MELDfest day 1
03-Nov-2020

SOFA and MELD

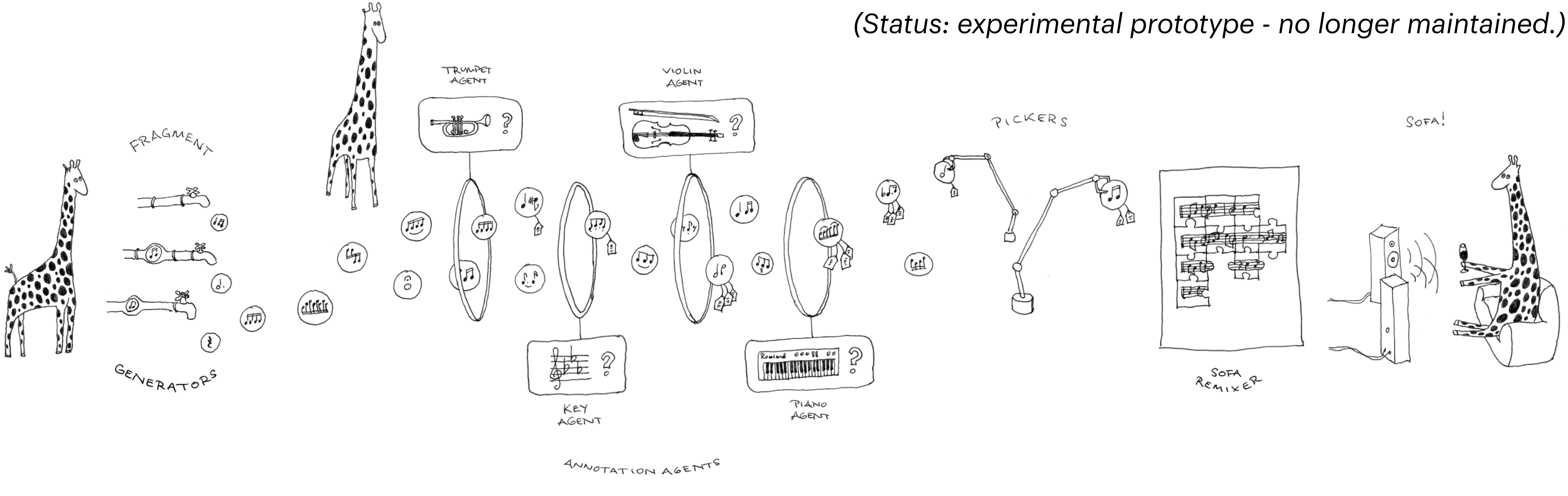
Introducing use of the MELD framework by SOFA

Graham Klyne - 2020-10-06
https://github.com/oerc-music/nin-remixer-public/blob/master/notes/20201004-SOFA-and-MELD.key

What is SOFA (1)

SOFA Ontological Fragment Assembler (SOFA) is an application for assembling elements, or “fragments”, into a composed whole.

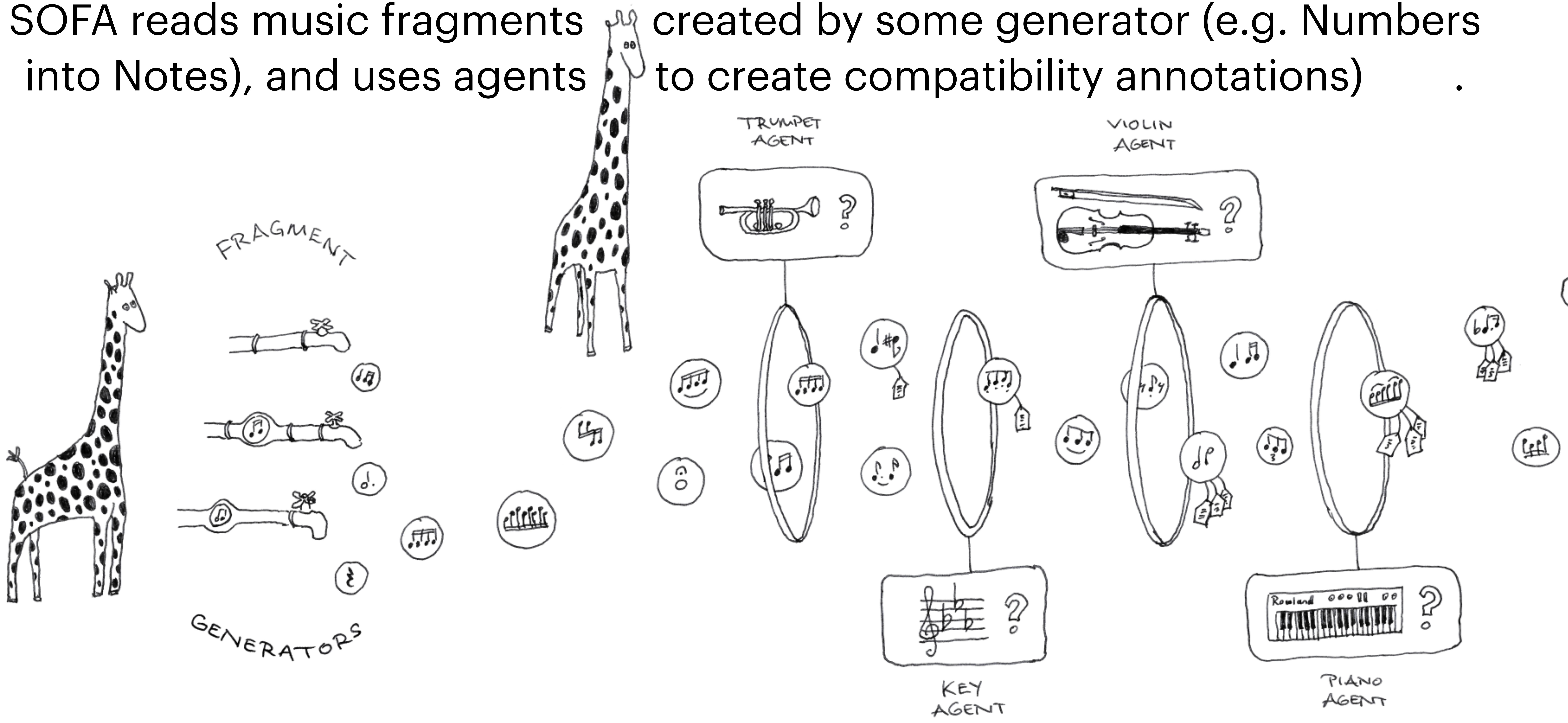
(Status: experimental prototype - no longer maintained.)



BEHIND THE SCENES AT THE SOFA REMIXER

What is SOFA (2)

SOFA reads music fragments created by some generator (e.g. Numbers into Notes), and uses agents to create compatibility annotations) .

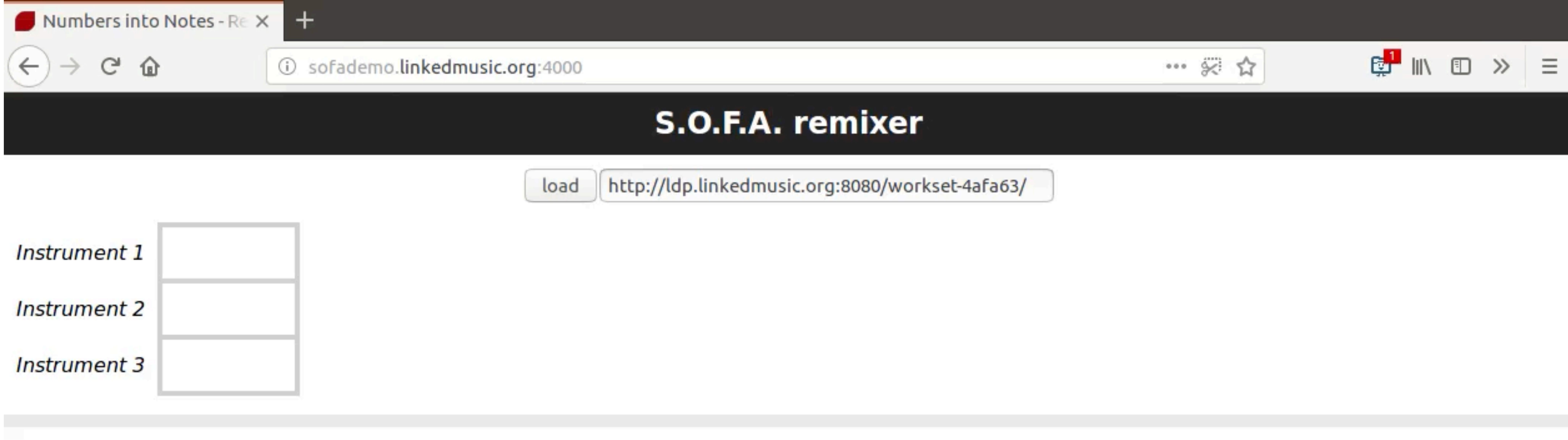


What is SOFA (3)

SOFA also provides an interface for guided assembly of the fragments into musical compositions, based on the compatibility annotations

VIOLIN AGENT
PICKERS
SOFA!
KEY AGENT
PIANO AGENT
SOFA REMIXER
Roland

SOFA demo video



S.O.F.A. remixer

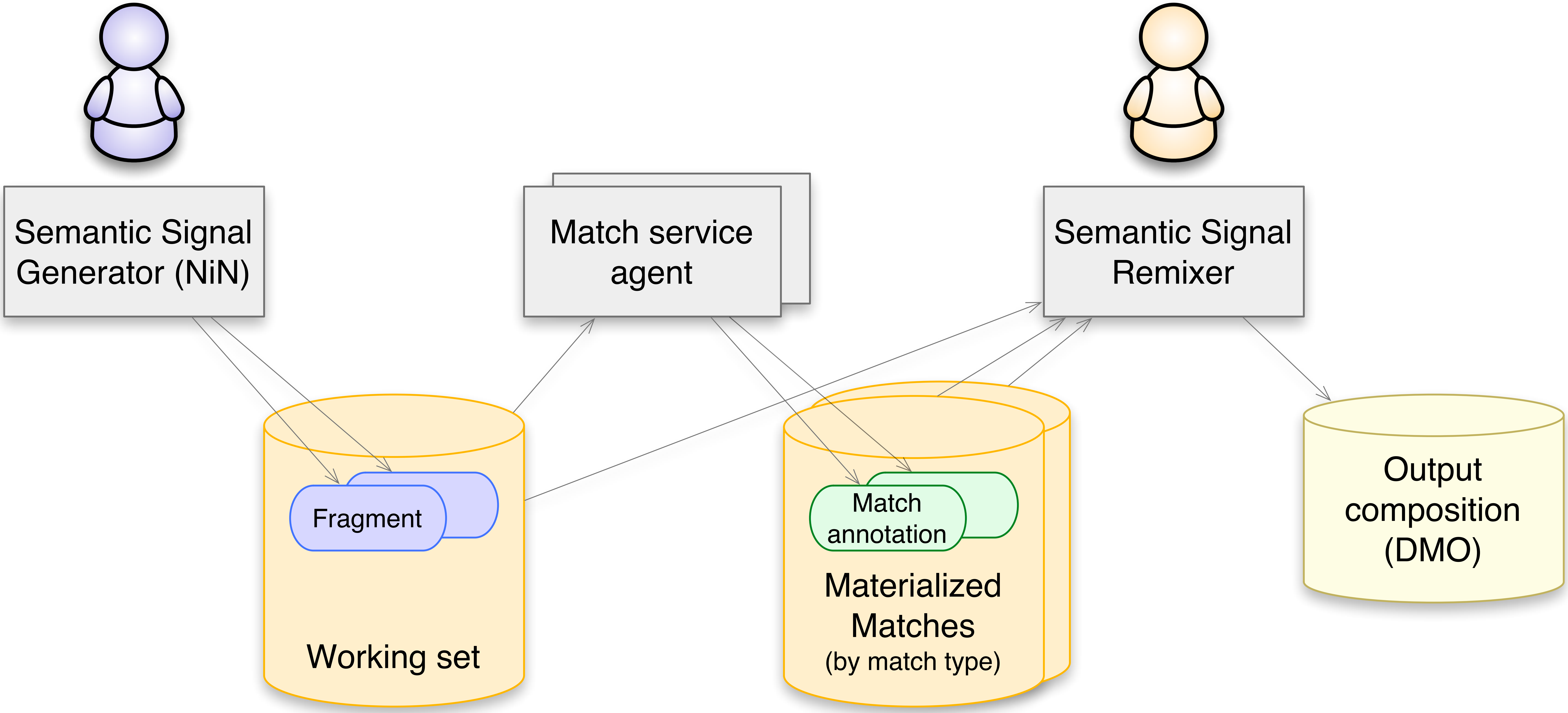
http://ldp.linkedmusic.org:8080/workset-4afa63/

Instrument 1

Instrument 2

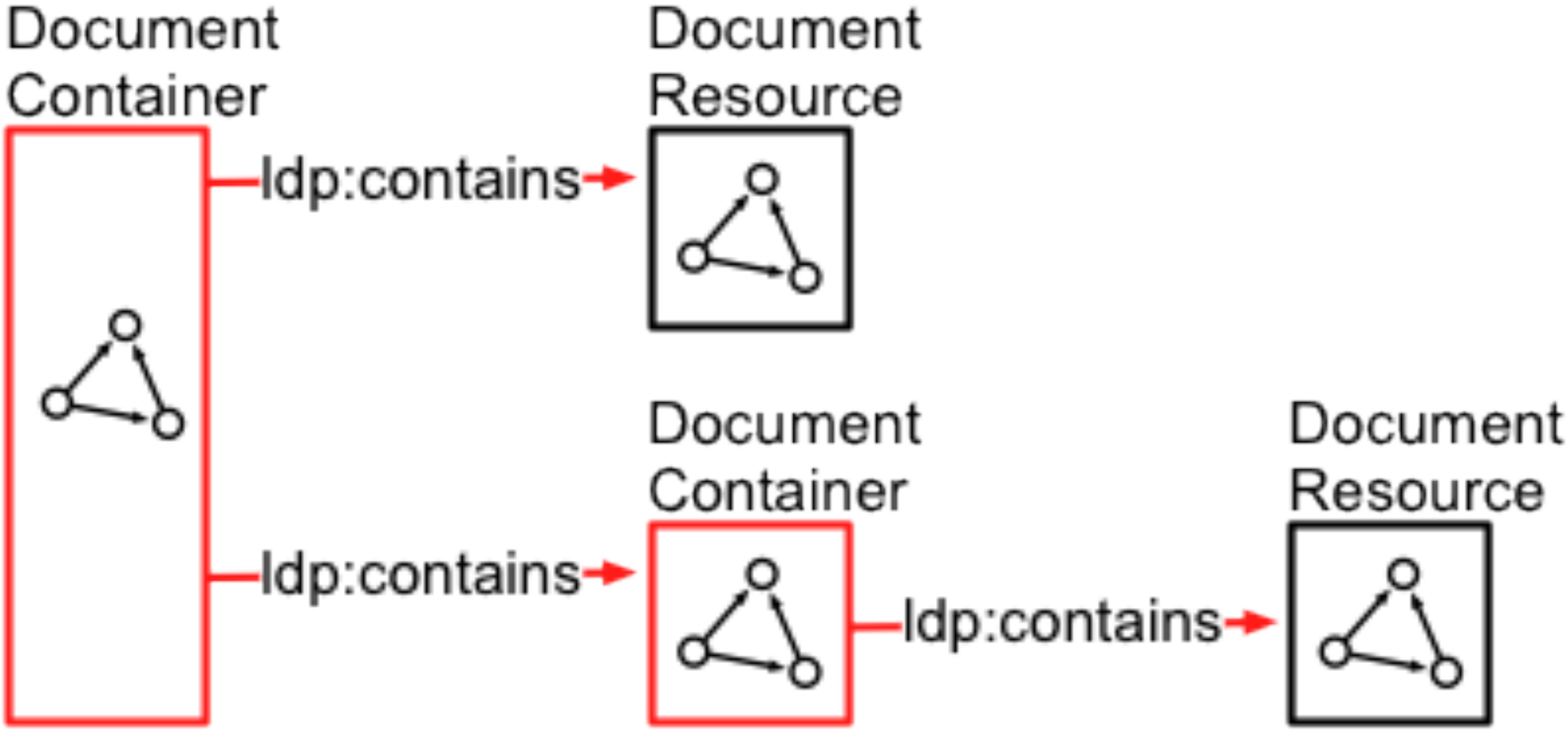
Instrument 3

How does SOFA work?



MELD elements (1)

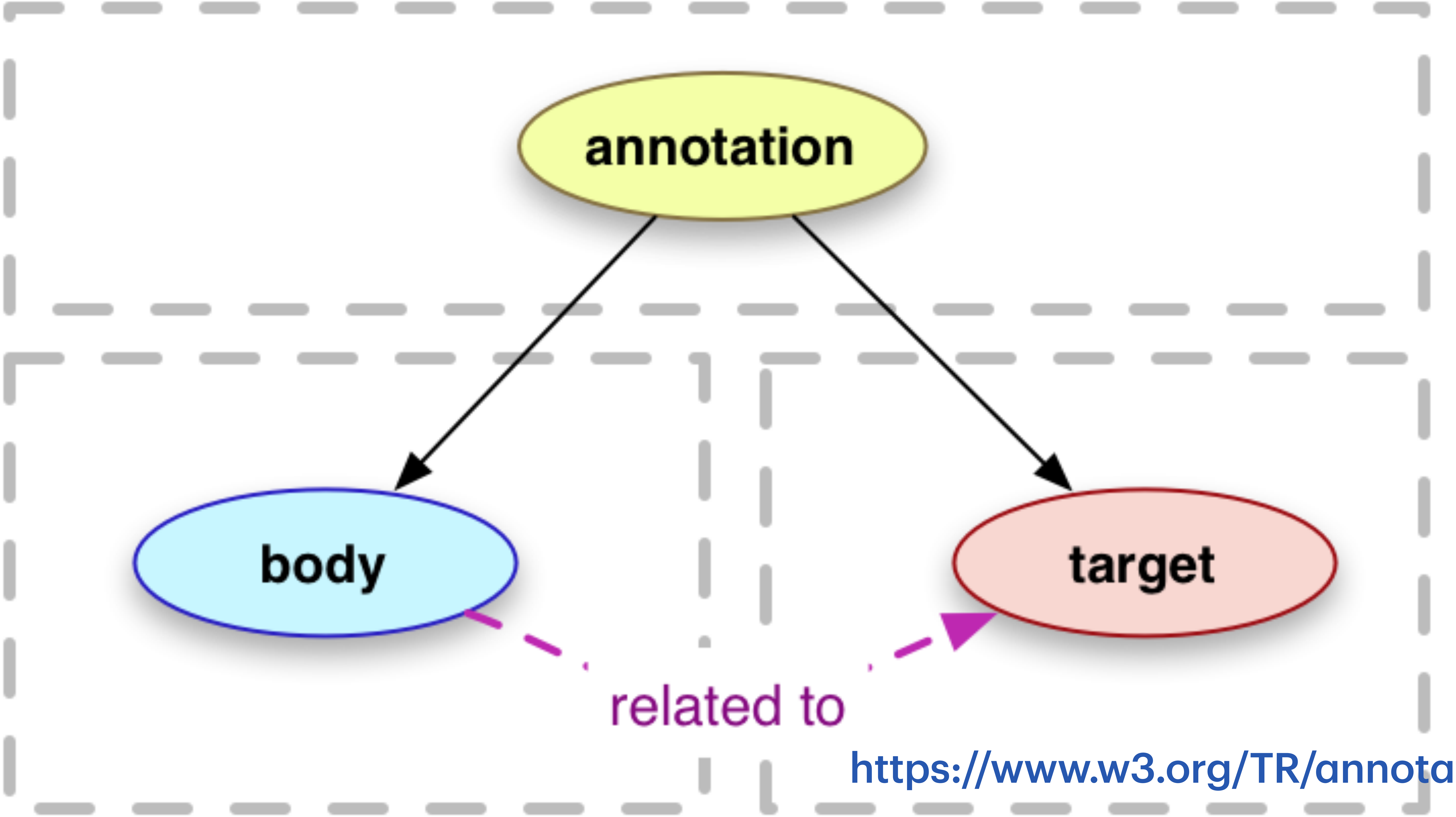
Linked Data Platform containers



https://www.w3.org/TR/ldp-primer/

MELD elements (2)

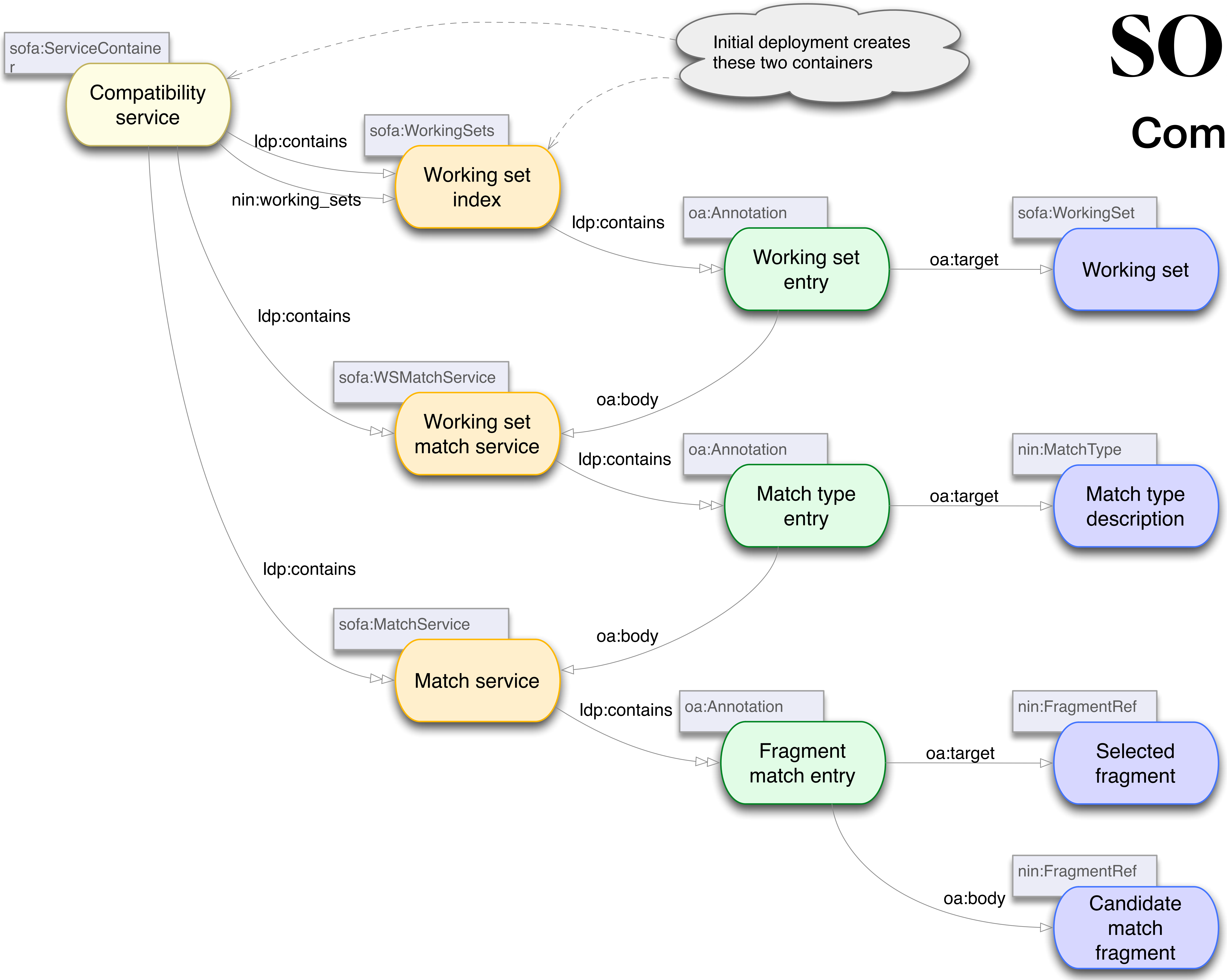
Web Annotations



https://www.w3.org/TR/annotation-model/

SOFA use of MELD

Compatibility service data model



SOFA outputs

Quilts, grids and sequences: SOFA generates grids and/or sequences

Container
Quilt
Co[*]
sequences
Sparse Grid
Grid

MELDfest day 2
04-Nov-2020

SSI3 - supporting sustainability of digital humanities research software

(An exemplar of) Best practice for testing in DH software development

SSI3 MELD activity

(An exemplar of) Best practice for testing in DH software development

- Realised as a testing framework for MELD
- Building upon command line MELD client developed in the later stages of FAST
- Extending this as a unit testing approach for MELD app development

- We have a small but varied ecosystem of music studies and their associated software the testing framework could be validated against:
    - Lohengrin opera analysis
    - Delius live performance annotation
    - Historical musicology with mixed media digital archives
    - TROMPA Rehearsal companion

SSI3 MELD activity

(An exemplar of) Best practice for testing in DH software development

- A reflective case study of software sustainability in DH projects. While the MELD framework itself benefited from the substantial research efforts of FAST, the apps above have been created within the more limited resourcing of humanities projects. This is an opportunity for SSI to reflect on effective software sustainability practice in such situations, which are typical for DH.

- Outputs
  - ○ The software testing framework for MELD (public repo).
  - ○ Unit tests for (some/all of) the above studies/apps (public repo).
  - ○ Documentation and a best practice report. (public, online).

SSI3 MELD activity

Anticipated scope of work

WHAT:

- MELD 2 API as used by MELD ecosystems applications
- Focus on HTTP interface, not intending to address UI presentation
- Validation of data collected and used by MELD applications

HOW:

- Support automated testing for catching regressions as software and/or data are updated
- Test support for profiling (not in actual applications)

SSI3 MELD activity

Anticipated approach for work

- Initial applications to study:
  - Lohengrin opera analysis
  - Delius live performance annotation

- Aim to develop tests as applications are migrated to use MELD 2 core libraries
- Possible early focus on graph traversal
- Possible construction of test suite alongside MELD application example (cf. documentation)

- Work will hopefully inform development of further sustainable MELD applications
- Role of tests as a form of documentation (sample code)

SSI3 MELD activity

Previous work and future directions

- MELD command line tools
  - https://github.com/oerc-music/meld-cli-tools
  - Javascript / Node tool for probing LDP containers
  - Some support for data structures used by MELD (annotations, etc.)
- Linux shell scripts to run tests as suite (ad hoc)
- Add test features that reflect MELD 2 core API functionality and data formats

- Think about alternative forms test runners: shell script, Javascript, or…?
- Think about tooling to capture test cases while exploring data
- Design for easy extensibility
- Possible assist with "Application setup in RDF: state, behaviour and appearance"?

SSI3 MELD activity

Previous work - MELD CLI tool - simplified testing example

```
CONTAINER_PATH=$(node meld_tool_cli.js make-container $TEST_PATH test_container)
test_sts $? "make-container" \
    && test_eq "$CONTAINER_PATH" "${TEST_PATH}test_container/" "make-container"

PUBLIC_CONTENT=$(node meld_tool_cli.js list-container $TEST_PATH)
test_sts $? "list-container" \
    && test_in "$PUBLIC_CONTENT" "$CONTAINER_PATH" "list-container"

node meld_tool_cli.js test-is-container $CONTAINER_PATH
test_sts $? "test-is-container"

CONTAINTER_CONTENT_TYPE=$(node meld_tool_cli.js content-type $CONTAINER_PATH)
test_sts $? "show-content-type" \
    && test_eq "$CONTAINTER_CONTENT_TYPE" "text/turtle"
```

SSI3 MELD activity

Testing frameworks - quick survey

SURVEYS:
https://geekflare.com/javascript-unit-testing/ - survey
https://designmodo.com/test-javascript-unit/ - another survey, more opinionated?

TEST FRAMEWORKS (pick one):
https://mochajs.org/ - flexible, backend and front; works with other test libraries
https://jestjs.io/ - "focus on simplicity" - Facebook-maintained - preferred for react

SUPPORT LIBRARIES:
https://karma-runner.github.io/5.2/index.html - flexible test runner
https://www.chaijs.com/ - useful assertion library
https://sinonjs.org/ - "Standalone test spies, stubs and mocks for JavaScript."

These slides at:
https://github.com/oerc-music/nin-remixer-public/blob/master/notes/20201004-SOFA-and-MELD.key.pdf

SOFA SAAM paper “Music SOFA: An architecture for semantically informed recomposition of Digital Music Objects” at:
https://ora.ox.ac.uk/objects/uuid:989f8931-ac42-43ed-b6cc-9d6b1386dd3c