Model Translation

And now for something completely different

translating data between models:

— (source-led) follow source structure, add semantic annotations, provide actions that build target from what occurs in the source

— (target-led) follow a generic target model, pick from source using selectors to incorporate source material into target structure

Analogies

source-led: Parser model, generally consumes all source material (unless deliberately skipped), pushes source material into the target

target-led: XSLT model, "style" model; only can use parts of the source that it knows about, pulls source material into the target (cf. YOUPI)

Other models?

Questions

Are the models incompatible, or is there just a continuum?

What is good, what is bad?

— Extensibility — what happens with unanticipated stuff in input?
— Incompleteness
— Dealing with weird sources/weird targets — can there be an abstraction layer? (multi-pass may work for that, too); rich intermediate representation
— Capture lineage, how good the translation was; provenance

DSSSL/FO, XSLT/XSL — i.e., strict target models evolve; is that unavoidable?

What are we translating?

— Translating information between models ("payload translation")
— Translating models between modeling languages

What is it about?

— data
— interactions ("protocols")
— serializations, generic models, ...

Why are we translating data?

- To "move" data from environment α (uses modeling language A) to environment β (uses modeling language B), moving from model a to model b (data cross migration)
- To "move" data from modeling language A1 to modeling language B1
- To "move" data from model a1 to model a2 (data forward migration)
- Ancillary to interaction translation
- Translate after translating models

Why are we translating models?

- Get rid of a stupid modeling language
  - Or simply expose information that is hidden by noise in that language
- Evolve a model from language version A1 to language version A2
- Make model a useful as a model in environment β

What are we translating between

- modeling language: serialization, structural, semantics
- vocabulary (e.g., IPSO units vs. SenML units), "semantics"
- structural decisions (array in α, map in β), data shape
- serialization decisions

Help in "guided model construction"

Recommender, "Tracker"
Aid developers — what has been done already; domain knowledge
(E.g., for data lifecycle analysis)
"Where's the toolkit?"

Use AI: Embody the knowledge of the participants

Role of inferencing (Ah → Wh — need V), filling in gaps

Add annotation to source model in order to facilitate upconversion

upconversions, downconversions

- Semantic levels may be incompatible
  - Add data, discard data
  - Add structure, discard structure

circuit breakers

- Find out whether a translation is "safe" or "dangerous"

What are the results we want to achieve?

Metaservice that translates, provide as a community
Assess how good those translations/translators are
"Distance" measure
Coverage
Recommender for upconversions

Enable developers to work against Semantic API
(Including Interaction Models)

Certified good reusable model components
Design patterns (and antipatterns)

Engineering principles

Success criteria

Usability

Take the developer view: Does it work?
Do we have a playbook for them?

Reducing the amount of manual work
(Doesn't have to be holy grail of fully autmated)

Helping create orgs such as schema.org

Engage domain experts

Things to look at for inspiration

Movie metadata for object ingestion

Learn as much from each translation as possible
Document those learnings

Berkeley Brick work
(And their methodology)

VSS: Automotive Infotainment?