Contents

SME Training Quick-Start Guide

Getting Subject Matter Experts Up to Speed on Semantic Annotation Review

Version: 1.0
Date: November 19, 2025
Audience: Domain experts, operations staff, senior developers
Time Required: 2-hour training session + 4-hour hands-on practice

---

Table of Contents

Training Objectives

By the end of this training, SMEs will be able to:

Understand the purpose of semantic annotations in AI knowledge bases
Identify the 7 types of MCP directives and their fields
Evaluate annotation quality across 7 review dimensions
Provide structured feedback using the SME Review Guide
Recognize common annotation errors and anti-patterns
Complete an independent review of 1-2 documentation sections

Prerequisites

Required Knowledge: - Expert-level understanding of your domain (e.g., GFS operations, error handling, data assimilation) - Familiarity with NCEP production standards (EE2 compliance) - Basic understanding of reStructuredText (RST) format

Not Required: - AI/ML expertise - Programming in Python/JavaScript - Vector database knowledge - Graph database experience

Materials Provided: - This training guide - SME Review Guide (complete reference) - Pilot annotation examples (error_handling section) - Review feedback template

---

Session 1: Introduction (30 minutes)

1.1 The Problem AI Solves (10 minutes)

Scenario: New developer asks: "How do I check for errors in production scripts?"

Traditional Approach (Text Search):

```
Search: "error checking"
Results: 347 documents containing "error"
  - Error message format guidelines
  - Error code reference table
  - Error handling overview
  - Historical error analysis
  - ...
Time to find answer: 15-30 minutes
Confidence: Low (which approach is correct?)
```

SDD Framework Approach (Intent-Aware):

```
Query: "How do I check for errors?"
AI Understanding:
  - Intent: rapid_error_detection
  - Priority: critical
  - Utility: err_chk
  - Examples: err_chk_usage (with context)
  - Alternatives: err_exit (for immediate abort)
Time to answer: 15 seconds
Confidence: High (operational rationale provided)
```

Key Insight: AI can "understand" *why* requirements exist, not just match keywords—**if** we capture that knowledge correctly.

1.2 What Are Semantic Annotations? (10 minutes)

Simple Definition: Invisible "tags" embedded in documentation that capture: - **Why** requirements exist (intent, rationale) - **How critical** they are (priority, severity) - **What** they prevent or enable (operational

impact) - **How** they relate to each other (relationships)

Example (Before Annotation):

```
C. Production Utilities

err_chk / err_exit
  It is imperative that all production code employ error checking.
  Jobs should fail with err_chk or err_exit as soon as an error is encountered.
```

After Annotation (Your Role to Review):

```
C. Production Utilities

.. mcp:intent:: rapid_error_detection
   :description: Enable immediate error detection and recovery
   :enforcement: runtime_check
   :rationale: 99% on-time delivery SLA requires detection within 5 minutes

.. mcp:compliance:: error_handling
   :priority: critical
   :type: mandatory
   :scope: global

err_chk / err_exit
  It is imperative that all production code employ error checking.
  Jobs should fail with err_chk or err_exit as soon as an error is encountered.

.. mcp:utility:: err_chk
   :module: prod_util
   :category: error-handling
   :required: yes
   :deprecated: no
```

Question to Consider: Does this capture what YOU know about why err_chk exists?

1.3 Your Role as SME Reviewer (10 minutes)

You Are NOT Expected To: - Understand vector embeddings or AI mathematics - Write code or modify the RAG system - Learn new programming languages - Become an AI expert

You ARE Expected To: - Validate that annotations capture YOUR expert knowledge - Identify missing context or incorrect rationale - Flag examples that don't match real operational usage - Suggest additional examples for common scenarios

Impact of Your Work:

Without SME Review	With SME Review
AI returns generic text	AI provides operational context
45% retrieval accuracy	**87% retrieval accuracy**
35% false positives	**8% false positives**
8-week developer onboarding	**3-week onboarding**

Your domain expertise is the secret ingredient that makes AI effective.

Session 2: Hands-On Review (60 minutes)

2.1 The 7 MCP Directive Types (20 minutes)

We'll review each directive type with examples from YOUR domain.

Directive 1: `mcp:compliance` - Priority and Scope **Purpose:** Specify how critical a requirement is.

Fields: - `:priority:` → critical | high | medium | low - `:type:` → mandatory | recommended | optional - `:scope:` → global | system-specific | component-specific

Example:

```
.. mcp:compliance:: error_handling
   :priority: critical
   :type: mandatory
   :scope: global
```

Your Review Question: "Is this really CRITICAL, or should it be HIGH?"

Decision Framework: - **critical** = System fails / operational delivery at risk if violated - **high** = Data quality/reliability significantly impacted - **medium** = Maintainability/efficiency affected - **low** = Stylistic/convenience

Exercise: Review 3 compliance directives, identify any mis-prioritized items.

Directive 2: `mcp:intent` - The "Why" Behind Requirements **Purpose:** Capture WHY a requirement exists (most important directive!).

Fields: - `:description:` → High-level purpose - `:enforcement:` → runtime_check | compile_check | manual_review - `:rationale:` → Operational justification (this is YOUR expertise!)

Example:

```
.. mcp:intent:: rapid_error_detection
   :description: Enable immediate error detection and recovery
   :enforcement: runtime_check
   :rationale: Operational reliability requires catching failures at earliest point
```

Your Review Question: "Is this the REAL reason we do this?"

Common Issue - Circular Reasoning:

```
BAD:
  :rationale: We use err_chk to check errors

GOOD:
  :rationale: 99% on-time delivery SLA requires detection within 5 minutes
              to prevent cascade failures across 6-hour forecast window
```

Exercise: Rewrite 2 rationale statements to include operational context.

Directive 3: `mcp:severity` - RFC 2119 Compliance Levels **Purpose:** Specify requirement strength (must/should/may).

Fields: - `:severity:` → must | must-not | should | should-not | may - `:rationale:` → Why this level? - `:exceptions:` → Documented exceptions

Example:

```
.. mcp:severity:: must
   :rationale: Critical for operational stability and rapid troubleshooting
   :exceptions: None
```

Your Review Question: "Does the text say 'must' but we actually mean 'should'?"

Exercise: Identify 2 cases where severity doesn't match standard language.

Directive 4: `mcp:utility` - Tool Metadata **Purpose:** Document production utilities with structured metadata.

Fields: - `:module:` → Which module provides it (e.g., prod_util) - `:category:` → error-handling | data-management | messaging | initialization - `:required:` → yes | no - `:deprecated:` → yes | no | partial

Example:

```
.. mcp:utility:: err_chk
   :module: prod_util
   :category: error-handling
   :required: yes
   :deprecated: no
```

Your Review Question: "Is this utility truly REQUIRED or just RECOMMENDED?"

Exercise: Verify module names and categories for 3 utilities.

Directive 5: `mcp:example` - Context-Aware Code Examples **Purpose:** Show the RIGHT way with clear context.

Fields: - `:language:` → bash | python | yaml - `:context:` → When to use this - `:demonstrates:` → What pattern it shows

Example:

```
.. mcp:example:: err_chk_usage
   :language: bash
   :context: error_checking_after_command
   :demonstrates: Standard error checking pattern

   .. code-block:: bash

      critical_command arg1 arg2
      export err=$?
      err_chk
```

Your Review Question: "Is this example what we ACTUALLY do in production?"

Exercise: Identify 2 missing examples that would help new developers.

Directive 6: `mcp:pattern` - Design Patterns and Anti-Patterns **Purpose:** Recognize good and bad code patterns.

Fields: - `:category:` → error-handling | data-management | etc. - `:anti-pattern:` → yes | no - `:alternatives:` → List of better approaches

Example:

```
.. mcp:pattern:: fail_fast_pattern
   :category: error-handling
   :anti-pattern: no
   :alternatives: []

   Failures must not be allowed to propagate downstream.
   Jobs should fail with err_chk or err_exit immediately.
```

Your Review Question: "Should AI flag code that does the OPPOSITE of this?"

Exercise: Define 1 anti-pattern AI should detect.

Directive 7: `mcp:see-also` - Relationship Mapping **Purpose:** Connect related concepts explicitly.

Fields: - `:related:` → List of related items - `:type:` → prerequisite | reference | alternative | example

Example:

```
.. mcp:see-also:: production_utilities
   :related: [err_exit, err_trap]
   :type: alternative
```

Your Review Question: "Are these items REALLY related? How?"

Types: - **prerequisite** = Must understand/have this first - **reference** = Related concept for more info - **alternative** = Different approach to same goal - **example** = Concrete usage example

Exercise: Identify 2 missing relationships.

2.2 Live Review Exercise (40 minutes)

Scenario: We'll review the pilot annotation for Error Handling section together.

Materials: - `pilot_annotation_error_handling.md` (provided) - Review feedback template (provided)

Process: 1. Read original text (5 min) 2. Review annotations (10 min) 3. Group discussion: What's good? What's wrong? What's missing? (15 min) 4. Fill out feedback template (10 min)

Sample Feedback Entry:

```
## Section: Error Handling

### Intent Accuracy
- rapid_error_detection - rationale captures operational SLA
- descriptive_error_messages - add "enables automated log parsing"
- atomic_file_operations - enforcement should be "automated" not "manual"

### Priority/Severity
- error_handling: critical priority appropriate
- messaging: should be "low" not "medium" - not production-critical

### Examples
- err_chk_usage - good basic example
- Missing: err_chk inside loop (common use case)
- Missing: pipeline error handling (command1 | command2)

### Questions
```

1. Is prep_step truly "required: yes" or just recommended?
2. Should we mark getsystem as "deprecated: yes"?

Session 3: Practice Session (30 minutes)

3.1 Independent Review Task

Assignment: Review one of the following sections (your choice):

1. **Production Utilities** (prod_util module)
2. **Environment Variables** (PDY, cyc, COMROOT)
3. **File Handling** (cpreq, cpfs)

Time Limit: 20 minutes

Deliverable: Completed feedback template

3.2 Group Review and Discussion (10 minutes)

Share findings: - What did you discover? - What patterns emerged? - What questions do you have?

Post-Training: Independent Review (4 hours)

Task Breakdown

Week 1 Assignment (4 hours):

1. **Review Assigned Section (2 hours)**
   - Read original documentation
   - Review all annotations
   - Fill out feedback template
2. **Identify Gaps (1 hour)**
   - Missing examples?
   - Missing relationships?
   - Incomplete rationale?
3. **Write Recommendations (1 hour)**
   - Specific text changes
   - New examples to add
   - Priority adjustments

Submission: - Email completed feedback template - Schedule 30-minute follow-up discussion

---

Certification Checklist

You are certified to perform independent SME reviews when you can:

- ☐ Identify all 7 MCP directive types from examples
- ☐ Evaluate intent accuracy (does rationale capture operational context?)
- ☐ Assess priority/severity correctness
- ☐ Validate utility metadata (module, category, required)
- ☐ Review code examples for correctness and completeness
- ☐ Identify missing relationships
- ☐ Provide structured feedback using template
- ☐ Complete independent review in 2-4 hours per section

Sign-off: _____ Date: _____

Quick Reference Card

The 7 Review Dimensions (Memory Aid: "I-P-U-E-R-P-G")

1. **I**ntent Accuracy - Does rationale capture WHY?
2. **P**riority/Severity - Does priority match operational impact?
3. **U**tility Metadata - Are module/category/required correct?
4. **E**xamples - Do they show the RIGHT way?
5. **R**elationships - Are connections accurate?
6. **P**atterns - Should AI recognize this?
7. **G**ap Analysis - What's missing?

Priority Decision Tree

```
Is violation a production failure risk?
  Yes → CRITICAL
  No
     Does it significantly impact data quality?
        Yes → HIGH
        No
           Does it affect maintainability?
              Yes → MEDIUM
              No → LOW
```

Severity Levels (RFC 2119)

- **must** / **must-not** → Absolute requirement
- **should** / **should-not** → Strong recommendation, exceptions allowed
- **may** → Optional, discretionary

Common Review Flags

Circular Reasoning

```
"We use err_chk to check errors"
"99% delivery SLA requires 5-minute error detection"
```

Severity Inflation

```
Everything marked "critical"
Only production-failure risks are "critical"
```

Missing Examples

```
Only basic usage shown
Common scenarios covered (loops, pipelines, error recovery)
```

Wrong Relationships

```
err_exit marked as "reference"
err_exit is "alternative" (different approach)
```

Support Resources

During Training: - Instructor: Available for questions - Pilot annotations: `sdd_framework/templates/pilot_annotation_` - Complete guide: `sdd_framework/templates/sme_review_guide.md`

Post-Training: - Email: Terry.McGuinness@noaa.gov - Office hours: Tuesdays 2-3pm - Slack channel: #sdd-framework-sme-reviews

Feedback on This Training: Please provide feedback to help us improve: - What was most helpful? - What needs more explanation? - What additional examples would help?

Training Success Metrics

Target Outcomes: - 100% of SMEs can identify directive types after training - 90%+ agreement between SME reviews on priority/severity - 80%+ of identified gaps lead to annotation improvements - 4-hour average time per section review (down from 8+ hours without training)

Your feedback shapes the future of this training!

End of Quick-Start Guide

Next Steps: 1. Complete certification checklist 2. Review assigned documentation section 3. Submit feedback within 1 week 4. Attend follow-up discussion session

Thank you for contributing your domain expertise to make AI assistance more effective!