S1000D Style Guides for Effective Content Reuse in Fault Isolation Manuals

Mark Langley
Chief Technology Officer
ATP CaseBank
E-mail: mlangley@casebank.com
Who am I?

• I’m not...
  • An S1000D Content Author, or manager of authors
  • A rep for a company selling S1000D tool sets
  • On any standards committee

• I am...
  • A developer of intelligent systems
  • A believer that structured content reuse can extend beyond data exchange and PDF vs. IETM
  • A believer that machine-actionable knowledge can be extracted from S1000D
Why am I here?

• To argue for the use of semantic style guides in authoring fault isolation procedures

• Why should you care?
  • You author info code 421 data modules
  • Your content comes from multiple teams – in house and suppliers
  • You want to unlock the value of your authored content – beyond PDFs and IETMs

• Why else might you care?
  • You want high quality, consistent content
  • You want cost-effective authoring and validation
But S1000D is Semantic – And I Have a Style Guide

• True - S1000D is highly semantic compared to iSpec 2200
  • “what is this content” instead of “how is it structured”
  • <isolationStep>, <action>, <isolationStepQuestion>, <isolationStepAnswer>
    vs. pageblocks, paragraphs, and nested lists

• And everyone already has a style guide
  • (though you might not always follow it)
  • Directive vs. question format for test instructions
  • Pin-to-pin wiring checks vs. “troubleshoot in accordance with schematic”
  • Support equipment called out at start of procedure vs. as needed

• So how is a “Semantic Style Guide” different?
A Semantic Style Guide...

• Enumerates the classes (types) of instructions (tests and corrective actions) that can be issued

• Requires all instructions of the same class to be consistently authored
  • Defines a template (boilerplate structure and fill-in-the-blanks parameters) for each class
  • Uses unambiguous wording to differentiate each class

• Allows intelligent behavior to be associated with each class

• Let’s dive into these in turn...
Instruction Class examples

- Perform procedure
- Do an inspection
  - General visual inspection
- Basic wiring check
  - Continuity check
  - Resistance check
- Clean/tighten
- Replace
  - Repair/replace

- Cycle power
- Checks
  - Function check
  - Check for related fault codes
  - Generic check
- Operational Test
- Fuel contamination check
- Circuit breaker reset/open/close
Achieving Consistency via Semantic Style Templates

- Each instruction class has one or more associated templates
- Each template accepts one or more parameters
- EG: Basic wiring check: “Refer to applicable wiring diagram and inspect the wiring between [P1] and [P2] for damage”
Why Consistency Matters

• Two DMs with same `<isolationStep>` “check DCM 2A card installation”
  • Two inconsistent names for the card

<title>Possible cause one (1), DCM 2A incorrectly installed</title>

<title>Possible cause one (1), DCM 2A card</title>
<action>Check the DCM 2A card for correct installation...</action>

• Not really an issue for PDF or IETM publications
• Significant issue when loading into an advanced reasoner

<table>
<thead>
<tr>
<th>3143F0333 - Perform fault confirmation</th>
<th>More details</th>
<th>Fault Remains ▼</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>3143F0334 - Perform fault confirmation</td>
<td>More details</td>
<td>Fault Remains ▼</td>
<td>x</td>
</tr>
<tr>
<td>Rerack the DCM 2A</td>
<td>More details</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rerack the DCM 2A card</td>
<td>More details</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unambiguous Wording to Identify Classes

**Template**

- Do a check for [x].
- Replace the [x] in accordance with [y].
- Clean/tighten the [x].
- Do an operational test of the [x].

**Sample content**

- Do a check for corroded seals.
- Replace the starter air valve in accordance with EMM 80-11-01-000-801.
- Clean/tighten the connectors.
- Do an operational test of the ignition system.
Intelligent Behavior

- Knowing the class of each instruction allows cool things to happen, once we think beyond the IETM
  - Automatically run and report results of a Built In Test when a “perform IBIT” instruction is encountered
  - Consult a data download system to answer a “check for associated fault codes” instruction
  - Consult an as-built PLM system or as-maintained configuration tracking system for “confirm configuration” instructions
  - In a dynamic reasoner, deduce and apply costs to drive sequencing of tests
    - “perform fault confirmation” → Very Fast, Very Inexpensive
    - “check for associated codes” → Fast, Very Inexpensive
    - “check for fluid contamination” → Slow, Moderately Expensive
Sounds Interesting... What’s the Catch?

- You may, like us, start out thinking you’ll need a dozen or so instruction classes
  - In reality, we ended up with 40-50 classes and subclasses
- Why so many classes? Why can’t I get away with “test”, “reset”, “adjust”, and “replace?”
  - “A test by any other name, would take as long to perform” (not Shakespeare)
  - More specific guidance may be needed than simple pass/fail
    - e.g: contaminated/clear, high/low/within spec, ...
  - Data module consumer may respond to subclasses differently
- Rule of thumb – if a pattern is used in 10 or more data modules, it deserves its own subclass
So what’s the payoff?

• Consistency of authoring
• Content validation
• Intelligent behavior driven by semantics
Consistency of authoring

- Does this look familiar?

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make sure that the mid avionics compartment door 812 is closed.</td>
<td>A-J06-30-02-00AAA-030A-A - Sub zones - Technical data</td>
</tr>
<tr>
<td>Make sure that mid avionics compartment door 812 is closed. Refer to</td>
<td>A-J06-30-07-00AAA-030A-A - Doors zones - Technical data</td>
</tr>
<tr>
<td>Make sure that the mid avionics door 812 is closed. Refer to</td>
<td>A-J06-30-07-00AAA-030A-A - Doors zones - Technical data</td>
</tr>
<tr>
<td>Make sure that mid avionics compartment door 812 is closed.</td>
<td>A-J06-47-00-00AAA-030A-A - External fuselage doors - Technical data</td>
</tr>
<tr>
<td>Make sure that mid avionics compartment door 812 is closed.</td>
<td>A-J06-47-00-00AAA-030A-A - External fuselage doors - Technical data</td>
</tr>
<tr>
<td>Make sure that the mid avionics compartment door 812 is closed.</td>
<td>A-J06-47-00-00AAA-030A-A - External fuselage doors - Technical data</td>
</tr>
<tr>
<td>Make sure that the forward avionics compartment door 812 is closed.</td>
<td>A-J06-47-00-00AAA-030A-A - External fuselage doors - Technical data</td>
</tr>
<tr>
<td>Make sure that the middle avionics compartment door 812 is closed.</td>
<td>A-J06-47-00-00AAA-030A-A - External fuselage doors - Technical data</td>
</tr>
<tr>
<td>Make sure that the mid avionics compartment door 812 is closed.</td>
<td>A-J06-47-11-00AAA-030A-A - Nose fuselage servicing door and panel - Technical data</td>
</tr>
</tbody>
</table>

- “Check access panel” instruction “Make sure that the compartment door [identifier] is [status].” ensures you never deal with this again

- Flow down of style guide to vendors ensures consistency across all info code 421 DMs
Content Validation

• Semantic class-aware tooling can automate QA of newly authored content and enforce class-specific policies

• For example...
  • ... every “circuit breaker” instruction must include a reference to the info code 913 DM describing how to open/close the breaker
  • ... every procedure containing a “remove and replace LRU” instruction must include a subsequent “close circuit breaker” instruction

• The style guide easily enables these patterns to be detected and enforced
  • Much harder to enforce without clear semantic rules for content authoring
Intelligent Behavior Driven by Semantics

• Thinking beyond the IETM... imagine a world where the diagnostic support system recognizes, interprets, and responds intelligently to each maintenance instruction
  • “check fault history” class → automated query to the equipment health tracking system

```
<isolationStep id="Step0002">
 ...
 <action>On the OMS, scroll to the Fault History ... </action>
 ...
 <isolationStepQuestion>Does the OMS fault history show the maintenance message ... </isolationStepQuestion>
 </isolationStep>
```
Intelligent Behavior Driven by Semantics

• Thinking beyond the IETM some more...

• “contact customer support” class →
  • Collect all details of troubleshooting so far
  • Format as a human-readable summary
  • open the CRM system, create a support ticket, and attach the summary
  • Flag the troubleshooting interaction and feed it back into the authoring system for review

<isolationProcedureEnd id="Step0006">
  <action>Contact [REDACTED] Technical Help Desk - CRC.</action>
</isolationProcedureEnd>
How do we implement these semantics?

• New semanticClass attribute on <action> in <isolationStep>?  
  • In the short term, no... it’s way too early for that  
  • In the longer term, probably not... everyone’s semantic classes will be different, and tailored to their own needs

• Text analysis heuristics driven by the style guide?  
  • Yes – works well for us so far  
  • Each instruction class corresponds to one or more regular expressions  
  • Heuristics use regular expressions and supporting conditions to check each isolation step against the style guide and associate an instruction class  
  • Automated tooling runs the heuristics at time of authoring to reject any DM with steps that can’t be matched against an instruction class
Heuristic Example #1: Installation status

- There is an action in the format “Check the [LRU] for [status] installation”
- The same action contains one dmRef with the infoCode “720”
- There is no other dmRef with infoCode 720 in the step

```
<isolationStep id="Step0002">
    <title>Possible cause 1 (one), DCM 2A card</title>
    <action>... </action>
    <action>Check the DCM 2A card for correct installation. Refer to ... </action>
    <action>Make sure that the circuit breakers opened before are closed.</action>
    <isolationStepQuestion>Does the OMS continue to show the Maintenance message DCM 2A CARD IN DMC 2(A344) INPUT BUS PFCC_3-2 IS DISCONNECTED OR FAILED?</isolationStepQuestion>
    <isolationStepAnswer>
        <yesNoAnswer>
            <yesAnswer nextActionRefId="Step0003"/>
            <noAnswer nextActionRefId="Step0007"/>
        </yesNoAnswer>
    </isolationStepAnswer>
</isolationStep>
```
Heuristic Example #2:
Basic wiring check (Repair)

• There is an action in the form “Refer to the applicable wiring diagram and inspect the wiring between [X] and [Y] for damage.”
• No table in the step has a title containing both the words “wiring” and “check”
• The rule must not trigger the Voltage Check rule
• There is a yes/no question containing “Was a short circuit or damage found?”

```xml
<isolationStep id="Step0002">
  <title>Possible Cause 1 (one), Aircraft Wiring</title>
  <action>Refer to applicable wiring diagram and inspect the wiring between A715P1 and A46P1 for damage.</action>
  <isolationStepQuestion>Was a short circuit or damage found?</isolationStepQuestion>
  <isolationStepAnswer>
    <yesNoAnswer>
      <yesAnswer nextActionRefId="Step0003"/>
      <noAnswer nextActionRefId="Step0004"/>
    </yesNoAnswer>
  </isolationStepAnswer>
</isolationStep>
```
Our experience to date

• 2 projects populating an intelligent reasoner from fault isolation DMs.

• First project:
  • Pre-existing data modules, from multiple suppliers, no semantic style guide
  • Heuristics took many months to define, by trial and error
  • 80/20 rule: 20% of data modules consumed 80% of development effort. Another 20% were more efficient to convert manually

• Second project:
  • Multiple suppliers, new data modules authored to semantic style guide, with tool support to immediately reject data modules with invalid isolation steps
  • Completed in ¼ the time and effort, with more consistent, higher quality results
  • Authors voiced appreciation for detailed guidance provided in style sheets.
Summary

• Semantic style guides require some up-front definition

• But pay off in:
  • Consistent authoring within an organization, and between collaborating suppliers
  • Constrained editing results in faster, more efficient, and more cost effective authoring
  • Enables better content reuse, in ways not originally anticipated, by DM consumers yet to be invented

• With flexible tooling, Instruction Classes can be designed up front but also adapted over time
Next steps

• How are we taking the concept further?
• Applying semantic style sheets to two more development projects in 2018-2019
• Continuing to build semantic style guides into our internal tooling – for both S1000D and non-S1000D content
• Working with others to spread the concept
• Down the road:
  • Working toward a common set of semantic classes?
  • Artificial Intelligence: A neural network that learn how to classify the steps with “supervised learning” by Diagnostic Database Developers
Thank you for your attention!

Questions?

Mark Langley
Chief Technology Officer
ATP CaseBank
E-mail: mlangley@casebank.com