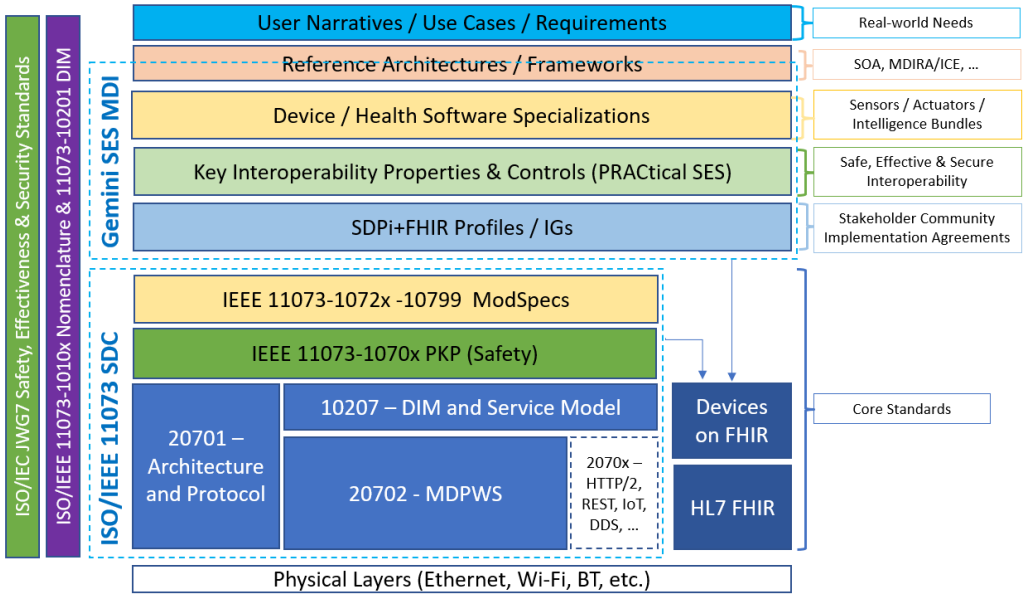
IHE-HL7 Gemini SES+MDI – *2022 RI+MC+RR Strategy– Word – Markdown – AsciiDoc ?*

Updated: 2022.07.22

Gemini SES+MDI
2022 RI+MC+RR Strategy
Word – Markdown – AsciiDoc ?



Current Strategy: *Word "Markup"*

ITI Use of Markdown for profiles.ihe.net Publication

ITI Special Pages & DEV OID Scheme

Door #3: AsciiDoc

Gemini SES+MDI – 2022 RI+MC+RR Strategy –

Current Strategy: Word "Markup"

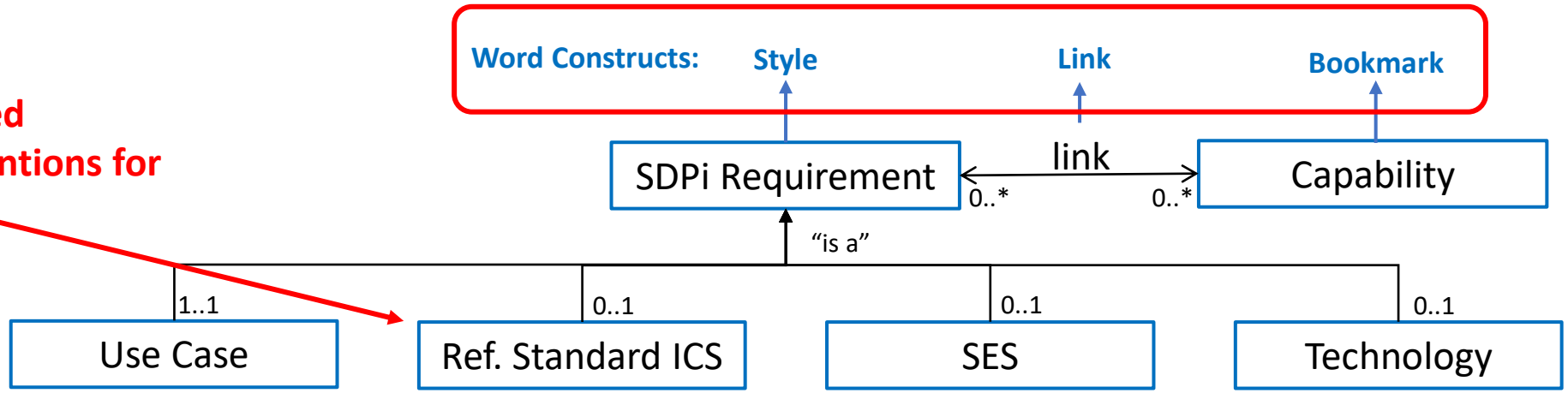
…Over the last 2+ years, the Gemini SES+MDI project has settled on am approach for creating and publishing the SDPi technical framework supplement that includes using the Word template as other IHE profiles, supporting RI using styles / bookmarks / links / references, and then processing using scripts that pull out the RI information. How does this kind of Word-based "markup" work and is it an unnecessary compromise?

Spec-to-Test Strategy – *RI+MC+RR Model*

Basic RI+MC+RR UML Model

"Basic" Requirements Interoperability Model *for SDPi 1.0 Document*:

Need standardized language / conventions for "Requirement" specification



Capability provides ***implementation*** for **Requirements**

Considerations / Homework:

- ❑ Word references / links require "bookmarks" – set places in the document (see also PKP examples)
- ❑ Word "styles" will require some naming method of each style / link / Bookmark to ensure continuity
- ❑ "Capability" Types needed?
- ❑ Identifiers / nomenclature required? SDPi Rxxxx? "Link" text?
- ❑ Create examples for each, including Text in Word + Word XML rendering
- ❑ Bidirectional navigation? Unidirectional (req to capability) sufficient? (Note: 0..* simply indicates that each end can be linked to / from multiples on the other side of the relationship)

Spec-to-Test Strategy – *RI+MC+RR Model*

Basic RI+MC+RR UML Model

ReqIF *Example*: *Requirement related to System Function Contribution*

```
<w:p w14:paraId="5672C5E0" w14:textId="77777777" w:rsidR="00743536" w:rsidRDefault="000A5E66">
   <w:pPr> <w:pStyle w:val="IEEEStdsParagraph"/> <w:jc w:val="left"/> </w:pPr>
   <w:bookmarkStart w:id="123" w:name="Requirement:R0062"/>
      <w:r> <w:rPr> <w:b/> </w:rPr> <w:t>R0062</w:t> </w:r>
   <w:bookmarkEnd w:id="123"/>
   <w:r> <w:rPr> <w:b/> </w:rPr>
•                                  <w:t>: If an SDC PARTICIPANT produces EXCESSIVE LOAD CONDITIONs, each
         SDC PARTICIPANT that is affected by these conditions SHALL maintain its SYSTEM FUNCTION
         CONTRIBUTIONs for other SDC PARTICIPANTs.</w:t>
   </w:r>
</w:p>
```

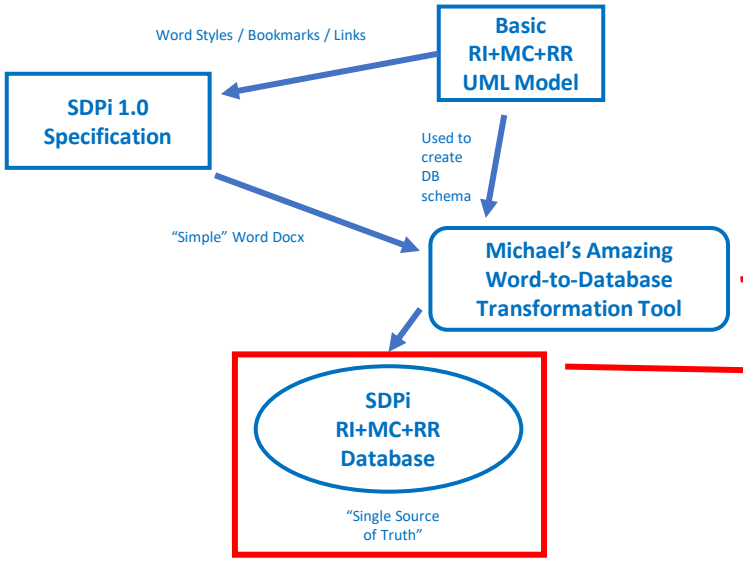
IEEE p11073-10700
(Draft 1 – *Under the Hood*)

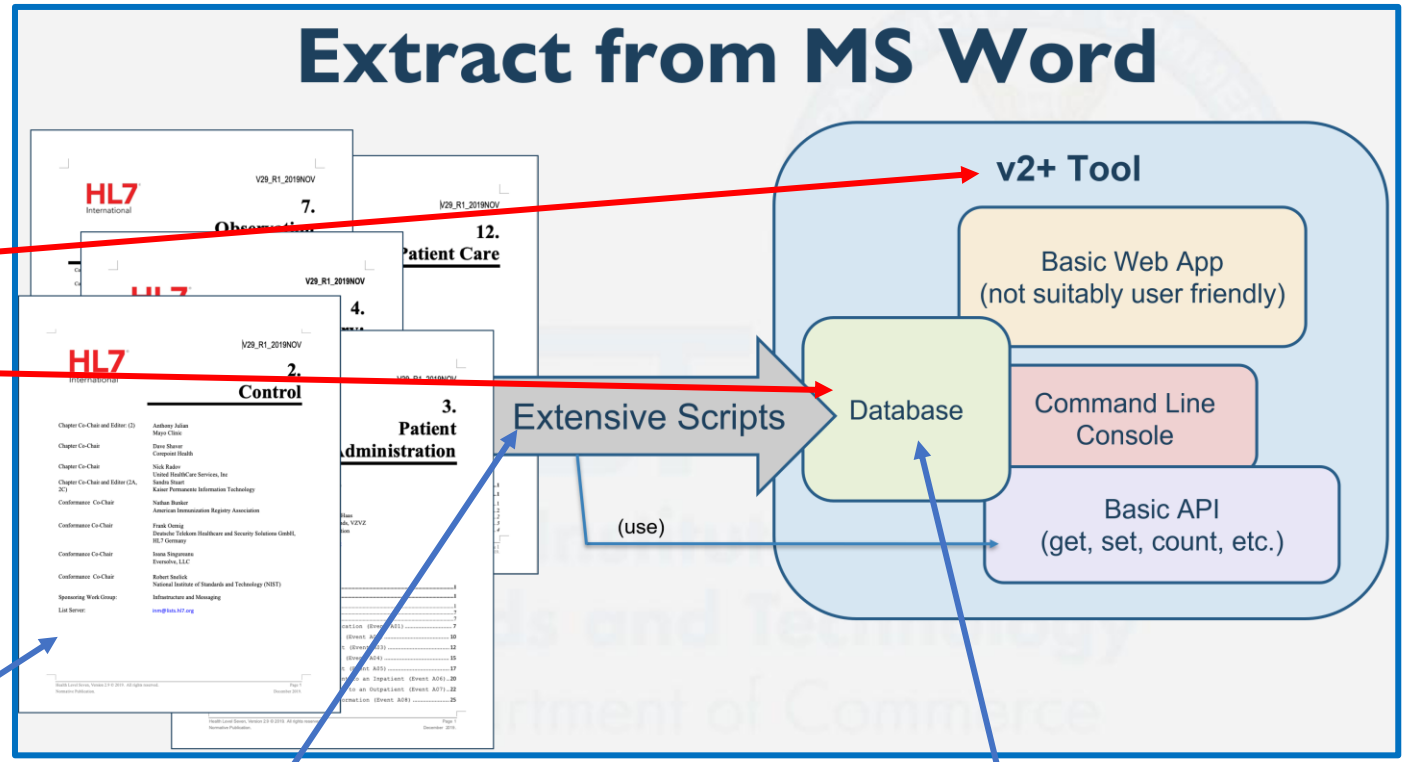
IEEE p11073-10700
(Draft 1)

19 **R0062: If an SDC PARTICIPANT produces EXCESSIVE LOAD CONDITIONs, each SDC**
20 **PARTICIPANT that is affected by these conditions SHALL maintain its SYSTEM FUNCTION**
21 **CONTRIBUTIONs for other SDC PARTICIPANTs.**

Observation: ReqIF Used for Document Creation –
Not requirements formalization!

Spec-to-Test Strategy – *To a SST Database*





Source: Michael Faughn 2022 January WGM Update / V2plus Project

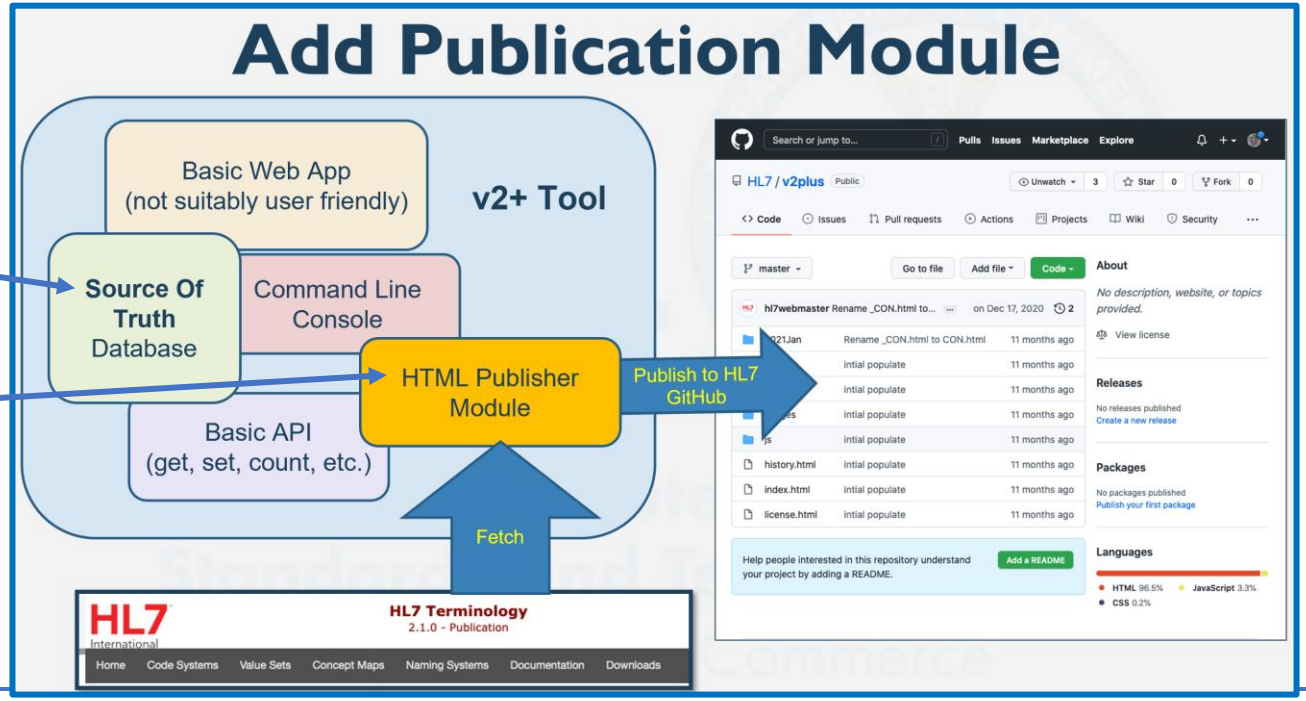
Road to an SDPi 1.0 Single Source of Truth Database:

- Build SDPi 1.0 Word document w/ content aligned to *basic* RI+MC(+RR) UML model
- Utilize "Extensive Scripts" to extract content from Word documents to Database
- Utilize UML Model to create database schema

Spec-to-Test Strategy – *SDPi 1.0 Specification*

SDPi 1.0 Specification

- **Word doc integrating "basic" RI+MC+RR support** (see preceding slides)
- **Constrained to roadmap capabilities**
- **Published as PDF (traditional)**
- **Supporting RI import to database**
  (See next slide)
- **Pilot Goal: HTML Publication**
  (may require integration of IHE TF elements to UML model – beyond "basic" RI+MC+RR)



Source: Michael Faughn 2022 January WGM Update / V2plus Project

Spec-to-Test Strategy – *Word Open Issues*

Word-approach Challenges …

- ✓ Finalize examples for each of the RI "types", including:
  - Use Case Based Requirements
  - Normative Referenced Standards "ICS" Requirements
  - Technical vs. Process (EP) requirements
  - SES Requirements
  - Technical-to-Technical Requirements
- ✓ Define "systematic name" scheme for Styles / Bookmarks / Links
- ✓ Finalize UML models for RI +
- ✓ "Extensive Scripts" for extracting RI information into a SST Database
- ✓ …

Gemini SES+MDI –
2022 RI+MC+RR Strategy –
ITI Use of Markdown for Publication

…ITI currently uses Markdown to format its specification, including the expansive ITI technical framework and supplement documents. These are then published to profiles.net.net utilizing HL7's IGPublisher tool, from the associated ITI and PUB github repositories. How does that work? Can IHE DEV SDPi utilize the same Markdown format without also leveraging the IGPublisher?

Spec-to-Test Strategy – *ITI TF & Markdown*

Considering that IHE TF publication – especially for ITI TF – is now:

- ✓ Markdown based (aligned with HL7 FHIR content OR simple IHE TF markdown template)
- ✓ Content either morphed from Word (existing profiles) / recommended hand-crafted markdown
- ✓ Utilizes the FHIR IGPublisher for conversion to HTML
- ✓ Publishes to a HTML-navigatible ("short form") at profiles.ihe.net/iti
- ✓ Recommendations for future IHE specification work is to leverage this process & tooling

How would the SDPi 1.0 TF utilize this current IHE approach?

Resources to review …

- IHE publication information / github wiki topics are github.com/IHE/publications/wiki
- The ITI index page is profiles.ihe.net/ITI/
- The ITI github repo is github.com/IHE/ITI
- Examples for creating a new repo for a specific supplement for the DEV TF …
  - https://github.com/IHE/ITI.mCSD (created via IG Publisher – FHIR-based profile)
  - https://github.com/IHE/ITI.IUA (created using GitHub and Markdown – non-FHIR-based profile)

Spec-to-Test Strategy – *From Publications Wiki*

Overall steps

Given a "development Committee" has an approved work item to develop a long-form Supplement or Whitepaper. Where "development Committee" might be a Domain Technical committee, Domain Planning committee, or Regional Deployment committee.

The overall approach is:

1. development Committee development of content
2. development Committee review of content
3. development Committee approval of content
4. Publications Committee preparing html output (of content, and navigation to the content)
5. Publications Committee, Co-Chairs, and author reviewing staging of html output
6. Publications Committee migrating html output to profiles.ihe.net
7. Publications Committee and development Committee tagging the GitHub repositories
8. Publications Committee announcement to general public

*

Home

John Moehrke edited this page on Mar 30 · 6 revisions

Welcome to the publications wiki!

Guidance Articles

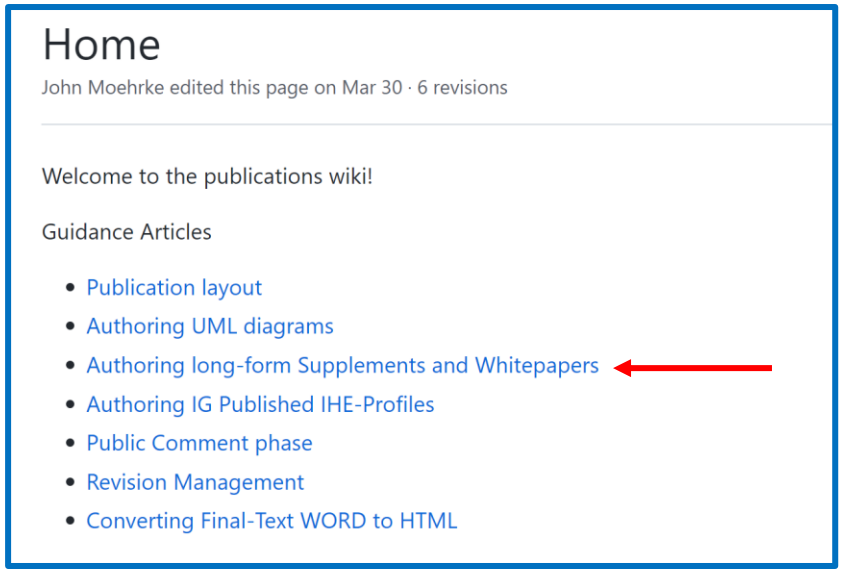
- Publication layout
- Authoring UML diagrams
- Authoring long-form Supplements and Whitepapers
- Authoring IG Published IHE-Profiles
- Public Comment phase
- Revision Management
- Converting Final-Text WORD to HTML

* ❑ DEV / Gemini SDC/SDPi Team Responsible for crafting +
❑ AsciiDoc-based Content can be easily integrated into IHE github repo's & rendered to ihe.github.io for domain-internal + public comment review

Spec-to-Test Strategy – *From Publications Wiki*

Long-form / Short-form

- ✓ Long-form = IHE "Classic" Documents (published as a single HTML page)
- ✓ **SDPi – *Short-form only*** *(file granularity? … see next)*
- ✓ ***IHE DEV TF 2022 Edition?*** *(use AsciiDoc … easier future integ.)*



Markdown & GitHub Integration

- ✓ *Markdown* & *Github* vs. Word & Google Drive
- ✓ Repo Name: DEV.SDPi or DEV.SDPi-P (refactor sdpi-fhir for now)
- ✓ Publish repo to "Github Pages" (ihe.github.io/)
- ✓ Handcraft markdown content in GitHub Editor (can use markdown WYSIWYG editor but …)
- ✓ Markdown ➔ HTML via PanDoc

  Note: *PanDoc supports AsciiDoc as well*

```
pandoc -r gfm --template=https://github.com/IHE/publications/wiki/ihe_template.html --metadata title="HIE-Whitepaper" --metadata path-prefix="../../" -w html -o index.html README.md
```

- ✓ Pre-pub review @ ihe.github.io/publications (via repo "Pages")
- ✓ Published via FileZilla to profiles.ihe.net/

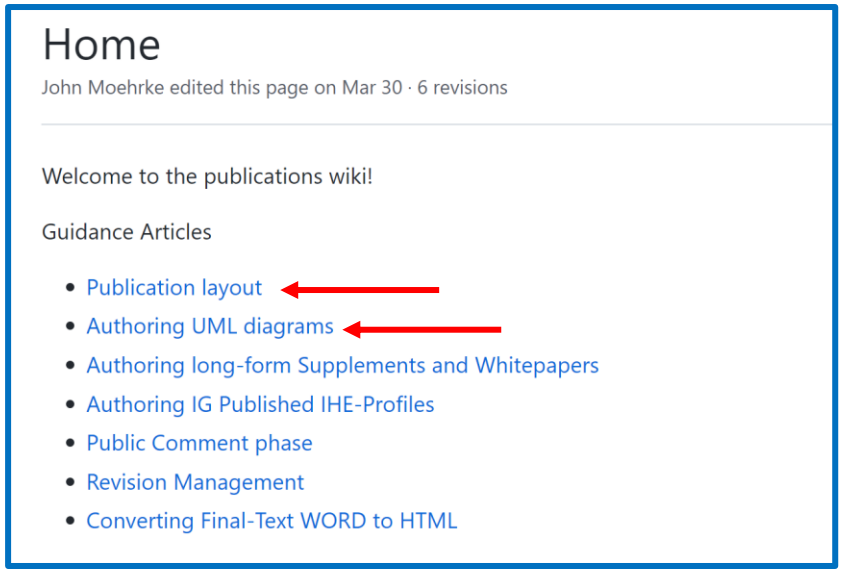
Spec-to-Test Strategy – *From Publications Wiki*

Publication Layout

- ✓ Provides folder structure @ profiles.ihe.net
- ✓ TF Volumes + Supplements + Papers + non-domain specific profiles.ihe.net/GeneralIntro

Authoring UML Diagrams

- ✓ Use PlantUML (.pu / .plantuml)
- ✓ Include a link to the .pu file next to the diagram
  - ✓ Note: auto rendering not currently supported – given potential risk of variation after publication with changes in underlying infrastructure
- ✓ File naming:
  - Figure 8.6.4-2 (per IHE naming conventions)
  - media/Figure_8.6.4-3.png & media/Figure_8.6.4-3.pu
- ✓ IGPublisher – Call for SVG image …
- ✓ HTML Example ➔



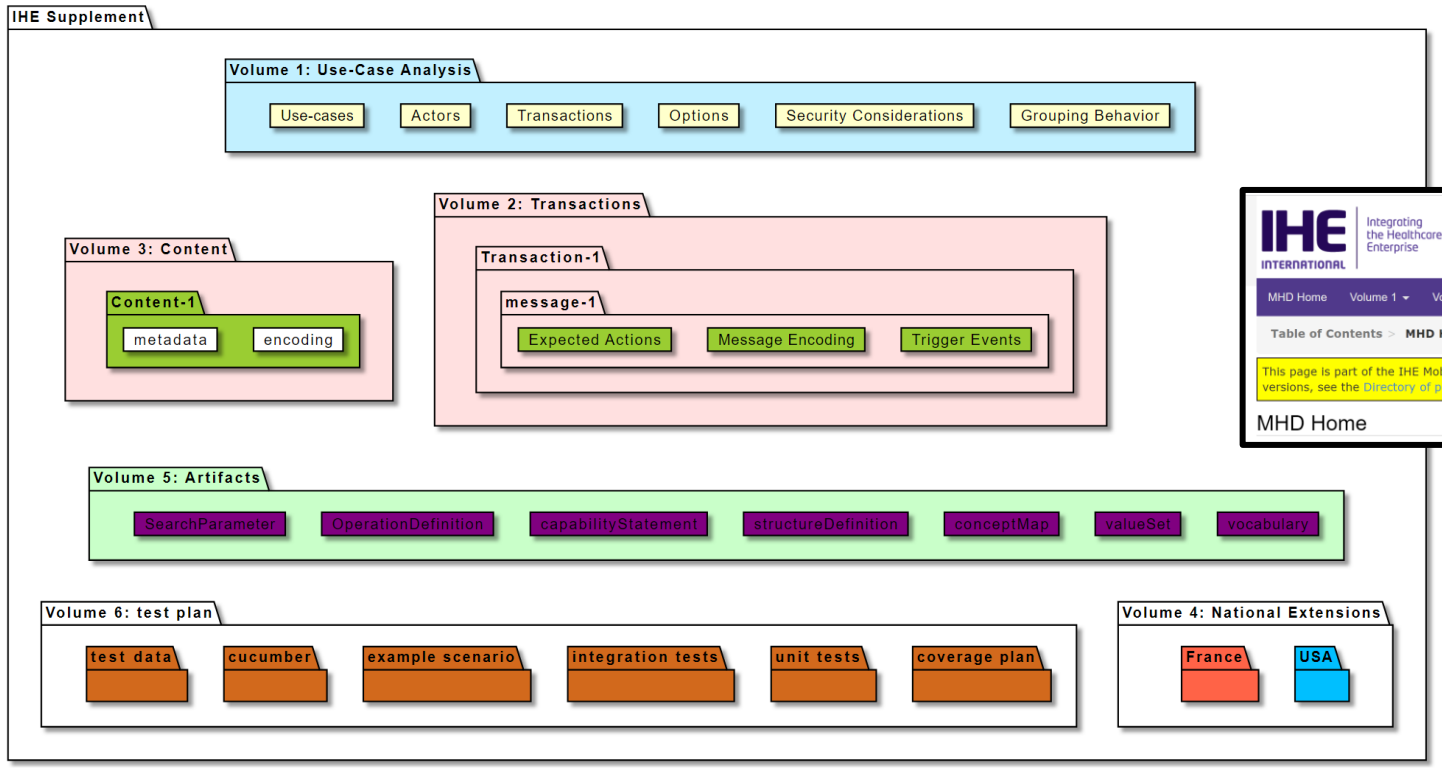

```
<h5 id="26.4.2.1.2">26.4.2.1.2 Unexpected Notification Process Flow</h5>
<p class="figureTitle">
  <img src="media/Figure_26.4.2.1.2-1.png" alt="" />
  Figure 26.4.2.1.2-1: Interaction Diagram for Unexpected Notification Use Case
</p>
<p class="note">
  <a href="media/Figure_26.4.2.1.2-1.pu">UML source for Figure 26.4.2.1.2-1</a>
</p>
```

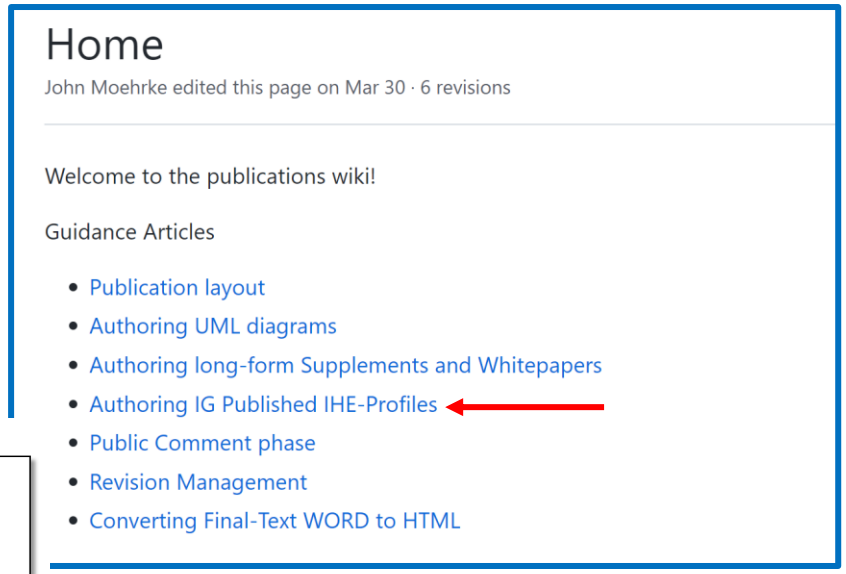
Source: github.com/IHE/publications/wiki (drafted by John Moehrke)

Spec-to-Test Strategy – *From Publications Wiki*

See: *Active IHE Projects Using the IG Builder*

Supplement organization supported by IG Publisher …





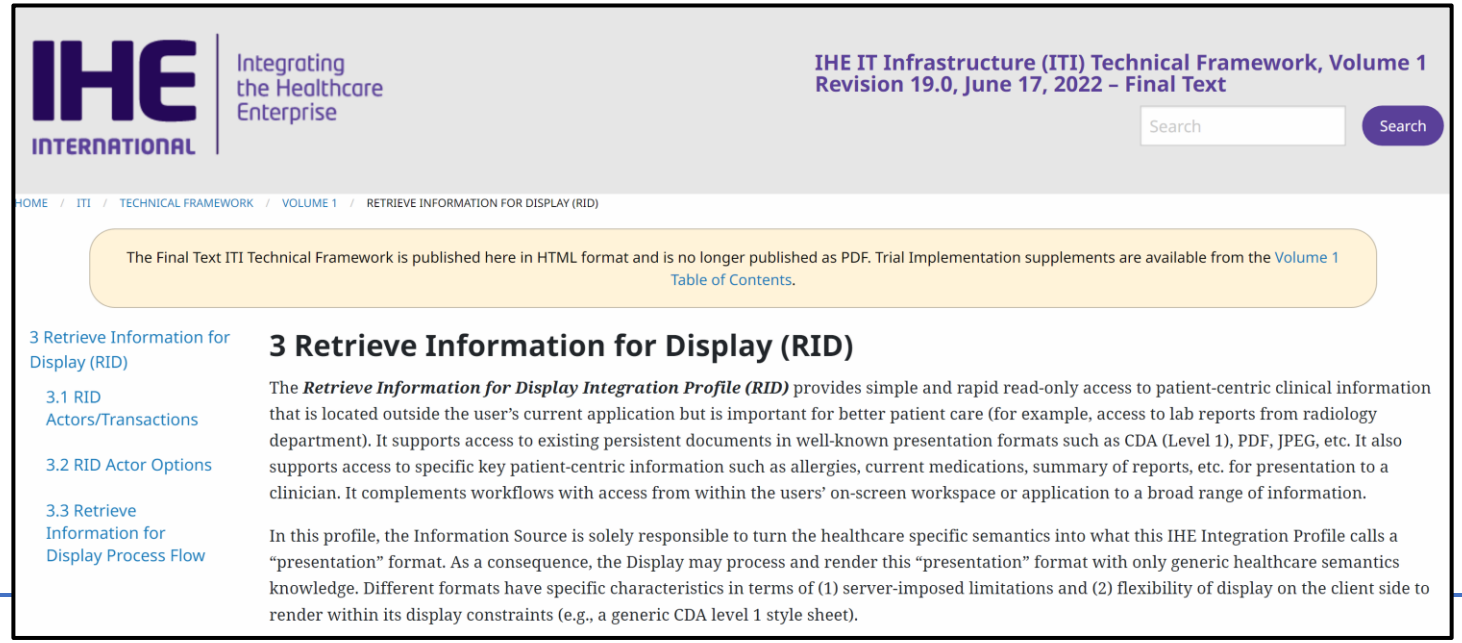
See IHE supplement template @ build.fhir.org

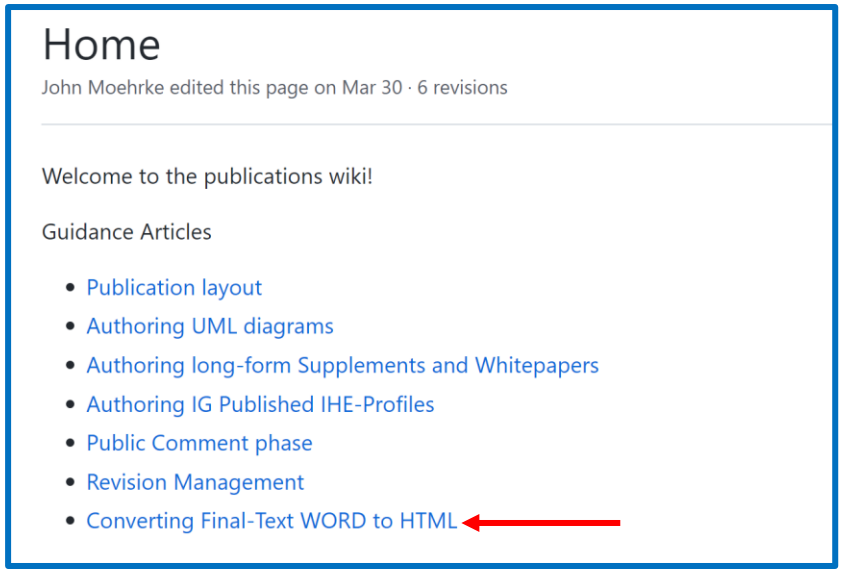
MHD Source @
https://github.com/IHE/publications/tree/master/ITI/TF

Spec-to-Test Strategy – *From Publications Wiki*

IHE Supplement Template in markdown:
raw.githubusercontent.com/wiki/IHE/supplement-template/suppl_template.md

See example ITI IUA Profile:
https://github.com/IHE/ITI.IUA/blob/master/IHE_ITI_Suppl_IUA.md
(uses gmf – github markdown format)





See the extended TF structure @
https://github.com/IHE/publications/tree/master/ITI/TF

(IHE ITI TF is ONLY maintained in HTML, no Word or markdown sources – since FT generally requires few and limited changes)

Spec-to-Test Strategy – *Markdown Open Issues*

Markdown-approach Challenges …

- ✓ Can the use of markdown in the IHE publication (esp. ITI TF) process & tool chain be utilized for SDPi 1.0 purposes? Into the future? ***[yes, but why…]***
- ✓ Can the general "short form" architecture be leveraged for SDPi specification purposes? (w/o sacrificing RI + w/o More-Hacking-Required) ***[short form – yes; markdown – no]***
- ✓ Can the current IHE TF (esp. ITI TF) production & publication process & tool chain be leveraged but utilizing AsciiDoc? ***[Generally – yes; IG Pub? No]***
- ✓ Can an "enhanced" approach for SDPi be established that will be … "welcomed" … by IHE stakeholders, especially publications? ***[Stay tuned!]***
- ✓ …

Also: **IHE Profiles** vs. **IHE Implementation Guides … *Same*? *Different***?

Markdown/kramdown – *What is it?*

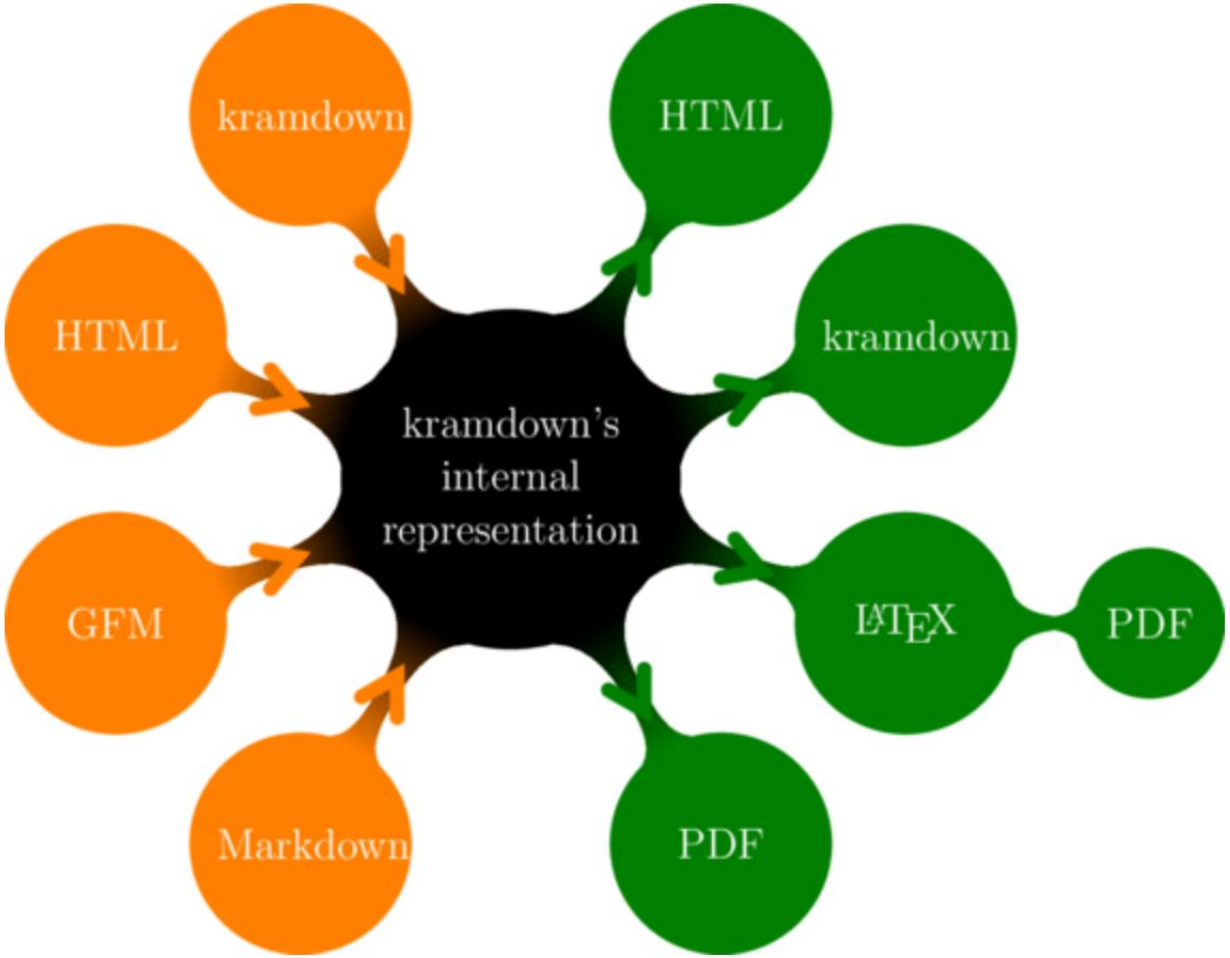
Used by HL7 FHIR & IHE FHIR-Based Publications …

kramdown

fast, pure-Ruby Markdown-superset converter

… is a ***TOOL*** for processing markdown "flavors"
+ for providing some extension capabilities

(note: kramdown claims to be a "superset" of markdown … but not a markdown flavor … ???)



See: [kramdown Quick Reference]

Source: [kramdown home page]

Gemini SES+MDI – 2022 RI+MC+RR Strategy –

ITI Special Pages & DEV OID Scheme

One approach to publishing the updated IHE Devices OID scheme, including extensions supporting the SDPi profiles, is to leverage the approach that ITI developed for publishing tables of various identifiers and value sets.

IHE DEV OID Publication – *Leverage ITI Approach?*

Given the need to update and re-publish the IHE DEV OID allocations …

Question:

Should we use an approach analogous to what IHE / ITI uses today (e.g., IHE FormatCode identifiers)?

Example: https://profiles.ihe.net/fhir/ihe.formatcode.fhir/ValueSet-formatcode.html

Github Source: https://github.com/IHE/FormatCode/tree/master/input/resources

Answer:

Yes, but in a way that is consistent with the DEV TF & SDPi Supplement

Gemini SES+MDI – 2022 RI+MC+RR Strategy –

Door #3: AsciiDoc

In addition to using a Word-based document or a Markdown-based document, AsciiDoc is a 3rd option that may be better able to author the technical SDPi profile specifications, as well as all artifacts created / generated in the SDC/SDPi+FHIR Gemini program.

Option #3: What is: *AsciiDoc?*

"AsciiDoc is a plain text markup language for writing technical content. It's packed with semantic elements and equipped with features to modularize and reuse content. AsciiDoc content can be composed using a text editor, managed in a version control system, and published to multiple output formats."

[AsciiDoc]

Note: ***AsciiDoc is NOT a markdown flavor*** – it is a language purpose-built for technical specifications, with an extensive support community and tooling integration.

Source: David Gregorczyk / Dräger

Option #3: AsciiDoc – *Comparisons?*

Markdown

index.md

```
# Heading 1

A paragraph with **bold** and *italic* text.
A link to [Eclipse](https://eclipse.org).
A reusable link to [GitLab](gitlab).

![An image](an-image.png)

## Heading 2

* Unordered list item
  * Nest items by aligning marker with text of parent item
* Another unordered list item

**NOTE:** An admonition can be emulated using a bold label.

### Heading 3

    Text indented by four spaces is preformatted.
```

Markdown is a **lightweight** markup language for producing HTML. Markdown builds on basic **plain text conventions** for formatting content. While approachable to a broad audience, it **stops short** of being a technical writing language. The need for **syntax extensions** quickly enters the picture. In reality, Markdown is the basis for a variety of markup languages that often **deviate widely**.

AsciiDoc

index.adoc

```
= Document Title
:toc:
:url-gitlab: https://gitlab.eclipse.org

A paragraph with *bold* and _italic_ text.
A link to https://eclipse.org[Eclipse].
A reusable link to {url-gitlab}[GitLab].

image::an-image.png[An image,800]

== Section title

* Unordered list item
** Add another marker to make a nested item
* Another unordered list item

NOTE: One of five built-in admonition block types.

=== Subsection title

 Text indented by one space is preformatted.
```

AsciiDoc appears **strikingly similar** to Markdown, making way for an easy transition. Where AsciiDoc shines is in **its depth**. AsciiDoc provides all the essential elements in **technical writing** out of the box. **No variants** needed. Its syntax can be elaborated without having to fundamentally change the language, assuring users that it's still **standard AsciiDoc**.

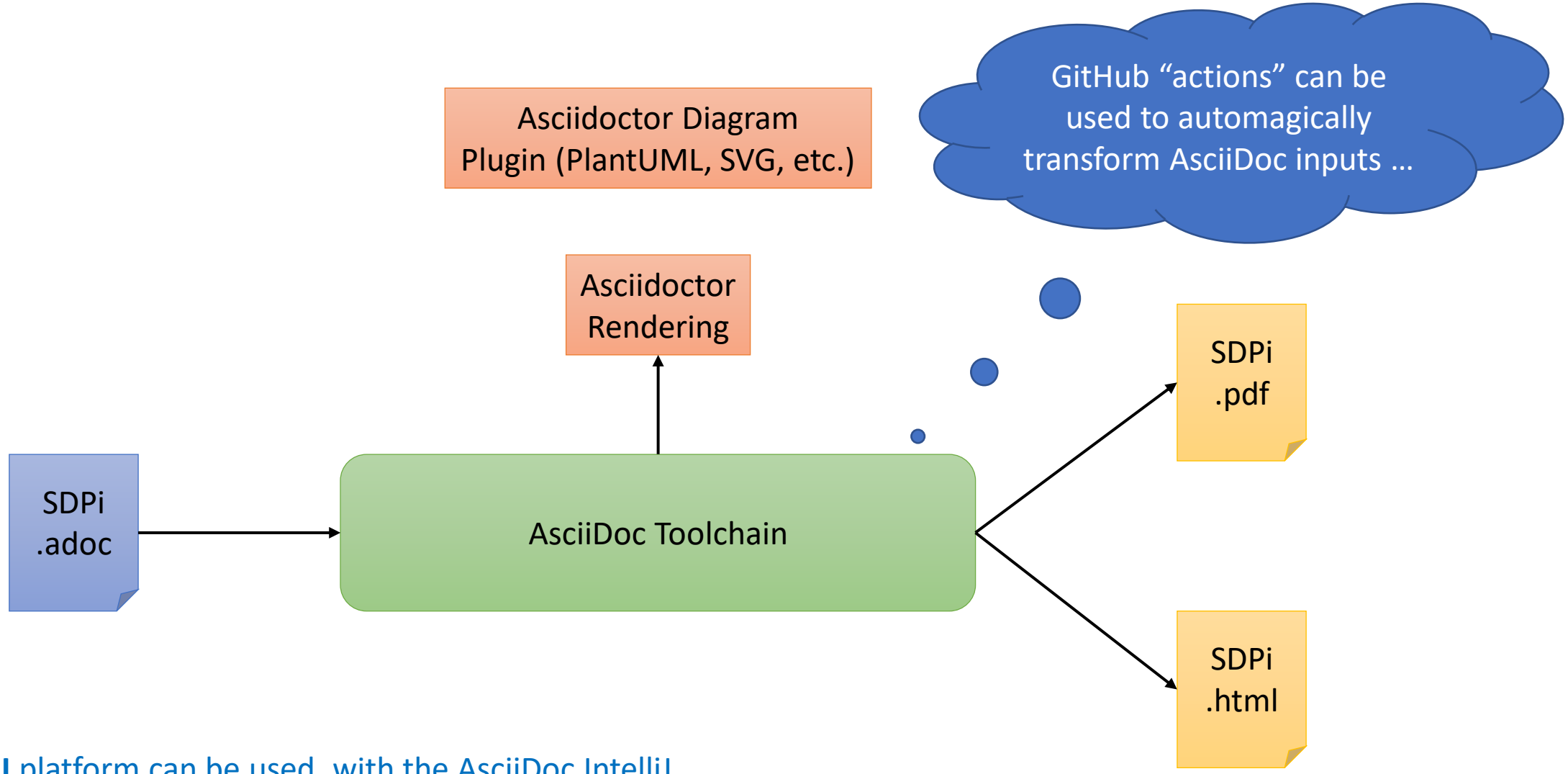
HL7 FHIR & IHE use:

markdown

To support integration with HL7 FHIR Tooling

Source: [AsciiDoc.org / "HOW ASCIIDOC STACKS UP"]

Option #3: AsciiDoc – *How does it work?*



Note: **intelliJ** platform can be used, with the AsciiDoc IntelliJ plugin + Asciidoctor Diagram (to render PlantUML)

Source: David Gregorczyk / Dräger

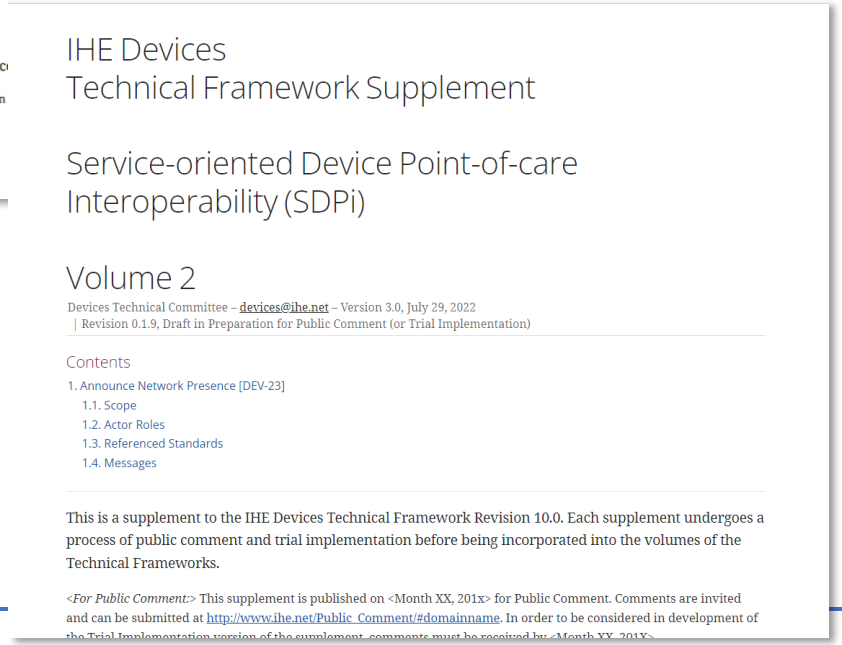
Option #3: AsciiDoc – *Artefacts?*

```
= IHE Devicespass:q[<br>]Technical Framework
upplementpass:q[<br>]pass:q[<br>]Service-oriented Device Point-of-care
Interoperability (SDPi)pass:q[<br>]pass:q[<br>]Volume 2
Devices Technical Committee <devices@ihe.net>
3.0, July 29, 2022: Revision 0.1.9, Draft in Preparation for Public Comment
 (or Trial Implementation)
:doctype: book
:preface-title: Preface
:toc-title: Contents
:toc:
:sectnums:
:icons: font
:title-logo-image: image::ihe-logo.png[pdfwidth=4in,align=center]

This is a supplement to the IHE Devices Technical Framework Revision 10.0.
 Each supplement undergoes a process of public comment and trial
 implementation before being incorporated into the volumes of the Technical
 Frameworks.

__<For Public Comment:>__ This supplement is published on <Month XX, 201x>
 for Public Comment. Comments are invited and can be submitted at http://www
 .ihe.net/Public_Comment/#domainname. In order to be considered in
 development of the Trial Implementation version of the supplement, comments
  must be received by <Month XX, 201X>.

__<For Trial Implementation:>__ This supplement is published on <Month XX,
 201X> for Trial Implementation and may be available for testing at
 subsequent IHE Connectathons. The supplement may be amended based on the
 results of testing. Following successful testing it will be incorporated
 into the Devices Technical Framework. Comments are invited and can be
```

To PDF

To HTML

I am edited in IntelliJ, VS Code, or just in a Text Editor



IHE Devices
Technical Framework Supplement

Service-oriented Device Point-of-care
Interoperability (SDPi)

Volume 2

Devices Technical Committee – devices@ihe.net – Version 3.0, July 29, 2022
| Revision 0.1.9, Draft in Preparation for Public Comment (or Trial Implementation)

Contents

1. Announce Network Presence [DEV-23]
   1.1. Scope
   1.2. Actor Roles
   1.3. Referenced Standards
   1.4. Messages

This is a supplement to the IHE Devices Technical Framework Revision 10.0. Each supplement undergoes a process of public comment and trial implementation before being incorporated into the volumes of the Technical Frameworks.

<For Public Comment:> This supplement is published on <Month XX, 201x> for Public Comment. Comments are invited and can be submitted at http://www.ihe.net/Public_Comment/#domainname. In order to be considered in development of

Gemini SES+MDI –
2022 RI+MC+RR Strategy –
Selection: Word / Markdown / AsciiDoc

Given the primary three options for SDPi specification content creation to publication, how do they compare today? Which should be selected for finalizing the SDPi 1.0 specification content?

Spec-to-Test Strategy – *Why AsciiDoc?*

AsciiDoc was selected as the best path forward for SDPi … why?

- ✓ ***Word is a short term "hack" at best …***
  - IHE has already moved from Word to markdown & HTML publication
- ✓ ***Markdown was created for simple commenting and on-line content –***
  - Publication in PDF and HTML works but support for RI+MC+RR … not so much
- ✓ ***AsciiDoc provides the best approach today for advancing to RI+MC+RR…***
  - Strong built-in support for embedding metadata + defining type / object extensions
  - Can support model-based content now, better supporting integration into a single-source of truth specification database
  - Can support ALL the document-based artifacts created / consumed by the SDPi+FHIR program … including test-related assertions, scripts, and reports
  - Broad tooling support and platform integration + support community is equivalent to that of markdown
  - Content editors today will have to use either markdown or AsciiDoc or similar – pathway for integration of other sources (e.g., Word) to AsciiDoc is easily achieved
  - Support for Gherkin and other extensions already in place

Spec-to-Test Strategy – *Moving to AsciiDoc*

How will the transition to AsciiDoc be achieved? Next steps?

- ✓ From the 7/15 SDPi Discussion …
  - Use ***github sdpi-fhir wiki pages*** to start documentation of the specification creation to publication approach
  - Start with a **simple *"How to author and generate AsciiDoc specs?"*** for newbies (including test editors, platforms (IntelliJ & VS Code ...) etc.
  - Add AsciiDoc base (master) file (<supplement name>.adoc main file) and (tbd) subfolders to the github SDPi-FHIR repo folder: SDPi Supplement / SDPi Rev 1.0 /
  - Add initial AsciiDoc_Support folder to collect items, including David Gregorczyk 's demo program project
  - Rename folders in this tree to replace spaces with underscores
  - ***Define the kinds of AsciiDoc blocks*** that will be needed for the specification (e.g., per IHE TF specifications and artifacts)
  - Include those needed for *requirements interoperability* & links to *testable assertions*
  - Create an ***initial set of AsciiDoc templates*** to support the "blocks"
  - ***Configure github actions to support production*** of the specification, both in a "long form" PDF document as well as in an HTML "small form" / multi-file publication
- ✓ Begin migration of content to AsciiDoc "platform" starting with ***simple end-to-end use case based threads***
- ✓ Explore use of AsciiDoc content for ***Test Scripting & Test Reporting***
- ✓ Question: *For requirements in SDPi 1.0, can SysML 2.0 KerML be used to define the requirements NOW, at least in the UML modeling, and thus better supporting migration to more robust MC specifications in the future?*

Spec-to-Test Strategy – *Next Steps++*

Based on discussions & decisions per IHE publications process & tooling review …

- ✓ *Create initial AsciiDoc-based content in the sdpi-fhir repo*
  - *SDPi Supplement folder*
  - *Update and archive the content under the "SDPi Rev 1.0" folder*
- ✓ *Content Authoring Platform:*
  - *Many AsciiDoc tools exist, but for a starting pass … see "tool chain" slide above:*
    - *intelliJ platform can be used (open "Community Edition), with the AsciiDoc IntelliJ plugin + AsciiDoctor Diagram plugin (to render PlantUML)*
- ✓ *Use sdpi-fhir repo wiki to document process and tooling being utilized*
  - *As is done with the current IHE Publications and related github wiki pages*
- ✓ *Configure sdpi-fhir repo "Pages" to auto-publish to ihe.github.io [TBD]*
- ✓ *Rendering: To the greatest degree possible, specification content will be separated from rendering + focus 1st on HTML .. PDF if / when needed*

Spec-to-Test Strategy – *Next Steps++*

- ✓ *Content File Structure*
  - *Multi-file approach that leverages both the current TF chapter level + IG Publisher level*
  - *Minimize "magic numbers" in the content (e.g., document outline sections embedded)*
  - *Support a Table of Contents but with URIs vs. pervasively embedded section #'s*
- ✓ *Use github …*
  - *"Actions" to automagically transform committed updates to rendered HTML*
  - *"wiki" to document content development processes & tool chain*
  - *"project" for migration of current Word version to initial baseline*
  - *"issues" to connect with Topics of Interest to resolutions to "Issues" in supplement*
  - *"Kanban" board to manage a To Do backlog*
- ✓ *Use PanDoc to convert Word to AsciiDoc …*
  - *1st pass at IHE DEV TF*
  - *Simple content from Word-based contributions to AsciiDoc for integration*
- ✓ ***AsciiDoc idiomatic paradigm will be fully utilized + github***

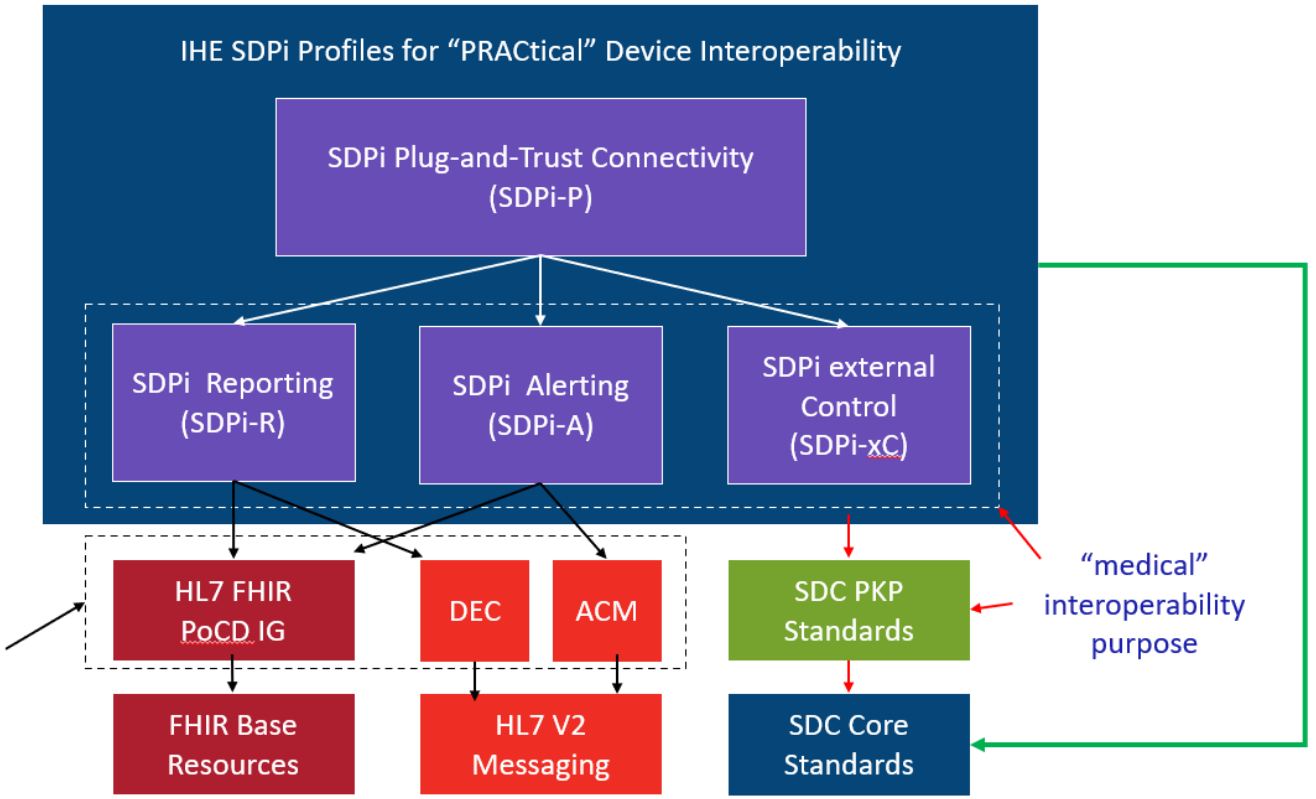
IHE-HL7 Gemini SES+MDI –
2022 RI+MC+RR Strategy–
Word – Markdown – AsciiDoc ???

Additional Materials

Orientation Tour: IHE TF & SDPi Profiles

Service-oriented Device Point-of-care Interoperability (SDPi)

- ✓ **Four profile specifications:**
  - SDPi-P for Plug-and-Trust Interoperability
  - SDPi-R for Reporting Medical Information
  - SDPi-A for Alerting
  - SDPi-xC for External Controlling
- ✓ **Three IHE DEV TF Volumes:**
  - TF-1 Profiles / use cases / actors / …
  - TF-2 Transactions / MDPWS messaging
  - TF-3 BICEPS content modules / device specializations



See draft SDPi Supplement Word Document for additional content detail & outline (https://github.com/IHE/sdpi-fhir/tree/master/SDPi%20%20Supplement/SDPi%20Rev%201.0)

Spec-to-Test Strategy – *Closing the Loop / EP Style*

Traceability & Coverage back to 11073-1070x PKP ICS Specifications (via "RI")

IEEE 11073-1070x Project Teams

Develop

11073-1070x PKP Standards

PKP ICS Tables

Gemini SES+MDI Community

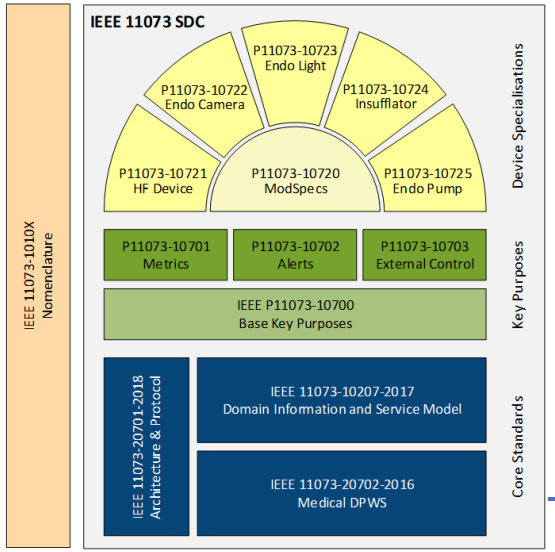
Develop

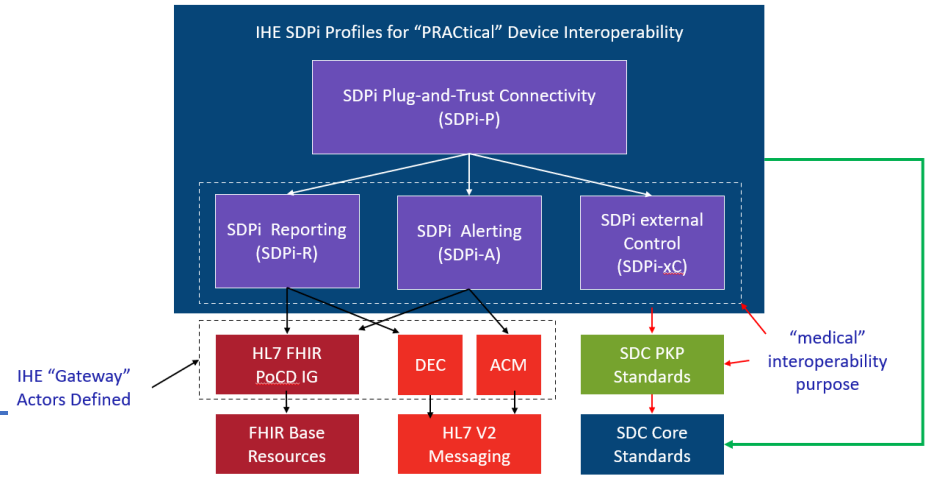
IHE SDPi TF Specifications (TF-1 / Appendix B)

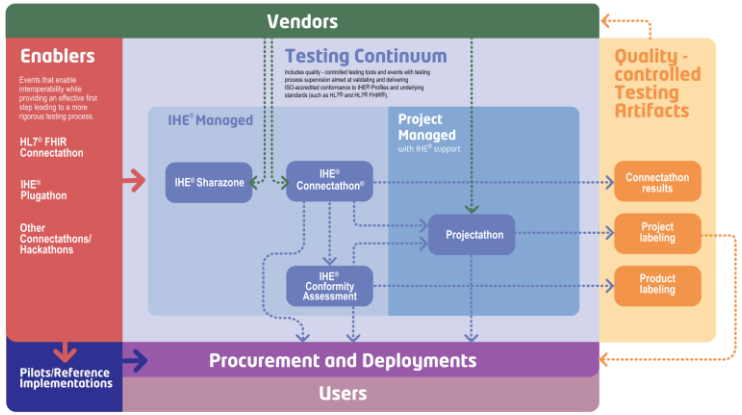
Require-ments

SES+MDI SDC/SDPi+FHIR Test Plans/Scripts + PAT/CAT + Product CA

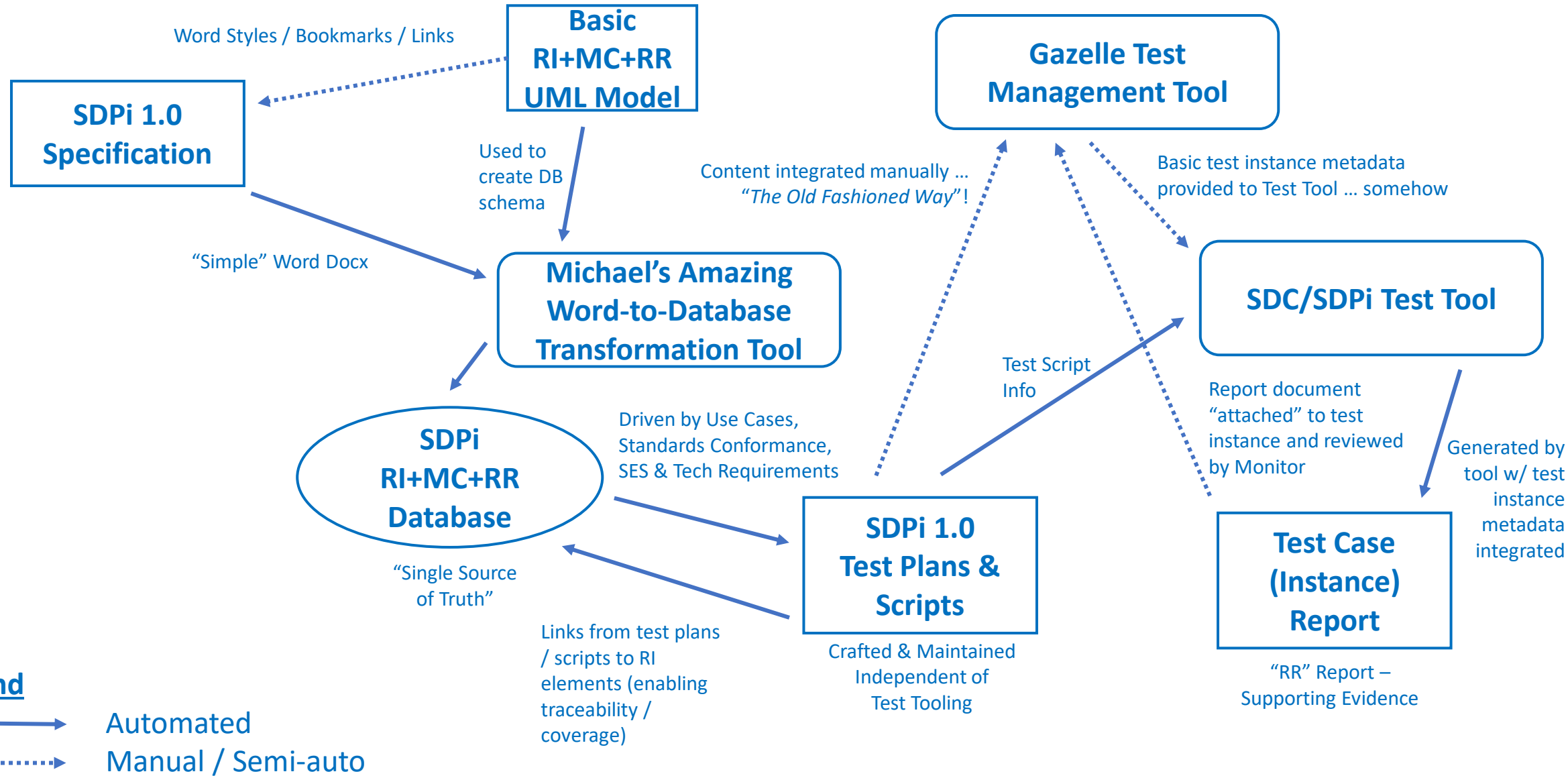
CA "RR" Test Report



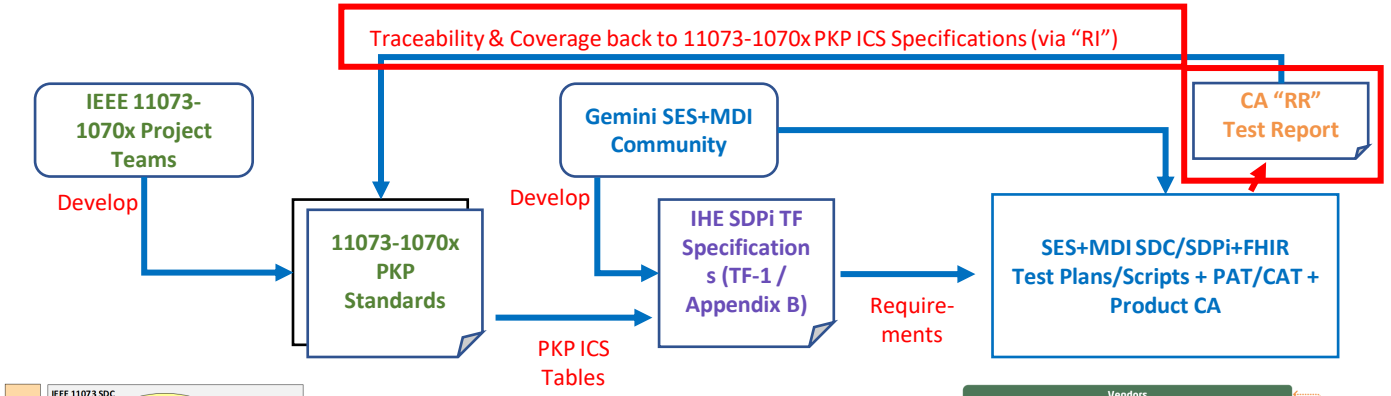




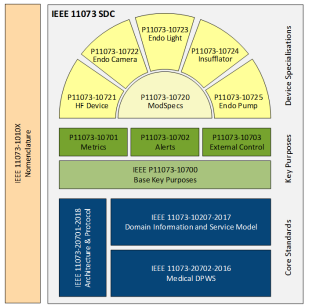
From Specs to Test Reports – *"Basic" Strategy*

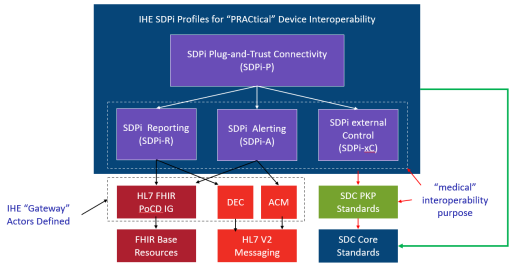


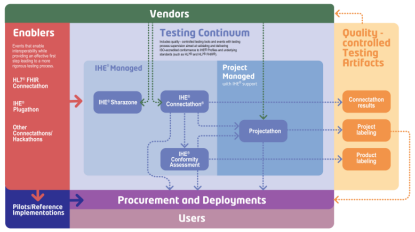
Spec-to-Test Strategy – *"RR" Test Reports?*



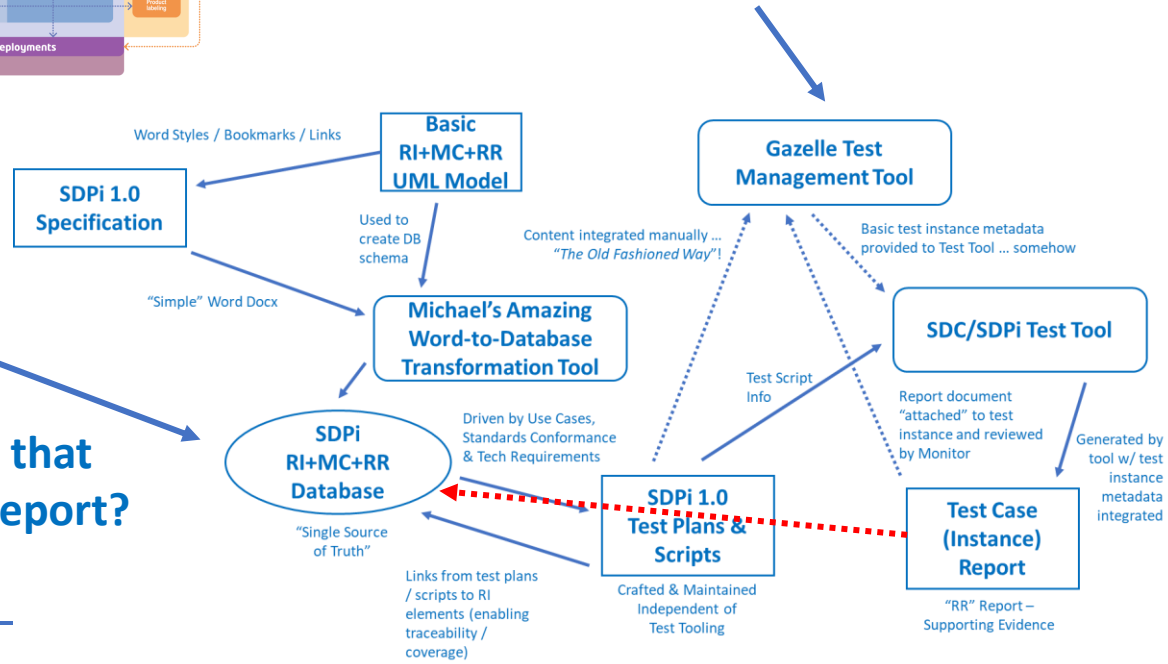
What is the strategy for getting to a "regulatory submission ready" product CA test report?







Test results currently are persisted in the Gazelle testing / CA database …

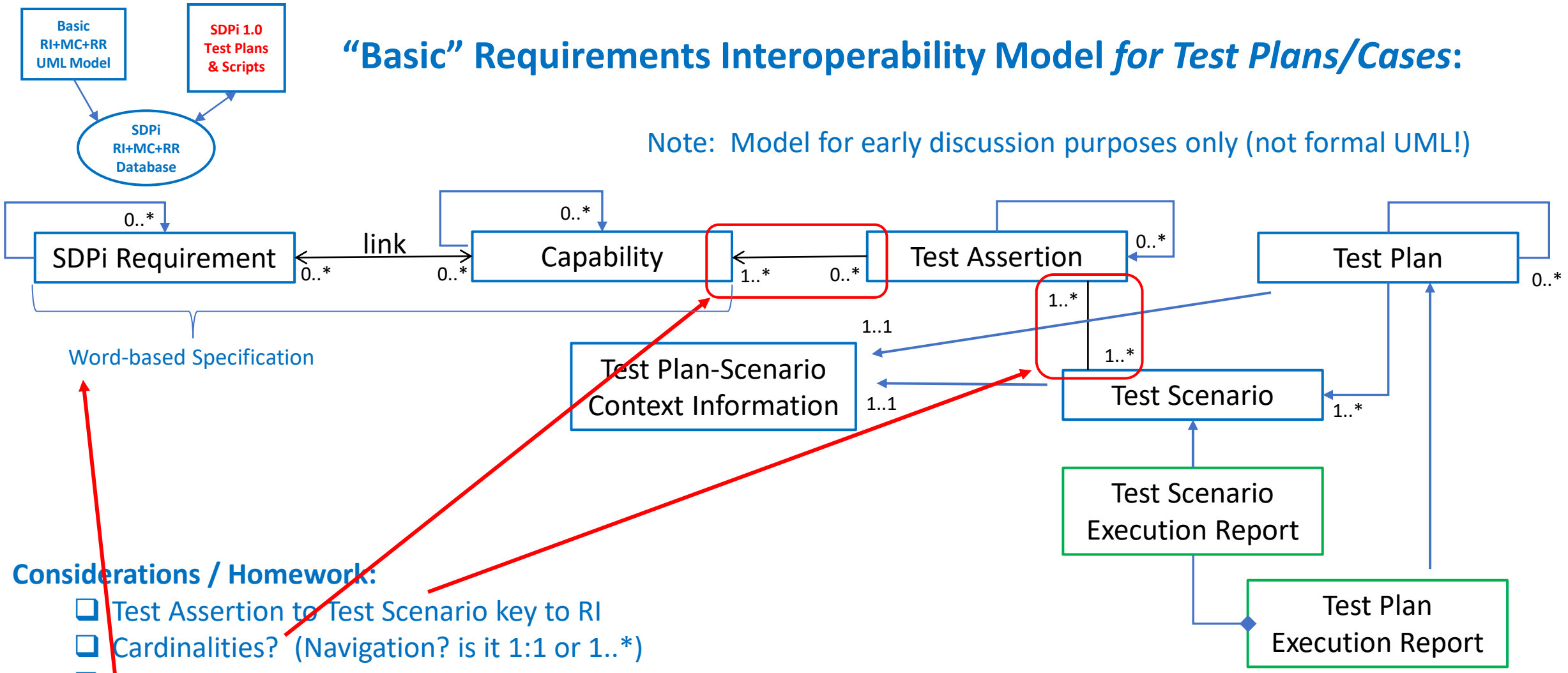


Should SDPi 1.0 test report results be incorporated into the SST database to link with RI-to-Test Case information?

Should a separate test results database be created? + A tool that can combine the multi-database content to create a RR CA report?

How are CA test reports created today? Historically?
Separate tool from Gazelle? Extended Gazelle instance?

Spec-to-Test Strategy – *RI+MC+RR Model*



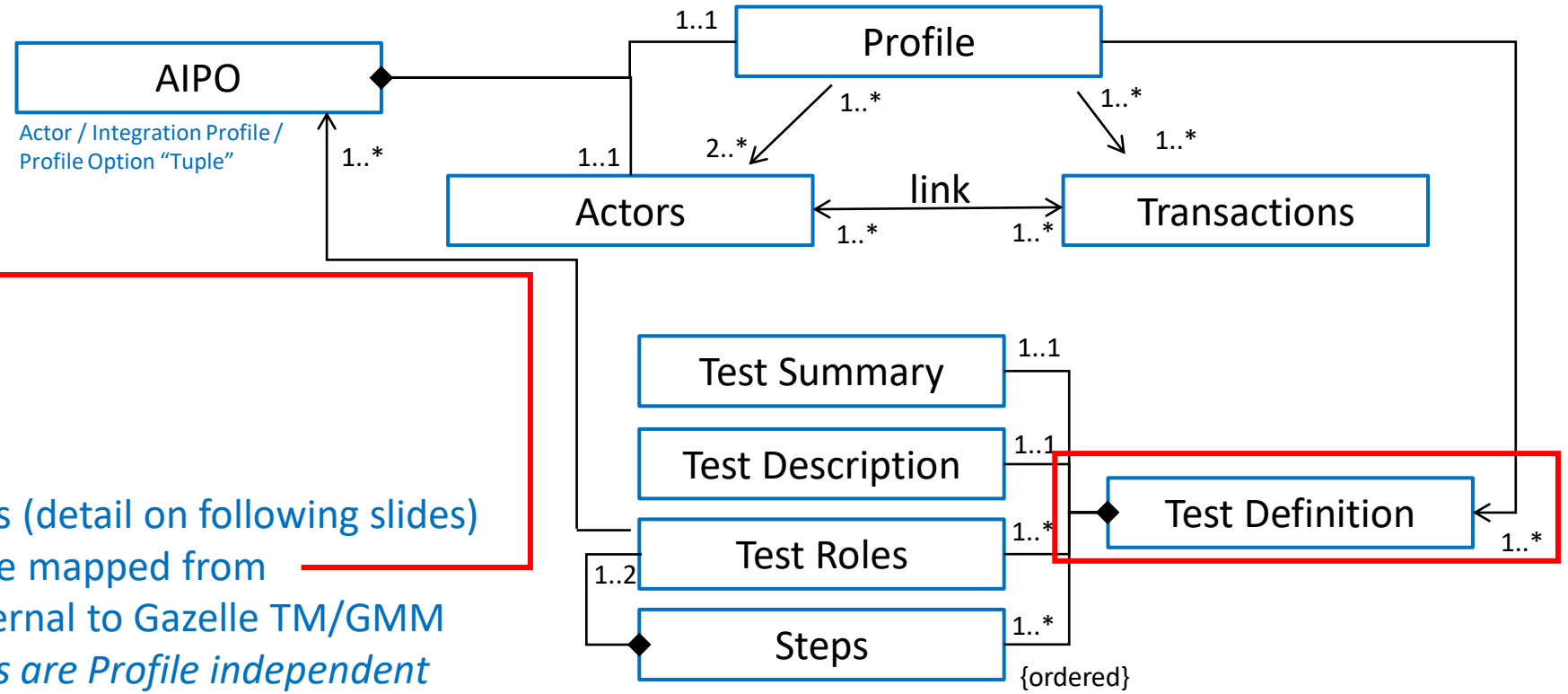
Spec-to-Test Strategy – *To Gazelle Integration*

Gazelle Test Management Tool

SDPi 1.0 Test Plans & Scripts

Gazelle Master Model (GMM) –

- GMM content leveraged by all instances of the test management tool
- All profile & test information entered manually via GMM UX



Gazelle Test Definitions –

- Consists of four components (detail on following slides)
- GMM Test elements must be mapped from
- Test automation mostly external to Gazelle TM/GMM
- Note: *Actors & Transactions are Profile independent*
- *Question: "link" cardinality 2..2 or 1..* or … ???*
- *Question: "links" are also "initiator" & "consumer"*

Source: GMM User's Guide