



FLUME Metadata

Allyn Treshansky

09 May 2006

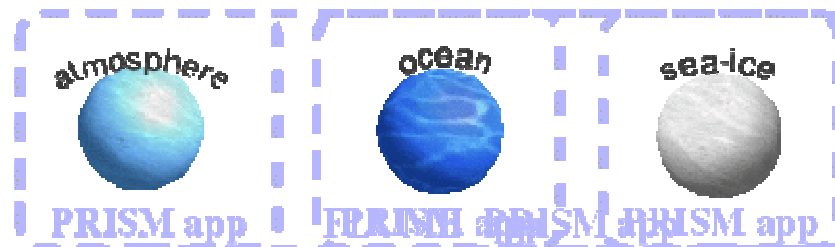
- what is FLUME (and why should you care)?
- FLUME Metadata overview
 - types of FLUME Metadata
 - manipulating FLUME Metadata (D^CCD)
 - grids
 - uninstantiated grids, instantiated grids, & grid instantiators

what is FLUME?

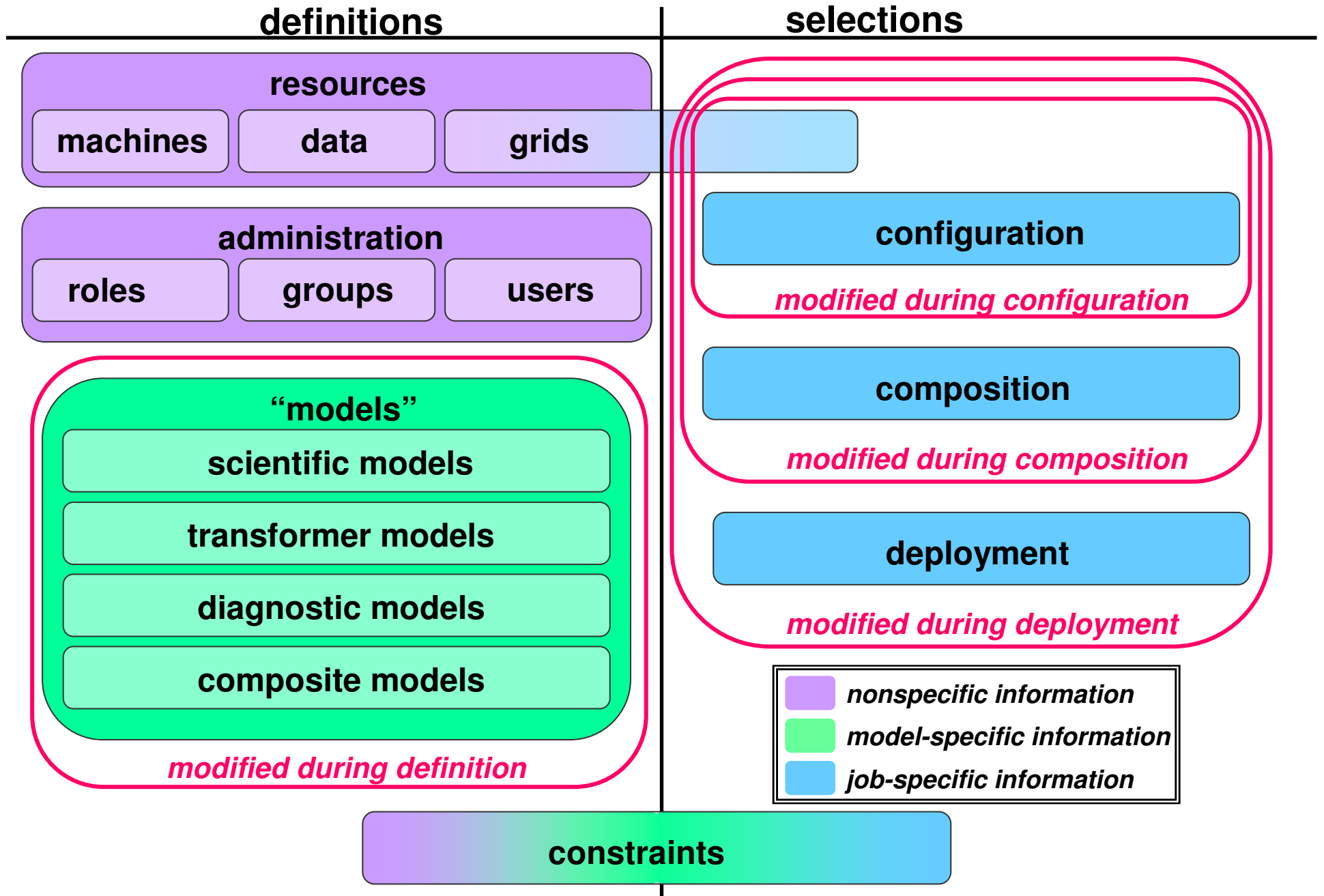
- *FL*exible *U*nified *M*odel *E*nvironment; future version of UM
- scientific code is *modularised*; infrastructure code is *generated*
- FLUME includes:
 - FLUME models (“everything is a model”)
 - minimal FLUME framework, including adapters to external technologies
 - user interface
 - code generation (via BFG, or at least something which is compatible with BFG)
 - **metadata to tie everything together**

why should you care?

- data generated by FLUME jobs get a free “trace” of the relevant job settings and job history
- FLUME users can easily share their models with other ESM groups since FLUME models are infrastructure-independent
- the opposite is also true – externally developed models should be easier to integrate into FLUME
- autogenerating infrastructure code allows the infrastructure to vary w/out requiring model code to change:



types of FLUME Metadata



manipulating FLUME Metadata

1. An administrator ensures that administration metadata exists
2. A developer ensures that resource metadata exists
3. A scientist ensures that CE metadata exists and is up to date with current FLUME model & transformer code. This is the *definition* phase of D^CCD
4. A user logs into the FLUME UI (all that is meant by “logs into” in this context is that the user is checked against the information specified in administration metadata; this will determine the set of metadata that the user can access and the set of constraints that are enforced during D^CCD)
5. The user retrieves CE metadata and/or previously saved sections of selections metadata (such as an entire composition).

manipulating FLUME Metadata, cont...

6. The user generates and/or updates selections metadata (creating configurations, compositions, deployments)
7. The user validates that selections metadata; this includes constraint checking
8. The user repeats steps 6-7 until happy. These steps form the *configuration*, *composition*, and *deployment* phases of D^CDC.
9. Once happy, the user saves the set of validated metadata as a specification of a FLUME job.
10. That specification gets passed to a code generation system which produces the actual job. The code generator reads selections metadata and returns source code and/or control files that can be parsed by source code

manipulating FLUME Metadata, cont... cont...

10. The user executes the job.
11. References to the job specification (or a subset of the actual job metadata) can be incorporated into the job output
12. A developer and/or scientist reviews and, if appropriate, adds further technical and/or scientific information (such as constraints) to store with the job specification.

FLUME grids include:

- uninstantiated grids
 - a high-level generic schema that contains standard names, question / answer pairs, and a reference to a grid instantiator – including the specific member grids that it can produce (even though specific details about those grids cannot be known prior to instantiating the grid)
- grid instantiators
 - an algorithm written for a particular instance of a grid; takes the pre-instantiated grid answers as input and returns (a set of) instantiated grid(s) as output
- instantiated grids
 - a “complete” description of a specific grid capable of being incorporated into job code (for FLUME, PRISM, ESMF, etc.)

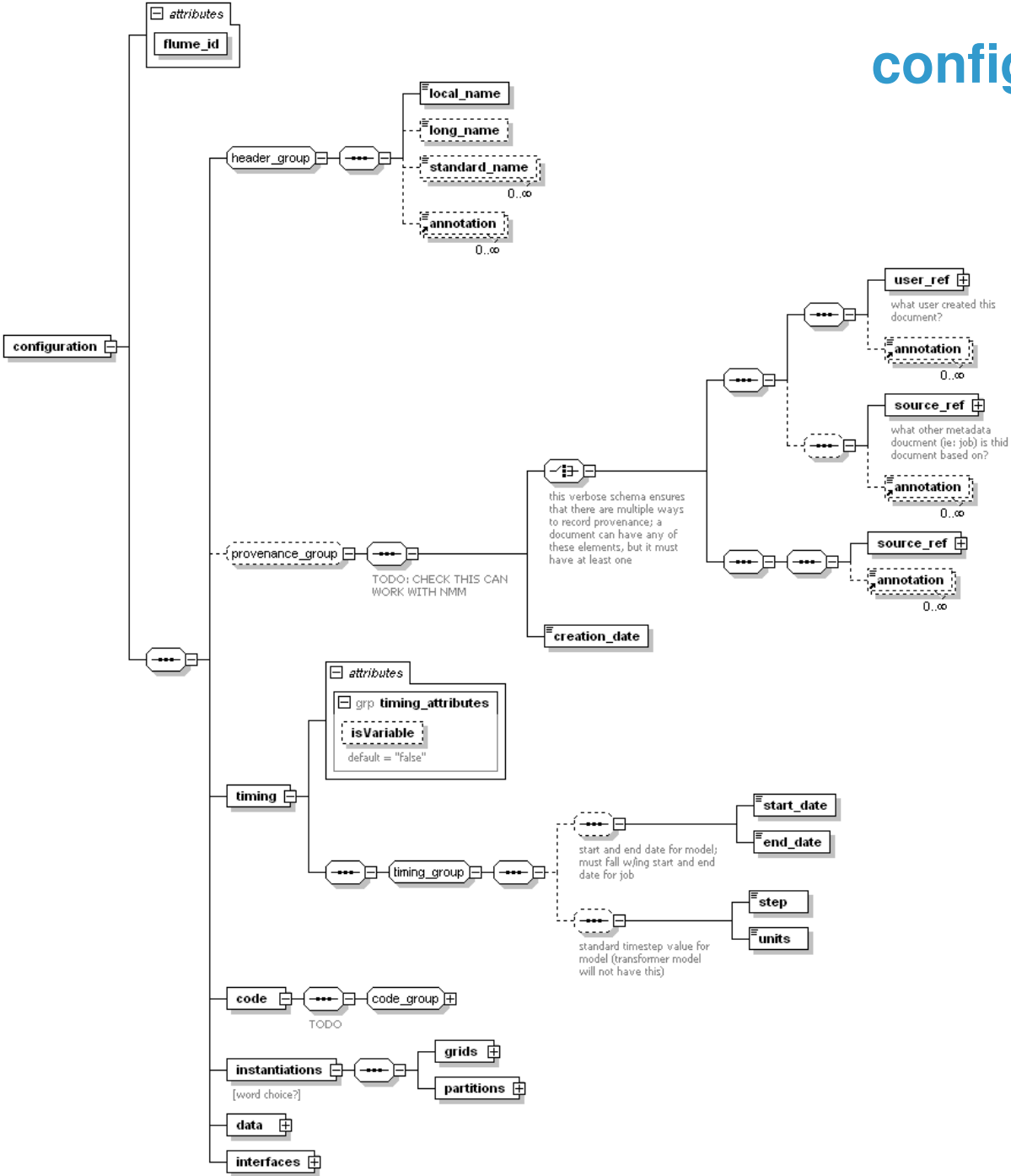
FLUME grids do not include:

- a single grid schema to cover all contingencies
 - nobody should maintain some big unwieldy XML that can cover all details of any grid; instead lots of people should create lots of small algorithms to produce the grids they care about
 - FLUME's abstract uninstantiated grid is isolated from as yet unconsidered grids; if additional grid features are used by model, then the set of question/answer pairs is simply extended and a new algorithm may be produced – no schema needs to change.

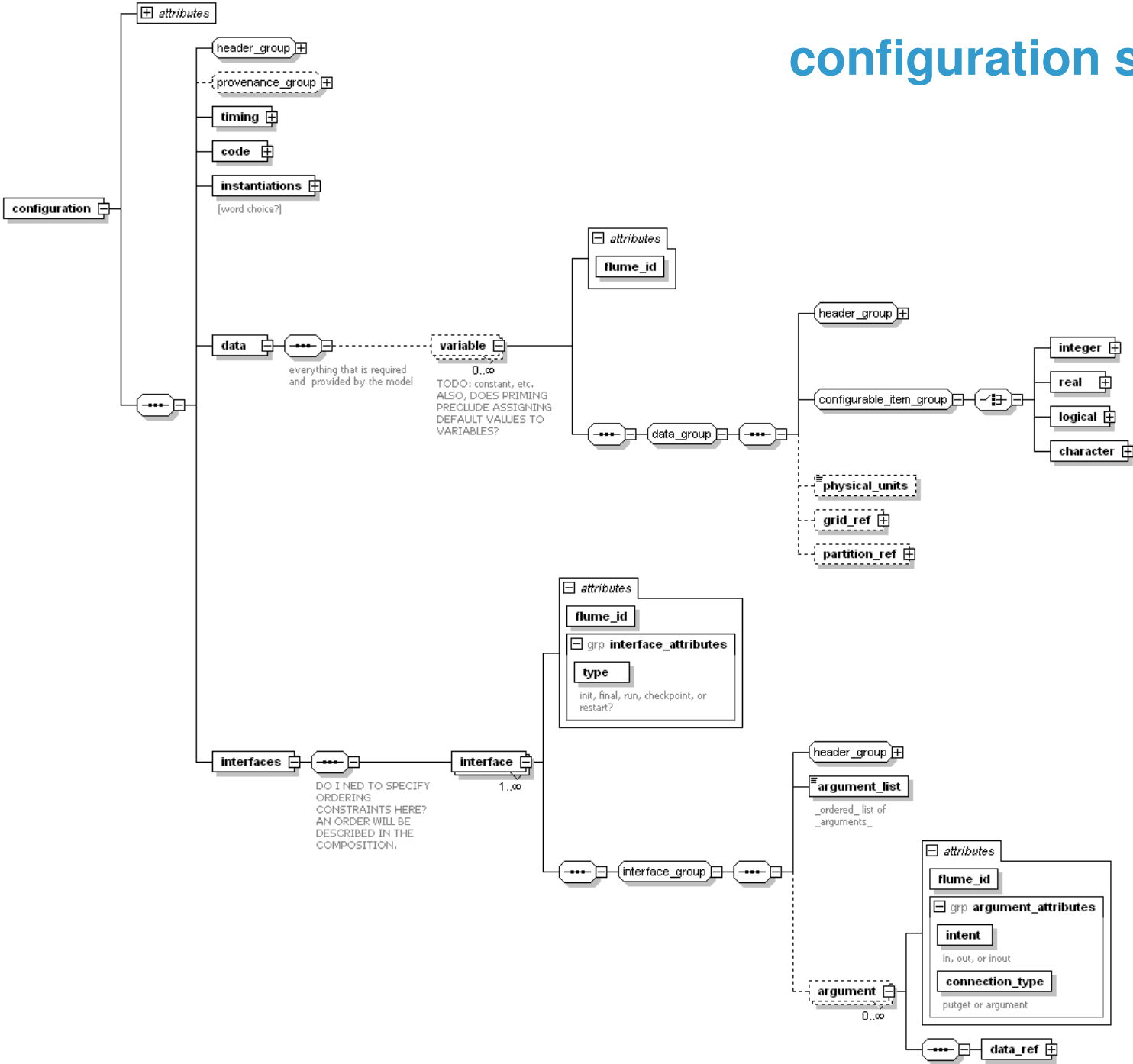
what next?

- agree structure of instantiated grids
- agree importance of distinction between uninstantiated & instantiated grids

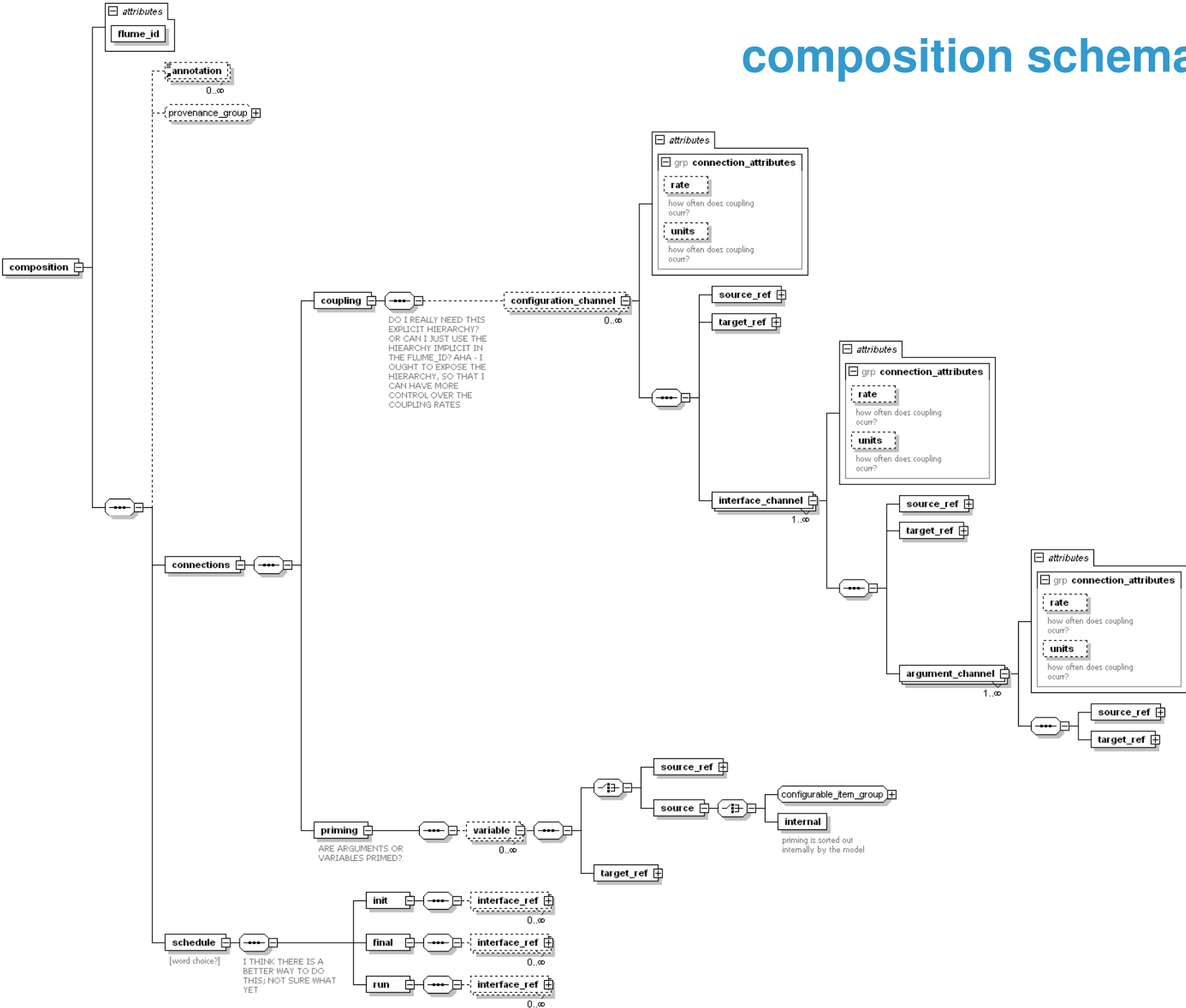
configuration schema (I)



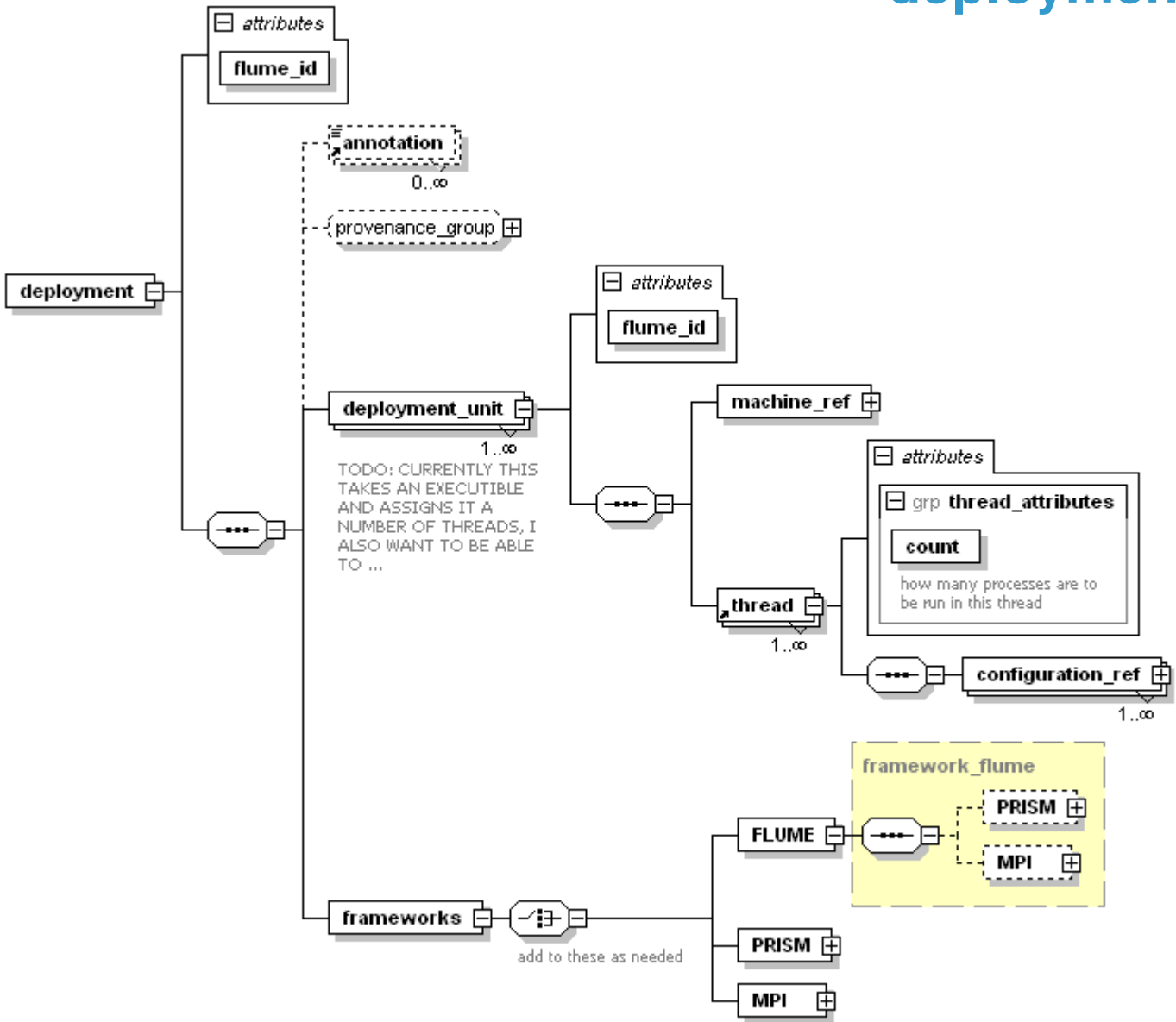
configuration schema (II)



composition schema



deployment schema



uninstantiated grid schema

