Code Generation with XSD Restrictions & Common Types
• No duplication of common objects (although no message-specific CT variation).
• Higher rate of re-use and increased consistency (getting used to common data structures).
• Simplified data mapping on persistence layer.
Each schema allows for message-specific variations of common types, as derived and restricted from AIDM Logical Model.

Different namespaces per message (to avoid element name collisions) results in one package per message.

Repetition of many of the same objects across packages.
JAXB Code Gen with XSD CTs & Restrictions

Homogenous & Chameleon Namespaces + CTs:

- Message 1
  - CT 1

- Message 2
  - CT 3

- Message 3
  - CT 2

Common Types:
- CT 1
- CT 2
- CT 3

Proposed Solution:
- Re-use of XSD common types, *and* ability to specify variations at message level with XSD restrictions and extensions.

Single Package:
- Message Class 1
- Message Class 2
- Message Class 3

Data Binding to DB:
- CT1 Class
- CT2 Class
- CT3 Class

CT1 Class
- Extends CT1--

CT2 Class
- Extends CT2++

CT3 Class
- Extends CT3--
Next Steps

- Assess ability to generate XSD restrictions from SparxEA
  - May require additional “interim staging area” specific to each standards project.
  - Changes to transformation scripts needed.

- Investigate interim solutions (e.g. repackaging code-generation-friendly release – XML validation would remain the same but internal organization of XSD would implement CTs).

- Test code generation on other platforms (JS, PHP, C# / VB.Net…)

- Consider converging “variant” types to consistent common types wherever possible, reducing footprint of extended classes