Expectations: Evidence-Based Maturity

Cycle 2 is shorter and should not become a feature binge.

It is not a restart.

It is also not a promise to complete every improvement idea listed in the project overview.

Cycle 2 work must be selected and scoped based on evidence from Cycle 1.

The Cycle 1 postmortem is the decision point.

Teams should use defects, missed estimates, weak tests, review feedback, architecture pain, AI-use issues, presentation questions, and known limitations to choose the highest-value maturity work.

The goal is to improve the credibility of the system, not simply add sparkle.

Cycle 2 Minimum Expectations During Cycle 2, the team should:

- Review Cycle 1 evidence before choosing Cycle 2 work
- Identify root causes, not just symptoms
- Re-estimate remaining and new work based on actual Cycle 1 performance
- Select a small number of maturity targets the team can complete and prove
- Improve tests, defects, architecture, observability, runtime visibility, security/governance, AI-use controls, or release documentation as appropriate
- Document what was intentionally deferred and why
- Prepare a final release argument supported by repository evidence
- Identify the repository state associated with major releases using release tags, versions, or equivalent release markers when appropriate

Cycle 2 Professional Standard Cycle 2 proves whether the team can learn.

A mature Cycle 2 release should show improved engineering judgment, stronger evidence, clearer release claims, and more realistic awareness of operational risk.

7. Evidence Ownership by Role and Cycle

The table below connects operational roles to the main evidence they are expected to keep healthy.

These are ownership lanes, not walls.

Teammates may and should help each other, but the owner is accountable for readiness at each milestone.

Engineering evidence should be reviewable without requiring verbal clarification from the original team.

Role	Evidence Owned	Why It Matters
Team Lead	Team charter, working agreements, meeting notes, milestone readiness	Keeps team decisions, communication rhythm, and submission readiness visible
Planning & Process Lead	Scope, task plan, estimates, risk register, schedule, traceability	Keeps work bounded, sequenced, owned, and connected to requirements

Role	Evidence Owned	Why It Matters
Architecture & Development Lead	Architecture package, decision records, repository structure, implementation coordination	Keeps the design understandable, changeable, and aligned with implementation
Quality & Review Lead	Tests, defect log, pull request review evidence, CI/CD evidence, validation evidence	Keeps quality claims backed by inspection, tests, reviews, and defect discipline
Operations & Evidence Lead	AI-use log, release notes, known limitations, observability evidence, security/governance evidence, final evidence index	Keeps release claims, AI accountability, operational maturity, and final presentation evidence defensible

8. How This Compares to Industry Practice

Industry teams often operate with short status rituals, explicit ownership, issue tracking, pull requests, build checks, release notes, and postmortems.

COMP 330 uses a classroom version of that pattern.

The goal is not to imitate every industry tool or ceremony.

The goal is to practice the engineering habits that make team software controllable.

Professional engineering maturity includes operational stewardship, accountable review behavior, honest risk communication, and evidence that another engineer can independently inspect.

In professional teams, a 15-minute status check is not where all engineering work happens.

It is where the team synchronizes.

The real work happens in issues, design notes, pull requests, tests, reviews, documentation updates, and focused working sessions.

COMP 330 expects the same distinction.

Professional teams also experience uneven contribution, missed estimates, unclear ownership, and communication failures.

The professional response is not denial.

The professional response is visibility: identify the problem, adjust responsibilities, document risks, and make sure the evidence reflects reality.

9. What Is Not Acceptable

The following behaviors are not acceptable in COMP 330 project work:

- Waiting until the deadline to create GitHub evidence
- Treating the project as separate individual parts that are never integrated
- Using AI-generated work that the team cannot explain or verify
- Skipping reviews because the code appears to work
- Holding vague meetings with no decisions, owners, or next actions
- Submitting a polished document that does not point to repository evidence
- Presenting a demo while hiding known limitations, defects, or risks
- Treating the final presentation as proof instead of repository-supported engineering evidence
- Allowing one or two students to carry the project while others remain invisible

10. What Teams Should Do Immediately

Teams should do the following as soon as project work begins:

1. Confirm team membership by Week 2.
2. Select a team name and primary communication channel.
3. Assign the five professional responsibility roles and identify backups.
4. Choose the default project or propose an instructor-approved alternative.
5. Schedule the weekly cadence meeting.
6. Create the GitHub repository and minimum folder structure.
7. Draft the team charter, role matrix, working agreements, and AI-use policy.
8. Define the Cycle 1 vertical slice and the first small set of issues.
9. Identify which evidence artifacts each role owner must keep current before the first submission.

Relationship to the ETIS Learning Models

This document operationalizes the Two-Cycle Engineering Model inside the Loyola COMP330 flagship implementation.

It connects directly to the broader ETIS learning model architecture.

Professional Transformation Model The two-cycle model helps students move from:

Student
↓
Responsible Engineer
↓
Reviewer
↓
Release Defender

Cycle 1 introduces responsible engineering behavior.

Cycle 2 strengthens reviewability, release defense, and operational awareness.

Engineering Maturity Model Cycle 1 primarily moves students from task completion toward structured participation.

Cycle 2 moves students toward evidence-based engineering and reviewable engineering judgment.

Software Engineering Learning Progression The two-cycle model gives the learning progression a project rhythm.

Cycle 1 emphasizes orientation, intent formation, engineering structure, and initial construction.

Cycle 2 emphasizes improved construction, verification, release defense, operational awareness, and professional reflection.

Final Takeaway

The project succeeds when the team can show both a useful system and the engineering evidence behind it.

A professional team does not ask others to trust a demo.

It shows the requirements, design, review, tests, risks, AI-use decisions, release notes, and maturity evidence that make the demo credible.

Part III

Educational Products

ETIS Course Design Guide

The **ETIS Course Design Guide** provides a structured approach for designing courses that teach *Engineering Trustworthy Intelligent Systems (ETIS)*.

ETIS course design is fundamentally different from traditional software engineering course design.

Traditional courses often organize instruction around technical topics.

ETIS organizes instruction around engineering responsibility, evidence generation, reviewability, operational thinking, and professional accountability.

The objective is not to help students build software.

The objective is to help students become trustworthy engineers.

Purpose

The purpose of this guide is to help instructors design ETIS-based learning experiences that are coherent, teachable, maintainable, and aligned with ETIS doctrine.

This guide helps instructors answer questions such as:

- What should students become by the end of the course?
- Which ETIS capabilities should be emphasized?
- How should assignments be sequenced?
- What evidence should students produce?
- How should AI-assisted work be governed?
- How should class time be used?
- How should engineering maturity be measured?
- How should final engineering accountability be demonstrated?

Course design should optimize for professional formation rather than content coverage.

ETIS Course Design Philosophy

ETIS courses are not collections of lectures.

ETIS courses are engineered systems.

Every component should exist for a reason.

Students should understand why they are performing an activity and what engineering responsibility it teaches.

Course components should work together as an integrated system.

This includes:

- lectures,
- readings,
- repositories,
- assignments,
- exercises,
- reviews,
- assessments,
- AI-use governance,
- release readiness activities,
- operational thinking,
- and final defense activities.

The course itself should model trustworthy engineering.

Class sessions should be engineered, not planned.

Sessions are not isolated instructional events.

Sessions are components of a larger educational system.

Each session should intentionally move students along the ETIS transformation model while creating opportunities for:

- ownership,
- evidence generation,
- reviews,
- engineering accountability,
- AI governance,
- and operational thinking.

Course design creates educational systems rather than collections of class meetings.

Foundational Course Design Principle

Course design begins with a simple principle:

> Design for professional transformation, not information transfer.

Information is abundant.

AI can generate explanations, examples, summaries, and artifacts.

The educational value of ETIS comes from helping learners practice engineering judgment.

Course Design Hierarchy

ETIS course design follows a hierarchy.

Professional Transformation Goal

↓

Engineering Capabilities

↓

Learning Experiences

↓

Assignments and Reviews

↓

Repository Evidence

↓

Assessment

↓

Professional Accountability

Design should always proceed from top to bottom.

Never start by designing assignments.

Assignments emerge from the transformation goals.

Step 1 — Define The Professional Transformation Goal

Every ETIS course should begin by defining who the learner should become.

Examples include:

Student

↓

Responsible Engineer

Student

↓

Reviewer

Student

↓

Release Defender

Student

↓

Operational Thinker

Student

↓

Future Trustworthy Engineer

A course may support multiple transformations, but one should be primary.

This transformation goal becomes the anchor for every design decision.

Step 2 — Select Primary Engineering Capabilities

ETIS contains many engineering capabilities.

No single course should emphasize all capabilities equally.

Select a focused set.

Examples include:

Foundation Capabilities

- requirements discipline,
- planning,
- estimation,
- risk identification,
- repository-centered engineering,
- architecture reasoning,
- technical communication.

Construction Capabilities

- implementation,
- testing,
- integration,
- pull requests,
- review participation,
- defect management.

Governance Capabilities

- AI-use governance,
- decision documentation,
- traceability,
- accountability,
- transparency.

Operational Capabilities

- observability,
- postmortems,
- release readiness,
- incident response,
- operational maturity.

Stewardship Capabilities

- continuous improvement,
- long-term ownership,
- engineering memory,
- trustworthy systems thinking.

Capability selection creates focus.

Step 3 — Map The ETIS Book

The ETIS book remains authoritative.

Course design should map the book intentionally.

Instructors should identify:

Required Chapters Students must understand these concepts deeply.

Supporting Chapters Students should be exposed to these concepts.

Reference Chapters Students may consult these concepts as needed.

For example:

Undergraduate Software Engineering Course Primary emphasis:

- Part I
- Part II

Selective introduction:

- Parts III and IV

Graduate Course Primary emphasis:

- Parts II, III, and IV

Professional Training Focused subset tied to organizational needs.

The course should not attempt to cover all chapters equally.

Step 4 — Design Repository Expectations

The repository should become the authoritative engineering record.

Students should understand that the repository is evidence, not storage.

Repository expectations should be defined before assignments are created.

A repository may include:

/docs /requirements /planning /architecture /decisions /reviews /testing /quality /release /operations /observability /security /governance /ai /postmortems /presentations

/src

/tests

/.github

The exact structure may vary.

The evidence expectations should not.

Step 5 — Design Engineering Phase Gates

Assignments should function as maturity gates.

Each assignment should increase engineering responsibility.

Example sequence:

Phase Gate 1 Repository foundation.

Phase Gate 2 Requirements, planning, risk, and traceability.

Phase Gate 3 Architecture and review.

Phase Gate 4 Implementation, testing, and validation.

Phase Gate 5 Cycle 1 release readiness.

Phase Gate 6 Postmortem and stabilization.

Phase Gate 7 Operational readiness.

Phase Gate 8 Final engineering defense.

Students should feel the system becoming more mature over time.

Step 6 — Design The Two-Cycle Model

ETIS courses benefit from two engineering cycles.

Cycle 1 — Can It Work? Focus:

- requirements,
- planning,
- architecture,
- implementation,
- testing,
- controlled release.

Students demonstrate responsible construction.

Cycle 2 — Can It Survive? Focus:

- feedback,
- defects,
- observability,
- risks,
- governance,
- postmortems,
- operational maturity.

Students demonstrate responsible operation.

The second cycle is often where ETIS becomes differentiated from traditional courses.

Step 7 — Design AI Governance

AI should be integrated deliberately.

Students should understand:

AI May Assist With

- requirements exploration,
- ambiguity detection,
- planning,
- architecture critique,
- implementation,
- testing,
- documentation,
- review preparation,
- operational reasoning.

Students Remain Responsible For

- correctness,
- security,
- maintainability,
- architecture fit,
- testing,
- documentation,
- disclosure,

- validation,
- professional judgment.

The governing principle remains:

> AI-assisted work is not accepted engineering work until humans review, verify, and own it.

Step 8 — Design Classroom Experiences

Class time should not become lecture-only time.

ETIS encourages active engineering experiences.

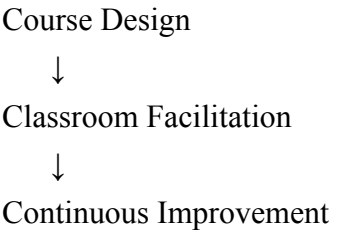
Examples include:

- requirements review,
- ambiguity workshops,
- architecture critiques,
- ADR discussions,
- AI-use reviews,
- pull request simulations,
- code review workshops,
- testing evidence reviews,
- release readiness table-tops,
- incident response simulations,
- postmortem analysis,
- final defense preparation.

Students should practice engineering, not only hear about engineering.

Relationship To Educational Operations Course design establishes classroom systems.

Those systems are later operated through the ETIS Classroom Facilitation engine.



Course design answers:

> What educational system should exist?

Classroom facilitation answers:

> How should instructors operate that system?

These educational engines intentionally work together.

Step 9 — Design Assessment

Assessment should evaluate both products and evidence.

Students should be assessed on:

- engineering intent,
- evidence quality,

- architecture reasoning,
- review participation,
- AI accountability,
- testing discipline,
- risk communication,
- release readiness,
- operational thinking,
- professional defense.

A functioning application alone is insufficient.

Grades measure performance.

Engineering maturity signals measure transformation.

Both are important.

Grades measure what students produced.

Maturity signals help instructors observe who students are becoming.

Course designs should intentionally create opportunities for instructors to observe engineering maturity throughout the semester.

Step 10 — Design The Final Defense

Every ETIS course should culminate in accountability.

Students should answer questions such as:

- What problem were you solving?
- Why was this architecture selected?
- How did AI assist?
- What evidence supports your claims?
- What risks remain?
- How would this system be operated?
- What would you improve?
- Why is this release defensible?

Final defense is one of the defining ETIS experiences.

Recommended Course Components

A mature ETIS course may include:

Course Introduction

Repository Foundation

Student Starter Kit

Reading Assignments

Engineering Phase Gates

Classroom Exercises

Reviews and Critiques

AI Governance

Release Readiness

Operational Thinking

Final Engineering Defense

Not every course needs every component.

The design should remain purposeful.

Common Course Design Mistakes

Mistake 1 — Designing Around Topics Topics are insufficient.

Design around professional transformation.

Mistake 2 — Overloading Students With Process Discipline is not bureaucracy.

Every artifact should teach responsibility.

Mistake 3 — Assessing Only Technical Output Assessment must include evidence and accountability.

Mistake 4 — Treating AI As Either Forbidden Or Unlimited AI should be governed.

Mistake 5 — Ignoring Operations Software does not end at deployment.

Mistake 6 — Copying COMP330 COMP330 is an implementation, not a universal template.

Mistake 7 — Forgetting The Final Defense Students should leave with experience defending engineering decisions.

Course Design Review Checklist

Before launching a course, the instructor should be able to answer:

Transformation

- Who should learners become?

Capabilities

- What engineering capabilities are primary?

Book Mapping

- Which ETIS chapters are required?

Repository

- What evidence will students create?

Phase Gates

- How will maturity increase?

AI Governance

- How will AI be governed?

Classroom Experiences

- How will students practice engineering?

Assessment

- How will accountability be measured?

Final Defense

- How will students demonstrate engineering maturity?

If these questions are unclear, the course design is incomplete.

Relationship To Educational Stewardship

Course design is the beginning of educational stewardship.

Educational systems should improve over time rather than restart every semester.

The ETIS Instructor Notes engine preserves educational wisdom gained while operating courses.

Course Design

↓

Classroom Facilitation

↓

Instructor Notes

↓

Continuous Improvement

Course design creates educational systems.

Classroom facilitation operates those systems.

Instructor notes preserve institutional memory and continuously improve those systems.

Educational systems are engineered.

Educational systems are also stewarded.

Stewardship Rules

Course design should preserve these boundaries:

- Do not duplicate shared assets.
- Do not duplicate the student starter kit.
- Do not duplicate the ETIS book.
- Do not treat COMP330 as the ecosystem.
- Do not create assignments without purpose.
- Do not create documentation theater.
- Do not teach AI without accountability.
- Do not teach software without operational responsibility.

The course should remain teachable, maintainable, and adaptable.

Guiding Standard

Every course component should answer at least one of these questions:

- What engineering responsibility is being taught?
- What evidence will students create?
- What judgment will students practice?
- What review will students survive?
- What risk will students learn to see?
- What professional habit will students carry forward?

If a component cannot answer one of those questions, it should be reconsidered.

Core Commitment

The ETIS Course Design Guide exists to help instructors build courses that form trustworthy engineers.

Students should leave the course understanding that software engineering is not merely software construction.

It is the disciplined practice of creating systems that can be understood, reviewed, governed, operated, improved, and trusted over time.

ETIS Student Starter Kit Architecture

Purpose

This document defines the architecture of Student Starter Kits within the ETIS (Engineering Trustworthy Intelligent Systems) Educational Ecosystem.

The purpose of this architecture is to establish how ETIS translates educational theory into student engineering practice.

Student Starter Kits provide the environment where learners repeatedly practice trustworthy engineering behaviors.

They are the operational bridge between educational design and student engineering work.

Core Philosophy

A Student Starter Kit is a product.

It is not a repository template.

It is not a collection of files.

It is not a starter assignment.

It is an intentionally assembled engineering environment.

The objective is to create environments where students naturally practice ETIS principles as part of their daily engineering activities.

The environment itself should teach engineering.

Educational Role

Student Starter Kits occupy a specific position within the ETIS Educational Ecosystem.

ETIS Book

↓

Educational Architecture

↓

Learning Models

↓

Shared Assets

↓

Student Starter Kit

↓

Student Engineering Practice

↓

Professional Transformation

Student Starter Kits operationalize ETIS.

The completed Instructor Course Package sits alongside the Student Starter Kit.

The Instructor Course Package designs, operates, and stewards educational systems.

The Student Starter Kit provides the engineering environment where students repeatedly practice those systems.

Their relationship can be summarized as:

Instructor Course Package

↓

Student Starter Kit

↓

Student Engineering Practice

↓

Professional Transformation

The two systems are intentionally complementary.

One teaches instructors how to create trustworthy engineers.

The other provides environments where students repeatedly practice becoming trustworthy engineers.

They transform educational concepts into engineering habits.

Core Educational Problem

Traditional software engineering courses often begin with an empty repository.

Students then gradually discover:

- Documentation
- Testing
- Architecture
- Reviews
- Traceability
- Operational thinking

These disciplines often appear late in the project.

ETIS intentionally reverses this pattern.

Students should begin inside an engineered environment that already expects professional engineering behavior.

The starter kit reduces educational randomness.

Architectural Goals

Student Starter Kits exist to achieve several goals.

Goal 1: Normalize Professional Engineering Structure Students should encounter professional engineering organization immediately.

Goal 2: Teach Repository-Centered Engineering Repositories should function as engineering memory systems.

Goal 3: Teach Evidence-Centered Engineering Claims should be supported by evidence.

Goal 4: Teach Responsible AI Collaboration AI should be treated as a bounded collaborator.

Goal 5: Teach Reviewability Engineering work should be understandable by others.

Goal 6: Teach Operational Thinking Students should think beyond implementation.

Goal 7: Teach Accountability Students should own outcomes.

Architectural Components

Every Student Starter Kit consists of three layers.

Engineering Workspace

↓

Engineering Evidence System

↓

Engineering Learning System

Each layer serves a different purpose.

Layer 1: Engineering Workspace

This is the visible repository structure students interact with.

Examples may include:

README.md

docs/

src/

tests/

scripts/

data/

The workspace organizes engineering activities.

This layer should remain intentionally simple.

Layer 2: Engineering Evidence System

This layer defines what evidence students are expected to preserve.

Examples include:

- Requirements
- Assumptions
- Risks
- Architecture
- Decisions
- AI-use logs
- Test evidence

- Review evidence
- Release evidence
- Operational assumptions

Students should learn that evidence accumulates continuously.

Evidence should not be generated at the end of the semester.

Layer 3: Engineering Learning System

This layer connects repository activities to educational outcomes.

Repository activities should reinforce:

- Professional transformation
- Engineering maturity
- Review skills
- AI accountability
- Operational awareness
- Stewardship

The environment itself should help learners mature.

The Seven Engineering Responsibilities

Every Student Starter Kit should reinforce seven ETIS responsibilities.

Define Intent

Engineer Context

Bound Authority

Verify Behavior

Operate Reality

Explain Decisions

Own Outcomes

These responsibilities should be visible throughout the engineering environment.

Design Philosophy

Student Starter Kits should embody several design characteristics.

Visible Students should be able to easily understand where work belongs.

Understandable The environment should not overwhelm beginners.

Reviewable Engineering evidence should be easy to inspect.

Sustainable The structure should remain useful throughout the entire project lifecycle.

AI-Aware Responsible AI practices should be integrated naturally.

Operationally Conscious Students should think beyond implementation.

Evolvable The environment should support future growth without structural instability.

Repository-Centered Engineering

Student Starter Kits reinforce one of the foundational ETIS concepts:

> Repositories preserve engineering memory.

Repositories should contain more than code.

They should preserve:

- Intent
- Decisions
- Evidence
- Reviews
- Risks
- Tests
- Operational assumptions
- Reflection

Students should gradually understand that engineering memory is one of their primary responsibilities.

Evidence-Centered Engineering

Student Starter Kits should teach students that evidence is continuous.

Evidence may include:

Requirements

↓

Architecture

↓

Implementation

↓

Tests

↓

Reviews

↓

Release Evidence

↓

Operational Learning

Every phase should leave evidence behind.

AI-Aware Engineering

Student Starter Kits should support responsible AI usage.

Students should have clear places to preserve:

- AI policies

- AI-use logs
- Verification notes
- Human review evidence

The starter kit should reinforce:

> AI proposes; engineers verify.

AI should not become an invisible participant.

Relationship to Learning Models

Student Starter Kits consume learning models.

Learning models explain how students develop.

Starter kits provide environments where that development occurs.

Examples include:

- Professional Transformation Model
- Engineering Maturity Model
- Software Engineering Learning Progression
- Two-Cycle Engineering Model

The starter kit operationalizes those models.

Relationship to Shared Assets

Student Starter Kits consume shared assets.

Shared assets remain authoritative.

Examples include:

shared ETIS educational assets

playbooks/

templates/

workflows/

ai_engineering/

assessment/

Student Starter Kits may surface these assets in student-friendly ways.

They should not permanently duplicate them.

Relationship to Adoption Examples

Adoption examples explain how starter kits are used.

The current flagship implementation is:

adoption_examples/ ‘— loyola_comp330/

The adoption example explains instructional use.

The starter kit provides the student environment.

The relationship is complementary.

The flagship implementation proves the doctrine.

It does not become the doctrine.

The COMP330 starter kit should never become the universal ETIS starter kit.

Future educational implementations should inherit ETIS principles while adapting to their own environments.

Institutions should inherit ETIS doctrine, not ETIS implementations.

Architectural Boundaries

Student Starter Kits do not own:

- Course schedules
- Syllabi
- Rubrics
- Instructor slides
- Instructor analysis tools
- Educational architecture
- Learning models
- Shared asset sources

Student Starter Kits own the student engineering environment.

Growth Model

New starter kits should only be created when educational environments materially differ.

Examples may include:

- Undergraduate software engineering
- Graduate software engineering
- Capstone programs
- Professional workshops
- AI governance programs

Different semesters alone should not create new starter kits.

Governance Rules

The following rules should remain stable.

Starter kits are products.

Starter kits teach through structure.

Starter kits should consume shared assets.

Starter kits should preserve provenance.

Starter kits should avoid unnecessary complexity.

Starter kits should evolve from educational evidence.

Starter kits should support long-term maintainability. These rules help preserve architectural consistency.

Layer Separation Rules

The Student Starter Kit intentionally separates engineering activities into distinct layers.

docs/ → Think

src/ → Build

tests/ → Verify

data/ → Support

scripts/ → Automate

These responsibilities should remain distinct.

Do not duplicate engineering evidence across layers.

Do not move educational artifacts into implementation layers.

Do not allow tooling implementations to become educational doctrine.

This separation helps preserve long-term maintainability.

Future Evolution

Future enhancements may include:

- Multiple starter kit variants
- Discipline-specific starter kits
- Professional training starter kits
- AI governance starter kits

Growth should occur intentionally.

The architecture should remain simple.

Core Thesis

Student Starter Kits are where ETIS becomes practice.

They provide environments where students repeatedly exercise the habits of trustworthy engineering.

The goal is not to provide files.

The goal is to create engineering environments that help learners define intent, engineer context, bound authority, verify behavior, operate reality, explain decisions, and own outcomes.

That repeated practice is what ultimately produces trustworthy engineers.

Starter Kit Design Principles

Purpose

This document defines the design principles that govern Student Starter Kits within the ETIS (Engineering Trustworthy Intelligent Systems) Educational Ecosystem.

The purpose of these principles is to ensure that Student Starter Kits remain intentional, understandable, maintainable, and aligned with trustworthy engineering practices.

These principles should guide the creation of all current and future starter kits.

They are intended to create consistency without preventing thoughtful adaptation.

Core Philosophy

Student Starter Kits should teach engineering through environment design.

The environment itself should reinforce trustworthy engineering behaviors.

Students should learn engineering practices naturally by working inside well-designed systems rather than being introduced to those practices as disconnected requirements throughout a course.

The environment should become an instructor.

The environment should teach without overwhelming.

Students should repeatedly encounter trustworthy engineering behaviors without becoming burdened by unnecessary complexity.

Well-designed environments reduce educational friction while preserving engineering accountability.

Principle 1: Design for Professional Practice

Student Starter Kits should resemble professional engineering environments.

Students should encounter:

- Repository organization
- Documentation expectations
- Review expectations
- Evidence expectations
- Operational awareness

Students should not begin inside artificial academic environments that bear little resemblance to professional engineering practice.

The objective is not realism for its own sake.

The objective is to normalize responsible engineering behavior.

Principle 2: Design for Repository-Centered Engineering

Repositories should function as engineering memory systems.

Starter kits should make it obvious that repositories contain more than code.

Students should have visible places to preserve:

- Intent
- Context
- Decisions

- Risks
- Reviews
- Tests
- AI usage
- Release evidence
- Operational knowledge

Repository organization should support long-term understanding.

Principle 3: Design for Evidence-Centered Engineering

Starter kits should teach that engineering claims require evidence.

Students should be encouraged to answer questions such as:

- What was required?
- What was built?
- What was tested?
- What was reviewed?
- What risks remain?
- What limitations exist?

The repository should make evidence creation a normal activity.

Evidence should accumulate continuously.

Principle 4: Design for Understandability

Students should be able to quickly understand the engineering environment.

Starter kits should avoid unnecessary complexity.

The organization should be intuitive.

Students should spend their effort solving engineering problems rather than deciphering repository structure.

Complexity should be introduced intentionally and only when educationally valuable.

The governing principle should be:

> Scale complexity, not accountability.

As students mature, engineering challenges may become more sophisticated.

Engineering accountability should remain visible from the beginning.

Principle 5: Design for Progressive Maturity

Starter kits should support learner growth.

The environment should remain useful from project launch through release readiness and reflection.

The same repository should support progression from:

Task Completion ↓ Evidence-Based Engineering ↓ Reviewable Engineering ↓ Operational Thinking

The environment should mature with the learners.

Principle 6: Design for Reviewability

Engineering work should be easy to inspect.

A reviewer should be able to understand:

- What problem is being solved
- What decisions were made
- What changed
- What was reviewed
- What was tested
- What risks remain

Starter kits should make review a natural activity rather than a special event.

Principle 7: Design for Responsible AI Collaboration

AI is an expected participant in modern engineering environments.

Starter kits should support responsible AI use.

Students should have places to preserve:

- AI policies
- AI-use logs
- Verification notes
- Human review evidence

Starter kits should reinforce:

> AI proposes; engineers verify.

AI should never become invisible.

Principle 8: Design for Operational Thinking

Students should think beyond implementation.

Starter kits should encourage learners to ask:

- What happens after release?
- How would someone operate this?
- How would failures be discovered?
- What would future maintainers need to know?

Operational awareness should appear early rather than late.

Principle 9: Design for Accountability

Students should own outcomes.

Starter kits should encourage students to preserve:

- Known limitations
- Risks
- Assumptions
- Decisions
- Release evidence

Students should understand that engineering accountability extends beyond implementation.

Principle 10: Design for Long-Term Maintainability

Starter kits should remain stable over time.

The structure should evolve carefully.

New folders, documents, and expectations should be introduced only when they improve educational outcomes.

Avoid complexity for its own sake.

The environment should remain sustainable for both students and instructors.

Principle 11: Design for Reuse

Starter kits should consume reusable ETIS assets rather than duplicate them.

Starter kits should rely upon:

shared ETIS educational assets

Examples include:

- Playbooks
- Templates
- Workflows
- AI Engineering assets
- Assessment assets

The starter kit assembles these assets into a student experience.

It should not become another repository for educational intellectual property.

Principle 12: Design for Provenance

Starter kits should preserve their educational origins.

Students and future adopters should understand:

- Where the starter kit originated
- Which educational implementation uses it
- How it relates to ETIS

Provenance preserves educational memory.

Principle 13: Design for Evolution Through Evidence

Starter kits should evolve based on observed educational outcomes.

Useful inputs include:

- Student difficulties
- Repository analysis
- Instructor observations
- Assessment trends
- Reflection artifacts

Starter kits should not change based solely on preference.

Changes should be evidence informed.

Anti-Patterns

The following anti-patterns should be avoided.

Empty Repository Syndrome Students begin with no engineering structure.

Documentation Dumping Students generate documentation only at the end.

File Proliferation Too many folders or documents overwhelm students.

Hidden Expectations Students are expected to produce evidence without knowing where it belongs.

Invisible AI AI usage occurs without disclosure or verification.

Demo Thinking Students equate working software with engineering completion.

Educational Drift Starter kits slowly accumulate unrelated materials over time.

These anti-patterns reduce educational effectiveness.

Relationship to the Educational Ecosystem

These principles support all layers of ETIS.

ETIS Book ↓ Educational Architecture ↓ Learning Models ↓ Shared Assets ↓ Student Starter Kit ↓ Student Practice

The principles help translate ETIS into repeatable engineering behavior.

Future Evolution

These principles should remain relatively stable.

Future starter kits may vary in structure, audience, and complexity.

However, the underlying design philosophy should remain consistent.

Stability helps create a recognizable ETIS educational experience across implementations.

Core Thesis

Student Starter Kits should be intentionally engineered environments that teach trustworthy engineering through repeated practice.

The goal is not to provide students with files.

The goal is to provide environments that naturally reinforce professional engineering behaviors every time students interact with the repository.

ETIS Adoption Model

Purpose

This document defines how the ETIS (Engineering Trustworthy Intelligent Systems) Educational Ecosystem is adopted.

The purpose of this model is to provide a repeatable, governed approach for implementing ETIS within educational, professional, and organizational environments.

ETIS is intentionally designed to be adaptable.

However, adaptation should not create fragmentation.

This model explains how institutions, instructors, organizations, and adopters can implement ETIS while preserving the integrity of the framework.

Core Philosophy

ETIS should not be installed.

ETIS should be adopted.

Installation implies copying files and following instructions.

Adoption requires understanding educational goals, selecting learning pathways, consuming educational assets, and intentionally building a trustworthy engineering experience.

ETIS adoption is an engineering activity.

Core Adoption Principle

ETIS is one framework with many implementations.

The framework remains stable.

Implementations adapt to local needs.

The framework should not change every time a new implementation appears.

Instead, implementations should consume ETIS intentionally.

Adoption Architecture

Every ETIS implementation should follow the same high-level flow.

ETIS Book

↓

Educational Architecture

↓

Learning Models

↓

Shared Assets

↓

Student Starter Kits

↓

Educational Implementation

↓

Learner Experience

↓

Professional Transformation

This flow preserves architectural boundaries.

The ETIS book remains authoritative.

Educational architecture governs teaching.

Learning models govern learner development.

Shared assets provide reusable educational building blocks.

Student Starter Kits provide practice environments.

Adoption examples provide local implementation.

Adoption Is Not Duplication

Adopters should consume ETIS assets rather than recreate them.

This distinction is important.

Poor adoption looks like:

Copy everything.

Rename everything.

Modify everything.

Create local versions everywhere.

Good adoption looks like:

Consume ETIS.

Adapt intentionally.

Preserve provenance.

Create only what is implementation-specific.

The objective is sustainable reuse.

The Six Adoption Decisions

Every implementation should answer six questions.

Decision 1: Who Is The Audience? Examples:

- Undergraduate software engineering students
- Graduate students
- Capstone teams
- Early-career engineers
- Professional engineers
- Organizational learners

This decision determines educational emphasis.

Decision 2: What Is The Educational Goal? Examples:

- Foundational software engineering
- AI engineering
- Capstone integration
- Governance education
- Professional development
- Organizational transformation

This decision determines depth and scope.

Decision 3: Which ETIS Learning Path Is Being Used? Different audiences consume ETIS differently.

The implementation should identify:

- Chapter emphasis
- Learning model emphasis
- Professional transformation goals
- Assessment emphasis

The complete ETIS framework remains available even when selective emphasis is used.

Decision 4: Which Shared Assets Will Be Consumed? The implementation should identify which assets are required.

Examples:

- Playbooks
- Templates
- Workflows
- AI Engineering assets
- Assessment assets

Only implementation-specific materials should be created locally.

Decision 5: Will A Student Starter Kit Be Used? Most implementations should use a starter kit.

The starter kit provides:

- Repository structure
- Documentation structure
- Practice scaffolding
- Engineering workspace organization

The starter kit creates consistency.

Decision 6: What Evidence Will Learners Produce? ETIS is evidence-centered.

The implementation should intentionally define learner evidence.

Examples:

- Requirements
- Architecture

- Risk registers
- AI-use logs
- Tests
- Reviews
- Release evidence
- Operational evidence
- Reflection artifacts

Evidence should be intentional rather than accidental.

Adoption Lifecycle

ETIS adoption should occur in phases.

Phase 1: Understand ETIS The adopter learns the ETIS philosophy.

Questions include:

- What problem is ETIS solving?
- How is ETIS different from traditional software engineering education?
- What educational outcomes are desired?

Phase 2: Select The Learning Path The adopter selects:

- Audience
- Educational goals
- Chapter emphasis
- Learning models

This establishes scope.

Phase 3: Assemble Educational Assets The adopter selects:

- Shared assets
- Starter kits
- Classroom exercises
- Assessments

This assembles the learning environment.

Phase 4: Build The Local Implementation The adopter creates:

- Schedule
- Assignments
- Rubrics
- Slides
- Local implementation notes

This is where implementation-specific materials emerge.

Phase 5: Operate The Educational Experience Learners participate.

The implementation produces:

- Student work
- Repository evidence

- Reviews
- Reflections
- Instructor observations

The implementation becomes a living educational system.

Phase 6: Learn And Improve The implementation reflects on:

- What worked
- What struggled
- What should change
- What may become reusable

The implementation contributes back to ETIS.

The Adoption Pyramid

Every ETIS implementation consists of three layers.

Local Implementation ↑ Reusable Educational Assets ↑ Stable ETIS Foundation

Stable ETIS Foundation Changes rarely.

Examples:

- ETIS book
- Educational architecture
- Learning models

Reusable Educational Assets Changes occasionally.

Examples:

- Playbooks
- Templates
- Workflows
- Assessments

Local Implementation Changes frequently.

Examples:

- Schedules
- Assignments
- Slides
- Instructor notes

This separation creates long-term maintainability.

The Flagship Implementation

The current flagship implementation is:

adoption_examples/ ‘— loyola_comp330/

COMP330 serves as the first implementation.

It is important because it provides:

- Real classroom experience
- Real student teams
- Real repositories
- Real assessments
- Real engineering challenges

However, COMP330 is not the ETIS Educational Ecosystem itself.

It is one implementation.

Provenance Matters

Every implementation should preserve provenance.

The following should be identifiable:

- Institution
- Course or program
- Audience
- Instructor or steward
- Relationship to ETIS
- Relationship to shared assets
- Local adaptations

Provenance preserves educational memory.

Promotion Model

Implementations will create new intellectual property.

Not all of it should become reusable ETIS assets.

The promotion path should be:

Implementation asset

↓

Demonstrated value

↓

Review for reuse

↓

Normalize to Markdown

↓

Preserve provenance

↓

Promote to shared_assets

Promotion should be deliberate.

What Success Looks Like

A successful ETIS adoption should create learners who can:

- Define intent

- Engineer context
- Bound authority
- Verify behavior
- Operate reality
- Explain decisions
- Own outcomes

The implementation itself should also become understandable by future adopters.

Future instructors should be able to answer:

- What was taught?
- Why was it taught?
- How was it taught?
- What evidence was produced?
- What was learned?

If those questions can be answered, the adoption is preserving educational memory.

Future Evolution

ETIS may eventually support many implementations.

Examples include:

- Undergraduate software engineering
- Graduate software engineering
- Capstone programs
- AI governance workshops
- Professional development
- Organizational training

Growth should occur through governed adoption rather than uncontrolled expansion.

Each implementation should contribute to the ecosystem without redefining it.

Core Thesis

ETIS adoption is not about copying a course.

It is about intentionally assembling a trustworthy engineering learning experience.

The framework remains stable.

The implementations adapt.

The learners transform.

That balance between stability and adaptation is what makes the ETIS Educational Ecosystem sustainable.

Asset Consumption Model

Purpose

This document defines how educational implementations consume assets within the ETIS (Engineering Trustworthy Intelligent Systems) Educational Ecosystem.

The purpose of this model is to establish clear relationships between ETIS educational components while preventing duplication, fragmentation, and architectural drift.

Educational implementations should consume ETIS assets rather than recreate them.

This document explains how that consumption occurs.

Core Philosophy

ETIS is an ecosystem.

Educational ecosystems are assembled from interconnected assets.

Not every implementation should create its own materials from scratch.

Instead, implementations should intentionally consume, adapt, and contribute educational assets while preserving ownership boundaries.

The objective is sustainable educational reuse.

Core Principle

Educational implementations should consume ETIS assets rather than own them.

The implementation assembles a learning experience.

The ecosystem supplies the building blocks.

This distinction is critical.

Asset Consumption Architecture

All ETIS implementations should follow the same high-level consumption model.

ETIS Book

↓

Educational Architecture

↓

Learning Models

↓

Shared Assets

↓

Student Starter Kits

↓

Educational Implementations

↓

Each layer consumes the layers above it.

The relationship is intentionally directional.

Educational Layers

The ETIS Educational Ecosystem contains six layers.

Layer 1: ETIS Book The ETIS book is the authoritative engineering framework.

The book defines:

- Principles
- Engineering philosophy
- Engineering methods
- Governance concepts
- Operational thinking
- Stewardship concepts

The book is authoritative.

Educational assets adapt the book.

Educational assets do not replace the book.

Layer 2: Educational Architecture Educational architecture defines how ETIS is taught.

Examples include:

educational_architecture/

README.md

ETIS_Educational_Architecture.md

educational_design_principles.md

audience_learning_paths.md

39_chapter_learning_map.md

Educational architecture provides teaching structure.

Educational implementations consume this architecture.

Layer 3: Learning Models Learning models define learner development.

Examples include:

learning_models/

ETIS_Learning_Models.md

professional_transformation_model.md

engineering_maturity_model.md

software_engineering_learning_progression.md

two-cycle_engineering_model/

Learning models explain how learners mature.

Implementations consume these models.

Layer 4: Shared Assets Shared assets provide reusable educational intellectual property.

Examples include:

shared ETIS educational assets

playbooks/

templates/

workflows/

ai_engineering/

assessment/

Shared assets are intentionally reusable.

They are consumed by implementations.

Layer 5: Student Starter Kits Student Starter Kits provide assembled engineering environments.

Examples include:

student starter kit

comp330/

Starter kits are educational products.

They are consumed by implementations.

Layer 6: Educational Implementations Educational implementations assemble the learner experience.

Examples include:

adoption_examples/

loyola_comp330/

Implementations orchestrate educational assets.

They do not redefine them.

Asset Ownership Model

Each educational component has ownership responsibilities.

Component	Owns
ETIS Book	Engineering framework
Educational Architecture	Teaching architecture
Learning Models	Learner development
Shared Assets	Reusable educational intellectual property
Student Starter Kits	Structured practice environments
Adoption Examples	Local implementation context

Ownership should remain clear.

Consumption Is Not Copying

The following patterns should be avoided.

Poor Pattern COMP330 creates its own requirements template.

Another course creates its own requirements template.

A third course creates another requirements template.

This creates duplication.

Healthy Pattern

shared_assets/templates/

↓

Consumed by COMP330

↓

Consumed by Course B

↓

Consumed by Course C

One reusable asset.

Multiple consumers.

The Consumption Rule

When creating a new educational artifact, implementations should ask:

Does this already exist somewhere else?

If yes:

Consume it.

If no:

Create it locally.

Then ask:

Can another implementation use this without significant modification?

If yes:

Consider promoting it later.

If no:

Keep it local.

Educational Asset Lifecycle

Assets should move through a deliberate lifecycle.

Implementation Need

↓

Local Asset Creation

↓

Demonstrated Value

↓

Review For Reuse

↓

Markdown Normalization

↓

Provenance Preservation

↓

Shared Asset Promotion

This lifecycle prevents uncontrolled growth.

Promotion Criteria

An implementation asset should generally satisfy several criteria before promotion.

Criterion 1: Reuse Potential Can multiple implementations use it?

Criterion 2: Minimal Local Assumptions Does it depend heavily on one institution?

Criterion 3: Stable Purpose Is its educational purpose well understood?

Criterion 4: Clear Ownership Can future adopters understand where it came from?

Criterion 5: Markdown Authority Can it become a Markdown-native ETIS asset?

Only then should promotion occur.

Loyola COMP330 Consumption Example

The current flagship implementation demonstrates the model.

ETIS Book

↓

Educational Architecture

↓

Learning Models

↓

Shared Assets

↓

Student Starter Kit

↓

Loyola COMP330

COMP330 assembles these assets into a semester-long learning experience.

COMP330 should not duplicate them.

Example Consumption Scenario

A future COMP330 implementation might consume:

Educational Architecture audience_learning_paths.md

39_chapter_learning_map.md

Learning Models professional_transformation_model.md

engineering_maturity_model.md

software_engineering_learning_progression.md

two-cycle_engineering_model/

Shared Assets requirements_review_playbook.md

risk_register_template.md

ai_use_log_template.md

release_readiness_workflow.md

Student Starter Kit student starter kitcomp330/

The implementation then adds local materials.

Examples:

Fall 2027 syllabus

Weekly schedule

Assignment due dates

Class slides

Instructor announcements

This separation is intentional.

Local Assets

Some assets should remain local.

Examples include:

- Syllabi
- Semester schedules
- Assignment due dates

- Classroom slides
- Instructor observations
- Semester-specific announcements

These assets support the implementation.

They are not educational infrastructure.

Instructor Tools

Instructor tools often begin locally.

Examples include:

- Repository analysis scripts
- Assessment automation
- Reporting tools
- Dashboard generation

These tools should initially remain inside the implementation.

Example:

adoption_examples/ `--- loyola_comp330/`— instructor_tools/

If multiple implementations can use them without major changes, they may later be promoted elsewhere.

Provenance Rules

Every asset should preserve provenance.

Assets should identify:

- Origin institution
- Origin implementation
- Educational purpose
- Relationship to ETIS
- Relationship to reusable assets

Provenance preserves educational memory.

Anti-Patterns

The following anti-patterns should be actively avoided.

Asset Duplication Creating multiple versions of the same asset.

Asset Hoarding Keeping reusable assets inside implementations.

Premature Generalization Promoting assets before reuse is demonstrated.

Hidden Dependencies Depending on undocumented materials.

Institutional Erasure Removing provenance too early.

Framework Drift Allowing implementations to redefine ETIS.

These anti-patterns increase complexity and reduce maintainability.

Relationship to Implementation Governance

This document explains **how assets move through the ecosystem**.

`implementation_governance.md` explains **how implementations remain healthy over time**.

The two documents are complementary.

Future Evolution

This model should remain relatively stable.

New educational implementations may appear.

New shared assets may emerge.

New student starter kits may be created.

However, the underlying consumption relationships should remain simple.

Growth should increase capability, not complexity.

Core Thesis

Educational implementations should assemble ETIS rather than recreate ETIS.

The ecosystem supplies reusable educational infrastructure.

Implementations consume that infrastructure and create local learning experiences.

That separation between reusable assets and local implementation is what allows ETIS to grow without fragmenting.

Implementation Governance

Purpose

This document defines how ETIS (Engineering Trustworthy Intelligent Systems) educational implementations are governed over time.

The purpose of implementation governance is to preserve alignment between local educational implementations and the broader ETIS Educational Ecosystem.

Educational implementations are intentionally adaptable.

However, adaptability should not become fragmentation.

Implementation governance provides a framework for allowing local flexibility while preserving the integrity of ETIS.

Core Philosophy

Educational implementations should evolve.

They should not drift.

Evolution is intentional.

Drift is accidental.

Implementation governance exists to help educational adopters make intentional decisions about what should remain stable, what may change locally, and what may eventually become reusable across the ETIS ecosystem.

The goal is sustainable adaptation.

Governance Principle

ETIS is one educational ecosystem with many implementations.

The framework remains stable.

Implementations adapt to local context.

The framework should not be redefined every time a new implementation appears.

Instead, implementations should consume, apply, evaluate, and contribute back to ETIS.

Governance Objectives

Implementation governance has five objectives.

Preserve educational alignment Ensure implementations remain connected to the ETIS philosophy and educational goals.

Preserve ownership boundaries Ensure reusable educational assets do not become trapped inside local implementations.

Preserve provenance Ensure future adopters understand where educational assets originated.

Preserve educational memory Ensure implementations remain understandable over time.

Support responsible evolution Ensure educational improvements can be incorporated without destabilizing ETIS.

Governance Architecture

Implementation governance operates across three layers.

Stable ETIS Foundation

↓

Reusable Educational Assets

↓

Local Educational Implementations

Each layer changes at a different rate.

Layer 1: Stable ETIS Foundation

This layer changes slowly.

It provides long-term stability.

Examples include:

- ETIS book
- Educational architecture
- Learning models

This layer should remain relatively stable.

Changes should occur infrequently and intentionally.

Layer 2: Reusable Educational Assets

This layer changes occasionally.

Examples include:

- Playbooks
- Templates
- Workflows
- AI Engineering assets
- Assessment assets

This layer grows as educational experience accumulates.

Assets should only enter this layer after demonstrating reuse potential.

Layer 3: Local Educational Implementations

This layer changes frequently.

Examples include:

- Course schedules
- Assignments
- Rubrics
- Slides
- Classroom exercises

- Instructor notes
- Local analysis tools

Local adaptation is expected at this layer.

Governance Questions

Every implementation should regularly answer the following questions.

Is this still aligned with ETIS?

Is this implementation-specific or reusable?

Are we preserving provenance?

Is educational evidence understandable to future adopters?

Are we creating unnecessary duplication? These questions help implementations remain healthy over time.

What Implementations May Change

Implementations may adapt:

- Reading emphasis
- Chapter sequencing
- Weekly schedules
- Assignment structure
- Classroom exercises
- Project scope
- Rubrics
- Local tooling
- Instructor notes

These changes allow implementations to fit local educational environments.

What Implementations Should Not Change

Implementations should not alter the foundational ETIS philosophy.

Core expectations should remain stable.

Examples include:

- Evidence-centered engineering
- Repository-centered engineering
- Human accountability
- Responsible AI usage
- Review culture
- Operational awareness
- Long-term stewardship

These principles should remain visible regardless of implementation.

Provenance Requirements

Every implementation should preserve provenance.

At minimum, implementations should identify:

Institution Example:

Loyola University Chicago

Course or program Example:

COMP 330 — Software Engineering

Audience Examples:

- Undergraduate students
- Graduate students
- Professional learners

Relationship to ETIS Implementations should identify which ETIS assets are being consumed.

Local adaptations Implementations should document important local decisions.

Provenance preserves educational memory.

Local Adaptation Documentation

Implementations should make local adaptations visible.

Future adopters should be able to understand:

- What changed
- Why it changed
- What problem it solved
- Whether the adaptation may be useful elsewhere

Adaptations should not remain hidden inside slides, emails, or instructor memory.

Asset Ownership Boundaries

Clear ownership boundaries reduce educational drift.

Educational Architecture Defines how ETIS is taught.

Learning Models Define how learners develop.

Shared Assets Provide reusable educational intellectual property.

Student Starter Kits Provide assembled practice environments.

Adoption Examples Provide implementation context.

Each implementation should respect these boundaries.

Reuse Promotion Model

Implementations will create new assets.

Not every asset should become part of ETIS.

An asset should generally remain local until reuse is demonstrated.

Promotion path:

Local Asset

↓

Demonstrated Value

↓

Review For Reuse

↓

Normalize To Markdown

↓

Preserve Provenance

↓

Promote To Shared Assets

Promotion should be intentional.

Instructor Tool Governance

Educational implementations may create instructor-specific tooling.

Examples include:

- Repository analysis scripts
- Assessment automation
- Reporting scripts
- Dashboard generation
- Review support tools

These tools should initially remain implementation-specific.

For example:

adoption_examples/ `--- loyola_comp330/`— instructor_tools/

If multiple implementations can use the tools without significant modification, they may eventually be promoted elsewhere.

Educational Memory Preservation

Implementations should preserve educational memory.

Future adopters should be able to answer:

- What was taught?
- Why was it taught?
- What evidence was expected?

- What tools were used?
- What worked well?
- What changed over time?

Educational memory should not depend on a single instructor.

Growth Without Fragmentation

Growth is expected.

Fragmentation is not.

Healthy growth looks like:

Shared Foundation

↓

Multiple Implementations

↓

Shared Learning

↓

Improved Educational Assets

Unhealthy growth looks like:

Multiple Implementations

↓

Multiple Frameworks

↓

Duplicated Assets

↓

Conflicting Philosophies

The goal is a single ecosystem with many implementations.

Relationship to Loyola COMP330

The current flagship implementation is:

adoption_examples/ ‘— loyola_comp330/

COMP330 provides the initial implementation experience for ETIS.

Its role is to:

- Demonstrate ETIS adoption
- Generate educational evidence
- Generate reusable assets
- Preserve Loyola provenance
- Inform future implementations

COMP330 is an implementation, not the framework itself.

Future Evolution

Over time, ETIS may support many implementations.

Examples include:

- Undergraduate courses
- Graduate courses
- Capstone programs
- Professional workshops
- AI governance programs
- Organizational training programs

Implementation governance should scale with that growth without increasing complexity unnecessarily.

The framework should remain stable while implementations diversify.

Core Thesis

ETIS implementations should be free to adapt without becoming disconnected.

Implementation governance exists to preserve that balance.

The framework remains stable.

The implementations evolve.

The ecosystem learns.

Part IV

Adoption and Stewardship

ETIS Adoption Planning Guide

The **ETIS Adoption Planning Guide** helps instructors, course designers, and educational leaders decide how to adopt *Engineering Trustworthy Intelligent Systems (ETIS)* within a specific course, program, institution, or professional training environment.

Adoption is not file copying.

Adoption is the responsible translation of ETIS doctrine into a teachable implementation while preserving educational purpose, architectural boundaries, provenance, evidence expectations, and long-term stewardship.

Purpose

The purpose of this guide is to help instructors plan an ETIS adoption before building course materials.

A successful ETIS adoption should be intentional, bounded, teachable, assessable, and maintainable.

This guide helps instructors decide:

- what kind of ETIS adoption is appropriate,
- which ETIS learning path should be emphasized,
- which parts of the ETIS book are primary,
- which shared assets should be consumed,
- which student starter kit should be used,
- which assignments should function as engineering phase gates,
- how AI use will be governed,
- how repository evidence will be evaluated,
- how release readiness will be demonstrated,
- and how the adoption will preserve alignment with ETIS over time.

Adoption Principle

ETIS adoption begins with one governing principle:

> Adopt ETIS to form trustworthy engineers, not to add more course documents.

An ETIS course should not become a paperwork exercise.

ETIS adoption also follows these principles:

> Institutions should inherit ETIS doctrine, not ETIS implementations.
>
> Standardize outcomes. Localize execution.

Local context is not architectural drift.

Institutions should preserve ETIS philosophy while adapting implementation to their own environments.

The purpose is to help learners practice disciplined engineering judgment through intent, context, evidence, review, governance, operation, and accountability.

Relationship to the ETIS Book

The ETIS book remains authoritative.

An adoption may emphasize different parts of the book depending on audience, course length, and learning goals, but the adoption should not contradict the book's doctrine.

> ETIS Book

↓

Educational Architecture

↓

Instructor Course Package

↓

Course Implementation

↓

Student Engineering Practice

The course implementation adapts ETIS.

It does not replace ETIS.

Educational Engine Architecture

The Instructor Course Package now consists of six educational engines.

Educational Intent Engine

↓

Educational Sequencing Engine

↓

Educational Accountability Engine

↓

Educational Evaluation Engine

↓

Educational Operations Engine

↓

Educational Stewardship Engine

Institutional adoptions should consume these engines as an integrated educational system rather than as isolated directories.

Together they create a complete educational operating model.

Adoption Is Not Implementation Copying

The Loyola COMP 330 implementation is the flagship adoption example.

It is a reference model, not a universal template.

Instructors should study COMP 330 to understand how ETIS can be taught through repository-centered engineering, phase gates, AI-use governance, team accountability, review, release readiness, and final engineering defense.

However, adopters should not copy COMP 330 blindly.

Every course has its own constraints:

- audience,

- semester length,
- student maturity,
- class size,
- institutional policies,
- project expectations,
- technical prerequisites,
- assessment rules,
- calendar structure,
- available tools,
- instructor capacity,
- and local educational goals.

A good adoption preserves ETIS doctrine while adapting implementation details responsibly.

The governing principle should now be:

> Institutions should inherit ETIS doctrine, not ETIS implementations.

Standardize outcomes.

Localize execution.

Local context is not architectural drift.

The educational outcomes should remain stable while implementation mechanisms adapt to institutional realities.

Adoption Planning Questions

Before creating course materials, the instructor should answer the following questions.

1. What is the adoption context? Identify the instructional environment.

Examples include:

- undergraduate software engineering course,
- graduate software engineering course,
- senior capstone,
- AI governance module,
- software architecture module,
- DevOps or operational readiness module,
- professional training program,
- corporate engineering workshop,
- or organizational governance training.

The adoption context determines scope, depth, pacing, assessment, and artifact expectations.

2. What learner transformation is expected? Define the professional growth expected from learners.

Possible transformation goals include:

- student to responsible engineer,
- developer to reviewer,
- reviewer to architect,
- team member to release defender,
- builder to operational thinker,
- or practitioner to trustworthy engineering steward.

The adoption should be designed around transformation, not merely content coverage.

3. Which ETIS capabilities are primary? Select the primary capabilities the adoption will teach.

Examples include:

- requirements discipline,
- repository-centered engineering,
- team coordination,
- architecture reasoning,
- decision documentation,
- AI-use governance,
- implementation under review,
- testing and validation,
- release readiness,
- observability,
- operational readiness,
- incident response,
- postmortems,
- security governance,
- review boards,
- and stewardship.

A course does not need to teach everything equally.

A strong adoption is focused enough to be teachable.

4. Which parts of the ETIS book are primary? Map the adoption to the book.

A full-semester undergraduate software engineering course may emphasize Parts I and II while selectively introducing Parts III and IV.

A graduate architecture or governance course may place more emphasis on Parts III and IV.

A professional training module may focus on a small number of chapters tied to a specific engineering capability.

The adoption should make explicit which chapters are required, which are supporting, and which are optional.

5. What evidence will students produce? ETIS learning is repository-centered.

Students should produce evidence that another reviewer can inspect.

Evidence may include:

- requirements,
- use cases,
- planning records,
- risk registers,
- architecture notes,
- ADRs,
- AI-use logs,
- pull request reviews,
- test evidence,
- defect records,

- release readiness records,
- postmortems,
- operational readiness evidence,
- governance records,
- and final defense materials.

The adoption should define the evidence standard before assignments begin.

6. What phase gates will structure the course? ETIS assignments should function as engineering phase gates.

Possible phase gates include:

1. repository foundation,
2. requirements and planning review,
3. architecture review,
4. implementation and validation review,
5. Cycle 1 release readiness,
6. postmortem and stabilization review,
7. operational readiness review,
8. final release defense.

Each phase gate should increase system maturity and student accountability.

7. How will AI use be governed? An ETIS adoption should define AI-use expectations clearly.

The course should specify:

- what AI use is allowed,
- what AI use is prohibited,
- what AI use must be disclosed,
- how AI output must be reviewed,
- where AI-use evidence is recorded,
- how students verify AI-assisted work,
- and how responsibility remains with the human team.

The governing principle remains:

> AI-assisted work is not accepted engineering work until humans review, verify, and own it.

8. How will assessment work? Assessment should evaluate both product and evidence.

A working demo is not enough.

The instructor should define how students will be assessed on:

- engineering intent,
- evidence quality,
- technical execution,
- review participation,
- AI-use accountability,
- testing discipline,
- architecture reasoning,
- risk visibility,
- release readiness,
- operational thinking,

- and professional defense.

Rubrics should reward disciplined engineering behavior, not only visible functionality.

9. What starter environment will students use? The adoption should decide whether students will use an ETIS Student Starter Kit.

A starter kit should provide a structured repository environment that supports evidence creation, team coordination, AI-use governance, reviews, testing, and release readiness.

The starter kit should be introduced as an engineering environment, not as a folder full of templates.

10. How will the adoption be maintained? An ETIS adoption should be maintainable beyond one semester.

The instructor should decide:

- what materials are local to the course,
- what materials are reusable,
- what improvements should be proposed upstream,
- what evidence should be preserved,
- what lessons learned should be recorded,
- and how the next offering will improve without drifting from ETIS doctrine.

11. How will educational stewardship be preserved? Educational systems should become smarter every semester.

Institutions should define:

- how lessons learned will be captured,
- how student patterns will be documented,
- how course corrections will be preserved,
- how engineering maturity signals will be observed,
- and how future instructors will inherit institutional memory.

Educational memory is educational infrastructure.

Educational systems should improve over time rather than restart every semester.

Adoption Models

ETIS can be adopted at different scales.

Full Adoption ETIS becomes the primary framework for the course.

This model is appropriate for a full-semester software engineering course, capstone course, or graduate engineering course.

A full adoption usually includes:

- repository-centered project work,
- team-based engineering,
- staged phase gates,
- AI-use governance,
- architecture and review,
- two-cycle delivery,
- release readiness,

- operational maturity,
- and final engineering defense.

Partial Adoption ETIS provides selected modules, practices, or assessments within an existing course.

This model is appropriate when the instructor wants to introduce ETIS capabilities without redesigning the full course.

Examples include:

- AI-use governance module,
- release readiness module,
- architecture review module,
- operational readiness module,
- repository evidence module,
- or postmortem module.

Capstone Adoption ETIS provides structure for student teams building larger systems.

This model emphasizes:

- intent,
- team accountability,
- evidence,
- architecture,
- review,
- release readiness,
- stakeholder communication,
- operational maturity,
- and final defense.

Professional Adoption ETIS supports professional or organizational training.

This model may emphasize:

- AI-assisted engineering governance,
- review boards,
- release governance,
- operational trust,
- system stewardship,
- incident learning,
- and engineering accountability.

Recommended Adoption Sequence

A disciplined adoption should proceed in stages.

Step 1 — Define the Adoption Boundary State what the course will and will not attempt to teach.

Avoid trying to cover all ETIS capabilities at equal depth.

Step 2 — Select the Learning Path Choose the professional transformation path most appropriate for the learners.

Examples:

- student to responsible engineer,
- student to release defender,
- engineer to reviewer,
- engineer to operational thinker,
- or practitioner to trustworthy system steward.

Step 3 — Map the Book Identify required, supporting, and optional ETIS chapters.

This prevents the course from becoming either too broad or disconnected from the authoritative text.

Step 4 — Select Shared Assets Choose reusable assets from the ETIS educational ecosystem.

Do not copy or modify shared assets unnecessarily.

Consume them intentionally.

Step 5 — Select or Adapt the Student Starter Kit Decide what repository structure students will use.

Preserve repository-centered teaching.

Step 6 — Design Phase Gates Define the assignment sequence as maturity gates.

Each gate should produce reviewable evidence.

Step 7 — Define Assessment Create or select rubrics that evaluate product, process, evidence, review, AI governance, and release defensibility.

Step 8 — Plan Facilitation Decide how class time will support reviews, critiques, simulations, tabletops, workshops, and defense preparation.

Step 9 — Preserve Provenance Record where materials came from.

Do not erase Loyola provenance when using assets derived from COMP 330.

Step 10 — Capture Lessons Learned After the adoption, preserve what worked, what failed, what confused students, what improved learning, and what should change next time.

Step 11 — Contribute Stewardship Memory Determine what educational wisdom should flow back into the ecosystem.

Examples include:

- common student patterns,
- AI transition observations,
- maturity signals,
- institutional adaptations,
- course corrections,
- and reusable instructor guidance.

Every implementation should leave the system smarter than it found it.

Adoption Planning Template

Instructors may use the following planning structure.

Course or Program Name:

Institution or Organization:

Adoption Type: Full / Partial / Capstone / Professional / Module

Primary Audience:

Course Length:

Primary ETIS Capabilities:

Required ETIS Chapters:

Supporting ETIS Chapters:

Optional ETIS Chapters:

Student Starter Kit:

Shared Assets Consumed:

Major Phase Gates:

AI-Use Governance Model:

Assessment Model:

Repository Evidence Expectations:

Final Defense or Capstone Evidence:

Local Adaptations:

Materials Created Locally:

Materials Proposed for Upstream Contribution:

Known Risks:

Maintenance Plan:

This template should be completed before major course materials are finalized.

Common Adoption Mistakes

Mistake 1 — Treating ETIS as a Topic List ETIS is not a checklist of subjects.

It is an engineering formation model.

Mistake 2 — Copying COMP 330 Without Adaptation COMP 330 is a flagship implementation.

It is not a universal course shell.

Mistake 3 — Assigning Documents Without Engineering Purpose Every artifact should teach responsibility, evidence, judgment, review, risk awareness, or professional accountability.

Mistake 4 — Overloading Students With Process ETIS should create disciplined engineering behavior, not paperwork fatigue.

Mistake 5 — Ignoring AI Governance AI use must be taught as accountable engineering practice.

Mistake 6 — Assessing Only the Demo A demo shows that something appears to work.

It does not prove that the system is understandable, reviewable, governable, operable, improvable, or trustworthy.

Mistake 7 — Erasing Provenance If an asset originated in Loyola COMP 330, that provenance should remain visible.

Educational history is part of engineering memory.

Mistake 8 — Treating ETIS As A Loyola Product Loyola COMP330 is the flagship implementation.

It is not the educational standard.

Institutions should inherit ETIS doctrine, not copy Loyola.

Adoption Readiness Checklist

Before launching an ETIS adoption, the instructor should confirm:

- The adoption type is clear.
- The learning transformation goal is explicit.
- Required ETIS chapters are identified.
- Shared assets have been selected.
- Student repository expectations are defined.
- Assignment phase gates are sequenced.
- AI-use rules are documented.
- Assessment criteria are clear.
- Release readiness expectations are defined.
- Final defense expectations are defined.
- Institutional constraints have been considered.
- Local adaptations are documented.
- Provenance is preserved.
- Maintenance responsibilities are understood.

If these items are unclear, the adoption is not ready.

Stewardship Expectations

Every ETIS adoption should leave evidence.

After the course or training experience, instructors should preserve:

- what was taught,
- what assets were used,
- what adaptations were made,
- what students struggled with,
- what evidence quality looked like,
- what assessment criteria worked,
- what AI-use issues emerged,
- what should improve next time,
- and what might be reusable by future adopters.

Adoption is not complete when the course ends.

Adoption is complete when learning from the course has been captured for future stewardship.

Every semester should leave the system smarter than it found it.

Educational memory is educational infrastructure.

Educational stewardship should become part of normal course operations rather than an afterthought performed after the semester ends.

Institutions should preserve educational memory as part of adoption.

Educational memory may include:

- common student patterns,
- maturity signals,
- AI transition observations,
- difficult conversations,
- course corrections,
- and institutional adaptations.

Educational stewardship should become part of normal operations.

Promotion Back Into ETIS

Not every local course artifact should become a shared ETIS asset.

Promotion should occur only when an artifact demonstrates reuse value beyond its original course.

An artifact may be considered for promotion when it:

- supports more than one implementation,
- strengthens ETIS teaching capability,
- preserves clear provenance,
- has a defined consumer,
- has a stable purpose,
- avoids unnecessary local assumptions,
- and can be maintained over time.

Promotion requires review.

Premature generalization should be avoided.

Final Standard

An ETIS adoption is successful when students do more than complete assignments.

A successful adoption produces learners who can explain:

- what they intended,
- what they built,
- how they used AI,
- what they verified,
- what evidence supports their claims,
- what risks remain,
- why their release is defensible,
- how the system could be operated,
- and what they would improve next.

The goal is not adoption for its own sake.

The goal is the formation of trustworthy engineers.

Successful ETIS adoptions should simultaneously produce three outcomes:

- trustworthy engineers,
- reusable educational assets,
- and educational knowledge for future instructors.

Educational systems are engineered.

Educational systems are also stewarded.

ETIS Course Readiness Checklist

The **ETIS Course Readiness Checklist** is a pre-launch review document for instructors preparing to teach an ETIS-based course.

Its purpose is to ensure that course design decisions have been made before students begin major engineering work.

This checklist is not administrative paperwork.

It is a quality assurance review for educational systems.

A course is considered ready when an instructor can confidently answer the questions in this document.

Purpose

This checklist helps instructors verify that an ETIS course has been intentionally designed.

The checklist validates that:

- educational goals are clear,
- professional transformation goals are defined,
- ETIS book coverage is intentional,
- repositories are prepared,
- assignments are sequenced,
- AI governance is established,
- assessments are designed,
- and engineering accountability is visible.

The checklist should be completed before the first major student deliverable.

Course Information

Complete the following information.

Course Name:

Institution:

Department:

Instructor(s):

Term:

Course Format: Semester / Quarter / Module / Professional Training

Delivery Model: In-Person / Hybrid / Online

Primary Audience:

Expected Enrollment:

Project Type: Individual / Team-Based / Mixed

Section 1 — Professional Transformation

The course should be designed around who students will become.

Verify

[ ] The primary professional transformation goal has been defined.

[ ] The transformation has been communicated to students.

[ ] The course is designed around engineering maturity rather than topic coverage.

Primary Transformation Goal

Student

↓

_____

Examples:

Student → Responsible Engineer

Student → Reviewer

Student → Release Defender

Student → Operational Thinker

Student → Future Trustworthy Engineer

Section 2 — ETIS Book Alignment

The ETIS book remains authoritative.

Verify

[ ] Required chapters have been identified.

[ ] Supporting chapters have been identified.

[ ] Optional chapters have been identified.

[ ] Book coverage matches course scope.

Record

Required Chapters:

Supporting Chapters:

Optional Chapters:

Section 3 — Engineering Capabilities

Select the primary capabilities students will practice.

Verify

[ ] Capabilities have been intentionally selected.

[ ] The course is not attempting to teach everything equally.

Primary Capabilities

Check all that apply.

[ ] Requirements Discipline

[ ] Planning

[ ] Estimation

[ ] Risk Management

[ ] Repository-Centered Engineering

[ ] Architecture Reasoning

[ ] Decision Documentation

[ ] AI Governance

[ ] Pull Requests and Reviews

[ ] Testing and Validation

[ ] Release Readiness

[ ] Operational Thinking

[ ] Observability

[ ] Security Governance

[ ] Incident Response

[ ] Postmortems

[ ] Stewardship

Section 4 — Repository Readiness

Students should receive a prepared engineering environment.

Verify

[ ] A Student Starter Kit has been selected.

[ ] Repository expectations are documented.

[ ] Evidence locations are defined.

[ ] Students understand that repositories are evidence systems.

Repository Areas

Verify expectations have been defined for:

[ ] Requirements

[ ] Planning

[ ] Architecture

[ ] Decisions

[ ] Reviews

[ ] Testing

[ ] Quality

[ ] Release

[ ] Operations

[ ] Governance

[ ] AI

[ ] Postmortems

Section 5 — Engineering Phase Gates

Assignments should function as maturity gates.

Verify

[ ] Phase gates have been defined.

[ ] Each phase gate increases engineering maturity.

[ ] Each phase gate produces evidence.

Planned Phase Gates

Phase Gate 1:

Phase Gate 2:

Phase Gate 3:

Phase Gate 4:

Phase Gate 5:

Phase Gate 6:

Phase Gate 7:

Phase Gate 8:

Section 6 — Two-Cycle Design

Verify

[ ] Cycle 1 activities have been defined.

[ ] Cycle 2 activities have been defined.

[ ] Students understand the difference.

Record

Cycle 1 — Can It Work? Primary Activities:

Cycle 2 — Can It Survive? Primary Activities:

Section 7 — AI Governance Readiness

AI use should be intentionally governed.

Verify

[ ] AI expectations are documented.

[ ] AI disclosure requirements are documented.

[ ] AI review expectations are documented.

[ ] AI ownership expectations are documented.

Record

Allowed AI Use:

Restricted AI Use:

Required Disclosures:

AI Evidence Location:

Section 8 — Classroom Experience Readiness

Students should actively practice engineering.

Verify

Check planned activities.

[ ] Requirements Review

[ ] Ambiguity Workshop

[ ] Architecture Critique

[ ] ADR Review

[ ] AI-Use Review

[ ] Pull Request Simulation

[ ] Code Review Workshop

[ ] Testing Evidence Review

[ ] Release Readiness Tabletop

[ ] Incident Response Tabletop

[ ] Postmortem Analysis

[ ] Final Defense Preparation

Section 9 — Assessment Readiness

Assessment should evaluate both product and evidence.

Verify

Assessment criteria exist for:

[ ] Engineering Intent

[ ] Evidence Quality

[ ] Technical Execution

[ ] Architecture Reasoning

[ ] Review Participation

[ ] AI Accountability

[ ] Testing Discipline

[ ] Risk Communication

[ ] Release Readiness

[ ] Operational Thinking

[ ] Professional Accountability

Section 10 — Final Defense Readiness

Every ETIS course should culminate in accountability.

Verify

Students will answer questions such as:

[ ] What problem were you solving?

[ ] Why was this architecture selected?

[ ] How did AI assist?

[ ] What evidence supports your claims?

[ ] What risks remain?

[ ] How would this system be operated?

[ ] What would you improve?

[ ] Why is this release defensible?

Section 11 — Local Adaptation Review

Every implementation will have local constraints.

Record

Institutional Constraints:

Calendar Constraints:

Technology Constraints:

Assessment Constraints:

Local Adaptations:

Section 12 — Stewardship Review

The course should be maintainable over time.

Verify

[ ] Reusable assets are separated from local assets.

[ ] Loyola provenance has been preserved.

[ ] Shared assets have not been duplicated.

[ ] COMP330 has not become the ecosystem.

[ ] Lessons learned will be captured after the course.

Final Readiness Decision

Complete this review before launching the course.

[ ] Ready To Launch

[ ] Mostly Ready

[ ] Significant Work Needed

Instructor Confidence Review

An instructor should be able to answer these questions confidently.

What engineering responsibilities are students learning?

What evidence will students create?

How will AI be governed?

How will engineering maturity increase?

How will students practice engineering?

How will students defend engineering decisions?

What professional habits should students carry forward?

If these answers are unclear, the course is not ready.

Guiding Principle

The purpose of ETIS is not to create more assignments.

The purpose of ETIS is to create engineers who can define intent, engineer context, bound authority, verify behavior, operate reality, explain decisions, and own outcomes.

This checklist exists to ensure the course itself is engineered responsibly before students are asked to engineer responsibly.

Educational Laboratory Insights

Purpose

This document captures the educational insights discovered through the ongoing implementation of ETIS (Engineering Trustworthy Intelligent Systems) within Loyola University Chicago COMP330/474.

These insights were discovered through real classroom experiences.

They represent observations about students, engineering behaviors, AI adoption, repository-centered engineering, accountability systems, and long-term educational outcomes.

The purpose is not to document mistakes.

The purpose is to preserve educational engineering memory.

These insights continuously contribute back to ETIS itself.

Core Philosophy

Educational laboratories are where educational frameworks become trustworthy.

Educational theory becomes stronger when continuously challenged by educational reality.

This implementation intentionally creates a bi-directional feedback loop.

ETIS

↓

COMP330/474

↓

Educational Evidence

↓

ETIS Refinement

↓

Future Implementations

Educational experiences should continuously improve educational frameworks.

Insight 1: Educational Work Should Resemble Professional Engineering Work

One of the earliest discoveries was that students respond positively when educational activities resemble professional engineering activities.

Students become more engaged when they understand that they are practicing real engineering behaviors rather than completing isolated academic exercises.

Professional engineering environments should not be introduced after graduation.

They should be introduced during education.

This insight significantly influenced ETIS.

Insight 2: Repository-Centered Engineering Creates Continuity

Repository-centered engineering proved highly effective.

Students quickly learn that repositories preserve more than source code.

Repositories preserve engineering memory.

Students begin seeing repositories as systems that preserve:

- requirements
- plans
- architecture
- AI usage
- reviews
- testing evidence
- release evidence
- operational thinking

This creates continuity between education and professional practice.

This insight became foundational to ETIS.

Insight 3: Accountability Should Be Distributed

Students naturally procrastinate when accountability is concentrated at the end of the semester.

Distributed accountability produces stronger engineering outcomes.

The six engineering phase gates intentionally address this challenge.

Smaller checkpoints create:

- continuous progress
- earlier intervention opportunities
- better visibility
- stronger engineering discipline

Engineering accountability should be continuous rather than episodic.

This insight significantly influenced ETIS.

Insight 4: Teams Need Early Intervention

Many teams naturally experience participation imbalances.

Most semesters include at least one team member who is not fully engaged.

These situations should be identified early.

Problems rarely resolve themselves without intervention.

Early visibility creates better outcomes.

Repository evidence and phase gates help surface these issues sooner.

Engineering environments should expose collaboration risks before they become larger problems.

Insight 5: AI Responsibility Outperforms AI Avoidance

Students who responsibly integrate AI often move faster than students who avoid AI entirely.

This became one of the strongest educational observations.

Students initially hesitant to use AI often struggle to keep pace with teams that responsibly adopt it.

Students should not be taught to avoid AI.

Students should be taught to govern AI.

The educational objective is AI responsibility.

The governing principle remains:

> AI proposes; engineers verify.

This insight heavily influenced ETIS.

Insight 6: Students Need Permission To Use AI Broadly

Many students initially assume AI should only be used for implementation.

Students should instead be encouraged to use AI throughout the entire engineering lifecycle.

Students should use AI for:

- brainstorming
- requirements
- architecture
- implementation
- reviews
- testing
- operations
- reflection

Students quickly learn that AI is an engineering collaborator rather than a coding tool.

This insight significantly expanded ETIS.

Insight 7: Students Build Better Projects When They Build For Themselves

Students become more invested when projects have meaning beyond the semester.

Students are encouraged to think about projects as professional portfolio assets.

Students should build systems they may continue developing for one to two years after the course whenever practical.

This changes student motivation.

Students begin thinking about long-term ownership rather than short-term completion.

Insight 8: Graduate Students Elevate Engineering Maturity

Mixed undergraduate and graduate teams create stronger engineering environments.

Graduate students frequently elevate expectations naturally.

Graduate students often provide:

- leadership
- mentorship
- perspective
- stronger engineering rigor

The objective is collaborative maturity rather than hierarchy.

This implementation model has proven valuable.

Insight 9: Students Want Real Engineering Experiences

Students consistently respond positively when educational activities resemble industry practices.

Students often express appreciation for learning approaches that feel authentic.

Students want to understand:

- how engineering actually works
- how teams collaborate
- how decisions are made
- how engineering accountability functions

Educational authenticity matters.

This insight strongly influenced ETIS.

Insight 10: Educational Systems Need Feedback Loops

Educational systems should continuously evolve.

Educational systems should never become static.

Educational evidence should be continuously collected.

Observations should continuously refine educational frameworks.

The relationship should remain bi-directional.

Educational laboratories strengthen educational systems.

Emerging ETIS Educational Doctrines

Several educational doctrines emerged directly from this implementation.

Educational work should resemble professional engineering work.

Engineering accountability is the educational outcome, not the side effect.

Students should graduate with evidence of engineering ability rather than evidence of course completion.

Repository-centered engineering creates continuity between education and professional practice.

AI responsibility outperforms AI avoidance.

Educational laboratories are where educational frameworks become trustworthy.

Core Thesis

COMP330/474 demonstrates that educational systems become stronger when educational theory and educational reality continuously inform one another.

The framework informs the course.

The course informs the framework.

The learners transform.

Future Evolution

Purpose

This document describes how the Loyola University Chicago COMP330/474 implementation may continue evolving over time.

The objective is not to create a fixed roadmap.

The objective is to preserve a continuous improvement mindset.

This implementation intentionally operates as a living educational system.

Educational systems should continuously evolve as technology, engineering practices, AI capabilities, and student needs change over time.

The implementation should remain stable in philosophy while remaining adaptable in practice.

Core Philosophy

Educational systems should evolve.

Educational doctrines should endure.

The goal is not educational stability.

The goal is educational resilience.

Future evolution should preserve engineering accountability while allowing educational environments to adapt over time.

Long-Term Vision

The long-term objective is to continuously strengthen the relationship between ETIS and real educational environments.

ETIS

↓

Educational Assets

↓

COMP330/474

↓

Educational Evidence

↓

ETIS Refinement

↓

Future Implementations

The implementation should remain a continuous educational feedback system.

Evolution Area 1: Strengthen Student Onboarding

Students enter the course with varying levels of software engineering maturity.

Future implementations should continue improving onboarding experiences.

Areas for continued investment include:

- repository-centered engineering
- ETIS principles
- AI accountability
- engineering evidence expectations
- trustworthy engineering concepts

Students should establish strong foundations early.

Early confusion often creates downstream friction.

Evolution Area 2: Strengthen AI Responsibility

AI capabilities will continue evolving rapidly.

Educational environments should evolve alongside them.

Future implementations should continue strengthening:

- AI governance
- AI verification
- AI disclosure
- AI risk awareness
- AI engineering workflows

Students should become responsible AI collaborators.

AI responsibility should become a normal engineering behavior.

Evolution Area 3: Strengthen Engineering Review Culture

Students often arrive with limited experience participating in formal engineering reviews.

Future implementations should continue strengthening:

- review board experiences
- peer reviews
- engineering defenses
- evidence evaluations
- engineering discussions

Engineering work should become increasingly defendable over time.

Evolution Area 4: Strengthen Repository Analysis

Students should become increasingly sophisticated in how they manage repositories.

Future implementations should continue strengthening:

- repository organization
- engineering evidence integration
- traceability
- repository hygiene
- repository storytelling

Repositories should continue evolving into engineering memory systems.

Evolution Area 5: Strengthen Professional Portfolio Development

Professional portfolio development should continue becoming more intentional.

Students should graduate with demonstrable engineering evidence.

Future implementations should continue helping students build artifacts they can use throughout their careers.

Students should be able to explain:

- what they built
- why they built it
- how they verified it
- what tradeoffs they made
- how AI was governed
- how risks were managed

Students should leave with engineering narratives rather than isolated assignments.

Evolution Area 6: Strengthen Operational Thinking

Operational thinking should continue moving earlier in the semester.

Students should begin thinking beyond implementation sooner.

Future implementations should continue strengthening:

- release readiness
- observability
- operations
- reliability thinking
- long-term ownership

Students should increasingly think like system stewards.

Evolution Area 7: Expand Future Institutional Adoption

This implementation should eventually serve as a model for future adopters.

Future implementations should help other institutions understand:

- what should be preserved
- what may be adapted
- how engineering accountability is maintained
- how ETIS is operationalized

The goal is educational scalability rather than educational replication.

Institutions should inherit doctrine, not implementations.

Evolution Area 8: Strengthen Educational Evidence Collection

Educational observations should continue being preserved.

Educational systems improve when evidence is continuously collected.

Future implementations should continue capturing:

- student patterns
- engineering behaviors
- AI usage patterns
- team dynamics
- repository patterns
- adoption insights

Educational memory should not be lost between semesters.

Evolution Area 9: Preserve Professional Relevance

Technology will evolve.

Tools will evolve.

AI will evolve.

Professional expectations will evolve.

The implementation should continuously adapt while preserving its engineering philosophy.

Students should continue learning durable engineering behaviors rather than transient technologies.

The implementation should remain future oriented.

What Should Never Change

Several principles should remain stable over time.

Educational work should resemble professional engineering work.

Repository-centered engineering should remain foundational.

Engineering accountability should remain continuous.

AI should remain governed and accountable.

Students should graduate with evidence of engineering ability.

Operational thinking should remain integral.

Educational systems should remain bi-directional.

The framework should inform the course.

The course should inform the framework.

Core Thesis

The future of COMP330/474 is not to become a static software engineering course.

Its future is to remain a continuously evolving educational system that transforms students into future trustworthy engineers while simultaneously strengthening ETIS itself.