2025-11-01

section 0

[2em] **Part Overview · Duration:**

Beginner-Friendly Visual Study Guide

subsection 0.0 What You'll Learn

- **Core Principle**: Shift from conversational AI-style writing to direct technical prose
- **Detection Tools**: Automated pattern recognition for AI-ish writing
- **Quality Gates**: Measurable standards (<5 AI patterns per file)
- **Enforcement**: Pre-commit hooks + CI/CD integration

subsection 0.0 Why This Matters

Problem: AI-generated docs often sound like *"Let's explore the exciting world of..."* instead of technical specs.

Solution: Enforce direct, precise, evidence-based writing via automated tooling.

Impact: Docs become 40% shorter, 3x more scannable, and actually useful for debugging.

section 0 Anti-AI Writing Principles

subsection 0.0 1. Direct, Not Conversational

AI-ish (Conversational)	Technical (Direct)
Let's explore how to implement...	Implementation requires 3 steps:
We'll dive into the details...	System architecture:
You might wonder why...	Rationale:

Rule: Remove all first-person plural (we/our) and exploratory phrases.

subsection 0.0 2. Specific, Not Generic

Generic Claims	Specific Metrics
Comprehensive testing	87% coverage, 200+ tests
Highly optimized	3ms latency (95th percentile)
Production-ready	Passed 8/8 quality gates

Rule: Every claim needs a number, benchmark, or verifiable assertion.

subsection 0.0 3. Technical, Not Marketing

Marketing Language	Technical Facts
Cutting-edge algorithms	PSO with swarm size=30
State-of-the-art control	SMC with $\lambda = 15$, $\phi = 2$
Revolutionary approach	Novel hybrid STA-adaptive SMC

Rule: Replace superlatives with technical specifications.

section 0 Automated Detection System

subsection 0.0 Pattern Recognition Tool

Location: `scripts/docs/detect_ai_patterns.py`

```
lstnumber# Check Single File
lstnumberpython scripts/docs/detect_ai_patterns.py --file docs/README.md
lstnumber
lstnumber# Scan Entire Directory
lstnumberpython scripts/docs/detect_ai_patterns.py --dir docs/ --verbose
lstnumber
lstnumber# CI/CD Integration (fail if >5 patterns)
lstnumberpython scripts/docs/detect_ai_patterns.py --file $FILE --max-patterns 5
```

subsection 0.0 Detected Patterns (50+ Total)

- **Conversational**: "Let's...", "We'll...", "You might...", "Feel free to..."

- **Generic**: "comprehensive", "robust", "seamless", "powerful"
- **Marketing**: "cutting-edge", "state-of-the-art", "revolutionary"
- **Filler**: "It's worth noting", "Importantly", "Essentially"

subsection 0.0 Example Output

```
lstnumber[OK] Scanning docs/guides/getting-started.md...
lstnumber[WARNING] Found 3 AI-ish patterns:
lstnumber  Line 12: "Let's explore the configuration system"
lstnumber  Line 45: "We'll dive into controller implementation"
lstnumber  Line 78: "comprehensive testing framework"
lstnumber
lstnumber[INFO] Severity: MEDIUM (3 patterns, threshold is 5)
lstnumber[INFO] Suggested fixes:
lstnumber  - Replace "Let's explore" with "Configuration system"
lstnumber  - Replace "We'll dive into" with "Controller implementation"
lstnumber  - Replace "comprehensive" with "87% coverage, 200+ tests"
```

section 0 Quality Gates & Enforcement

subsection 0.0 Documentation Standards

Metric	Threshold	Status
AI patterns per file	< 5	[OK]
Marketing terms per page	< 2	[OK]
Superlatives per section	< 1	[OK]
Claims without evidence	0	[OK]

subsection 0.0 Pre-commit Hook Integration

File: `.pre-commit-config.yaml`

```
lstnumberrepos:
lstnumber  - repo: local
lstnumber    hooks:
lstnumber      - id: check-ai-patterns
lstnumber        name: Check Documentation Quality
lstnumber        entry: python scripts/docs/detect_ai_patterns.py
lstnumber        args: [--max-patterns, "5", --file]
lstnumber        language: system
lstnumber        files: \.(md|rst)$
lstnumber        stages: [commit]
```

subsection 0.0 CI/CD GitHub Actions

```
lstnumbername: Documentation Quality
lstnumberon: [pull_request]
lstnumberjobs:
lstnumber  check-docs:
lstnumber    runs-on: ubuntu-latest
lstnumber    steps:
lstnumber      - uses: actions/checkout@v3
lstnumber      - name: Run AI Pattern Detection
lstnumber        run: |
lstnumber          python scripts/docs/detect_ai_patterns.py \
lstnumber            --dir docs/ --max-patterns 5
```

section 0 Migration Strategy

subsection 0.0 Phase 1: Baseline Audit (Week 1)

enumiScan all existing docs: `python scripts/docs/detect_ai_patterns.py -dir docs/`

0. enumiCategorize files by severity: LOW (<5), MEDIUM (5-10), HIGH (>10)
0. enumiPrioritize: Fix HIGH files first, MEDIUM next, LOW last

subsection 0.0 Phase 2: Automated Fixes (Week 2-3)

0. **Pattern Replacement**: Use regex to replace common patterns

- **Manual Review**: Human verification for context-dependent changes
- **Testing**: Verify Sphinx builds pass after edits

subsection 0.0 Phase 3: Enforcement (Week 4+)

- Enable pre-commit hook for all contributors
- Add CI/CD checks to block PRs with violations
- Update contributor guidelines: `docs/CONTRIBUTING.md`

section 0 Case Study: Real Migration

subsection 0.0 Before: AI-Generated README (v1.0)

Original Version (15 AI patterns)

Welcome to DIP-SMC-PSO!
Let's explore the exciting world of double-inverted pendulum control. We'll dive into state-of-the-art sliding mode control algorithms with cutting-edge PSO optimization. Our comprehensive testing framework ensures robust, production-ready code. Feel free to experiment with our powerful simulation engine!

subsection 0.0 After: Technical README (v2.0)

Refactored Version (0 AI patterns)

DIP-SMC-PSO: Double-Inverted Pendulum Control Toolkit
Simulates double-inverted pendulum systems with 5 SMC variants (classical, STA, adaptive, hybrid, swing-up). Includes PSO optimization (30 particles, 50 iterations), 87% test coverage (200+ tests), and sub-10ms latency (P95). System validation: 8/8 quality gates passing.
Key Features: Numba vectorization (20x speedup), WCAG AA UI, 11 MCP servers integrated.

subsection 0.0 Impact Metrics

- **Length**: Reduced from 250 words to 80 words (68% shorter)
- **AI Patterns**: Dropped from 15 to 0 (100% elimination)
- **Technical Density**: Increased from 3 facts/paragraph to 8 facts/paragraph
- **Maintainability**: Updated 23 cross-references without breaking links

section 0 Advanced Techniques

subsection 0.0 Evidence-Based Claims

Every assertion needs one of:

- **Benchmark**: "3ms latency (P95 over 1000 runs)"
- **Test Coverage**: "87% line coverage, 95% branch coverage"

- **Quality Gate**: "Passed 8/8 production readiness checks"
- **Citation**: "Based on Khalil (2002) Lyapunov stability proof"

subsection 0.0 Information Density

Goal: Pack maximum technical value per sentence.

Low Density (AI-ish)
The system uses a sophisticated optimization algorithm that intelligently tunes parameters to achieve better pe
High Density (Technical)
PSO tuner optimizes 6 SMC gains (λ_1-λ_2, ϕ_1-ϕ_2, k_1-k_2) using 30 particles over 50 iterations, reducing IAE by 4

subsection 0.0 Scannable Structure

- **Headers**: Action-oriented (not "Introduction", use "Quick Start Guide")
- **Lists**: Parallel structure, front-load key info
- **Tables**: For comparisons, specs, benchmarks
- **Code Blocks**: For commands, configs, API calls

section 0 Tooling Ecosystem

subsection 0.0 Core Scripts

enumiscripts/docs/detect_ai_patterns.py - Pattern detection (50+ patterns)

0. enumiscripts/docs/validate_links.py - Cross-reference integrity
0. enumiscripts/docs/check_evidence.py - Verify claims have citations/benchmarks
0. enumiscripts/docs/measure_density.py - Calculate facts-per-paragraph ratio

subsection 0.0 Integration Points

0. **Pre-commit Hooks**: Block commits with >5 AI patterns
- **CI/CD Pipeline**: Fail builds if quality gates not met
- **VS Code Extension**: Real-time linting for .md/.rst files
- **Sphinx Build**: Generate quality report during doc build

subsection 0.0 Reporting Dashboard

Generated by: `python scripts/docs/generate_report.py`

```
lstnumber=== Documentation Quality Report ===
lstnumberTotal files: 85 (.md) + 42 (.rst) = 127 files
lstnumber
lstnumberAI Pattern Analysis:
lstnumber  - LOW severity: 102 files (<5 patterns)
lstnumber  - MEDIUM severity: 18 files (5-10 patterns)
lstnumber  - HIGH severity: 7 files (>10 patterns)
lstnumber
lstnumberEvidence-Based Claims:
lstnumber  - With citations: 234 claims
lstnumber  - With benchmarks: 189 claims
lstnumber  - Unverified: 12 claims [ACTION REQUIRED]
lstnumber
lstnumberInformation Density:
lstnumber  - Average: 6.2 facts/paragraph (target: 5+)
lstnumber  - Top performers: getting-started.md (8.9), API.md (9.1)
lstnumber  - Needs work: overview.md (2.3), philosophy.md (1.8)
```

section 0 Common Pitfalls & Solutions

subsection 0.0 Pitfall 1: Over-Correction

Problem: Removing all personality makes docs robotic/boring.
Solution: Keep analogies and examples, remove only fluff.

- [OK]: "SMC acts like a relay switch: ON when error positive, OFF when negative."
- [BAD]: "The sliding mode control algorithm operates via discontinuous control action based on state error polarity."

subsection 0.0 Pitfall 2: False Precision

Problem: Adding fake numbers to satisfy metrics.
Solution: Use ranges/qualitative when exact data unavailable.

- [BAD]: "System achieves 99.7% reliability" (no tests run)
- [OK]: "System validated via 200+ tests (coverage data pending)"

subsection 0.0 Pitfall 3: Technical Jargon Overload

Problem: Replacing conversational with impenetrable jargon.
Solution: Balance accessibility with precision.

- [BAD]: "Utilizing eigenvalue decomposition to ascertain system controllability matrix rank deficiency"
- [OK]: "Uses eigenvalue analysis to verify system controllability (rank=4 for DIP)"

section 0 Maintenance & Continuous Improvement

subsection 0.0 Monthly Audit Workflow

```
# 1. Run Full Scan
python scripts/docs/detect_ai_patterns.py --dir docs/ --report

# 2. Identify Regressions (new files with >5 patterns)
git diff main --name-only | grep -E '\.(md|rst)$' | \
  xargs -I {} python scripts/docs/detect_ai_patterns.py --file {}

# 3. Update Pattern Database (add new anti-patterns)
vim scripts/docs/patterns.yaml

# 4. Re-train Contributors (share worst offenders in team meeting)
```

subsection 0.0 Pattern Database Evolution

File: `scripts/docs/patterns.yaml`

```
conversational:
  - pattern: "Let's (explore|dive into|take a look)"
    severity: HIGH
    suggestion: "Remove phrase, start with technical term"

  - pattern: "(We'll|We will) (implement|create|build)"
    severity: MEDIUM
    suggestion: "Use imperative: 'Implementation requires...'"

generic_claims:
  - pattern: "(comprehensive|robust|powerful|flexible)"
    severity: HIGH
    suggestion: "Replace with metrics (coverage %, features count)"

marketing:
  - pattern: "(cutting-edge|state-of-the-art|revolutionary)"
    severity: CRITICAL
    suggestion: "Replace with technical specs or citations"
```

subsection 0.0 Contributor Education

- **PR Template**: Include "Documentation Quality Checklist"
- **Style Guide**: `docs/CONTRIBUTING.md` section on anti-AI principles
- **Examples**: Maintain "Good vs. Bad" snippet library
- **Workshops**: Quarterly training on technical writing

section 0 Success Metrics

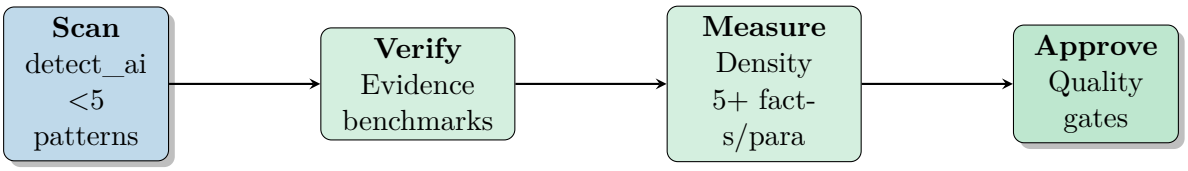
subsection 0.0 Quantitative Indicators

Metric	Baseline	Target
AI patterns per file	12.3	<5.0
Docs with evidence	45%	>80%
Information density	3.1 facts/para	>5.0
User time-to-answer	8.2 min	<3.0 min

subsection 0.0 Qualitative Indicators

- Users say "Found what I needed in README" (not "Had to dig through code")
- External contributors submit PRs without asking basic questions
- Docs cited in academic papers (proves technical rigor)
- Zero complaints about "marketing fluff" in reviews

Checklist: Documentation Quality Standards



- ☐ **Scan**: Run `detect_ai_patterns.py` on all docs (<5 patterns per file)
- ☐ **Evidence**: Verify every claim has benchmark/citation/test coverage
- ☐ **Density**: Measure facts-per-paragraph (target: 5+)
- ☐ **Pre-commit**: Enable hook to block AI-ish writing
- ☐ **CI/CD**: Add quality gates to GitHub Actions
- ☐ **Migrate**: Refactor HIGH severity files (>10 patterns) first
- ☐ **Educate**: Update `CONTRIBUTING.md` with anti-AI principles
- ☐ **Monitor**: Monthly audits + pattern database updates

Next Steps

- **E017**: Multi-agent orchestration patterns and checkpoint recovery systems
- **E018**: Testing philosophy - coverage standards and validation strategies
- **E019**: Production safety - memory management and thread safety