Building Blocks

A paradigm for product format design

Interlude edition

Scott Houchin
Exploitation Phenomenology and Analysis Department

4 March 2020

Approved for public release. OTR 2020-00361

Many slides are animated—view in slideshow mode
Narrative for mostly pictorial slides is in the notes


LEGO
LEGO
LEGO
LEGO
LEGO
LEGO
LEGO
LEGO

Building Block paradigm for product format design

Beyond microformats to combinations! Reusable profiles!

- Product formats defined in terms of sensor/system agnostic, well-defined, simple and modular building blocks
- Forces data modelling; product isn't just a dumping ground for whatever data we decide to put into it
  - *Forces product format designers to think beyond the immediate need*
  - *Data consumer can transparently handle multiple products/systems*

Part dimensions	Part 1	Part 2
Connector spacing	8	8
Connector diameter	5	5
Connector height	1.7	1.7
Width	24	16
Length	56	32
Height	9.6	9.6

Agnostic, unambiguous, reusable

Doesn't everyone develop their products this way?!

Yes, when the consumer is in charge

- It's easy to think things are just done this way because we have so many examples of ubiquitous use of standards in everyday life

DRIVESAFER™

- Would you switch to Verizon FiOS if their set-top boxes required that you bought all new TVs?
- Would you buy a Tesla Automobile if the turn signal was triggered by a foot pedal?

We take reusability and standards for granted, because they're pervasive in real life

Doesn't everyone develop their products this way?!

Not really that often, if you think about it

- Did you even consider what type of oil filter was used by the last car you bought?

- Fram makes

425

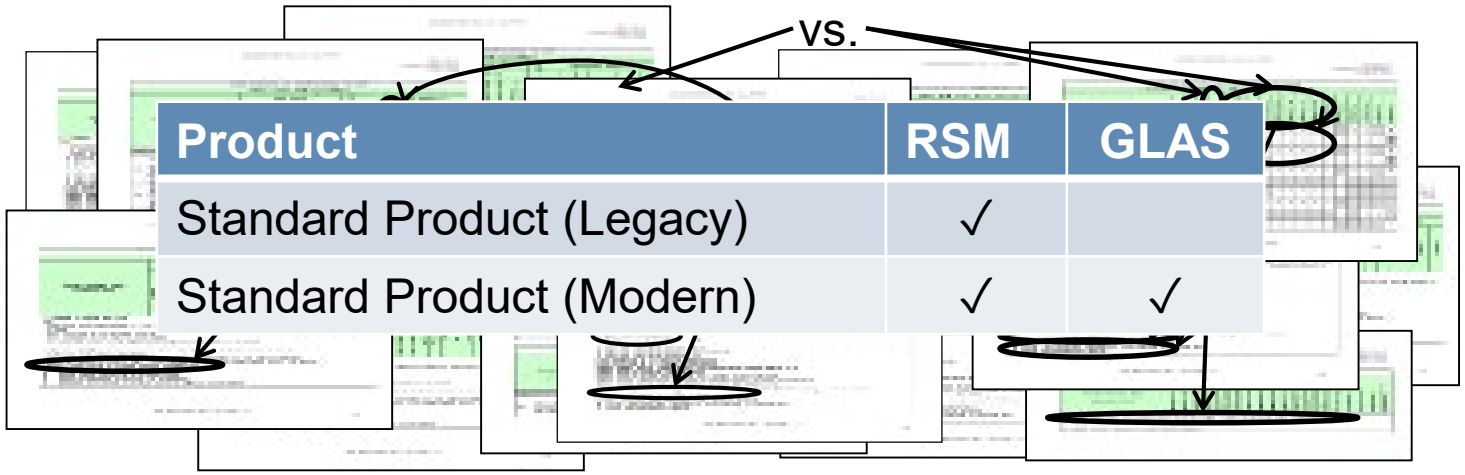
different oil filters
(at just one of four different quality levels) because auto makers have not standardized the interfaces to the engine

If the producer is in control, there is little incentive to use standards

Stovepipe focus prevents reusability, obfuscates info

Extreme focus on "what" minimizes "how" and "why"

- Example: What geolocation methods are supported for the Standard Product?
  - ***Without*** Building Block Answering high level questions of the geolocation approach TREs and DESs. Then look through tables and cross reference the relevant columns and notes



Product	RSM	GLAS
Standard Product (Legacy)	✓	
Standard Product (Modern)	✓	✓

Easy questions → hard answers; hard questions → impossible answers

Building Blocks

Well-defined elements with normative, unambiguous definitions

- A Building Block is not just a thought concept!
- There are correct & incorrect ways
  - *To use the blocks*
  - *To make the blocks*

- Data Consumer **must** be able to use a Building Block in one product using the **exact same rules and methods** as it uses for the same Building Block in another product!

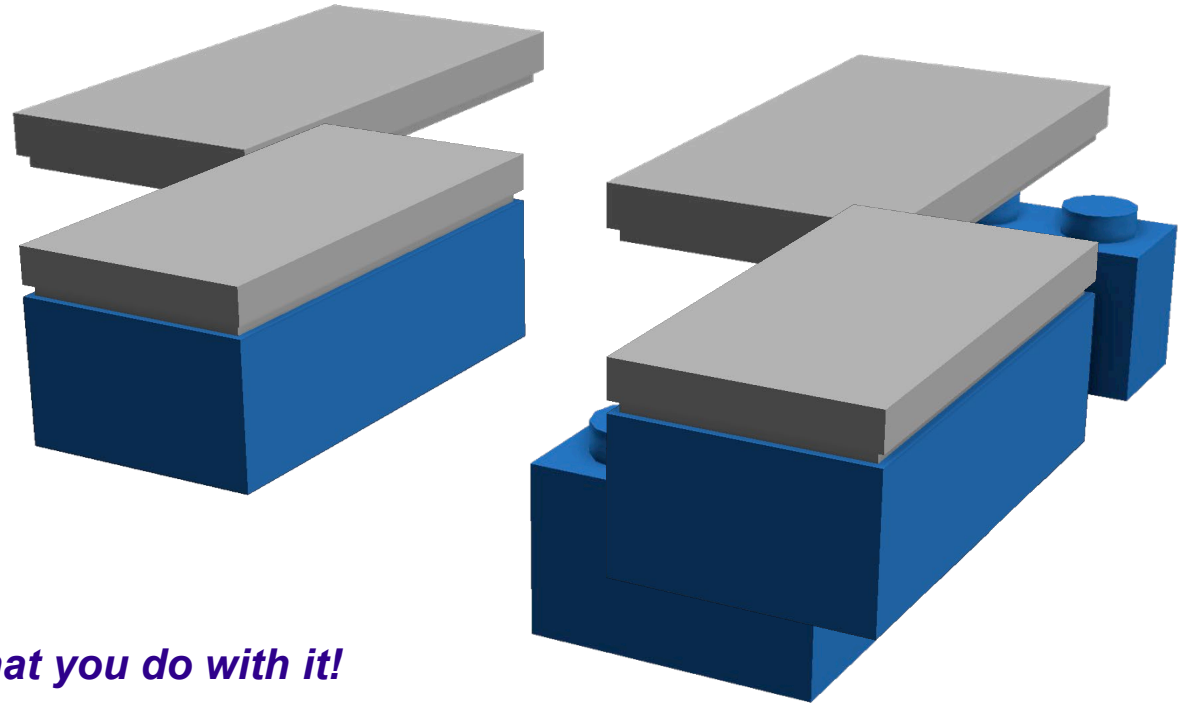
- If I specify LEGO, I don't mean the cheap imitations that don't always work exactly as claimed

Building Blocks are normative and unambiguous

Building Blocks

Specified from the consumer's perspective

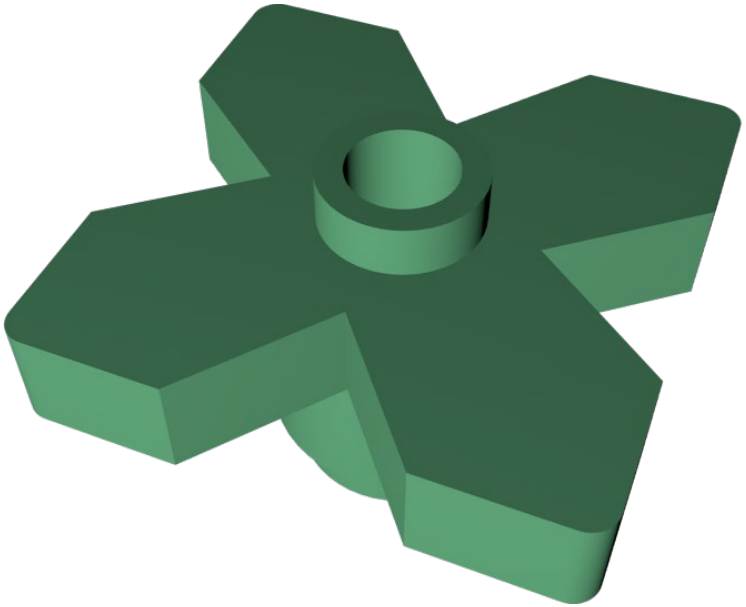
- How the block is to be used or exploited is ***many times more important*** than how it is created
- If the definition is sufficiently precise
  - *It should be obvious how it **can** be created*
  - *And it should be obvious where the producer has flexibility in creation*



Focus on what you do with it!

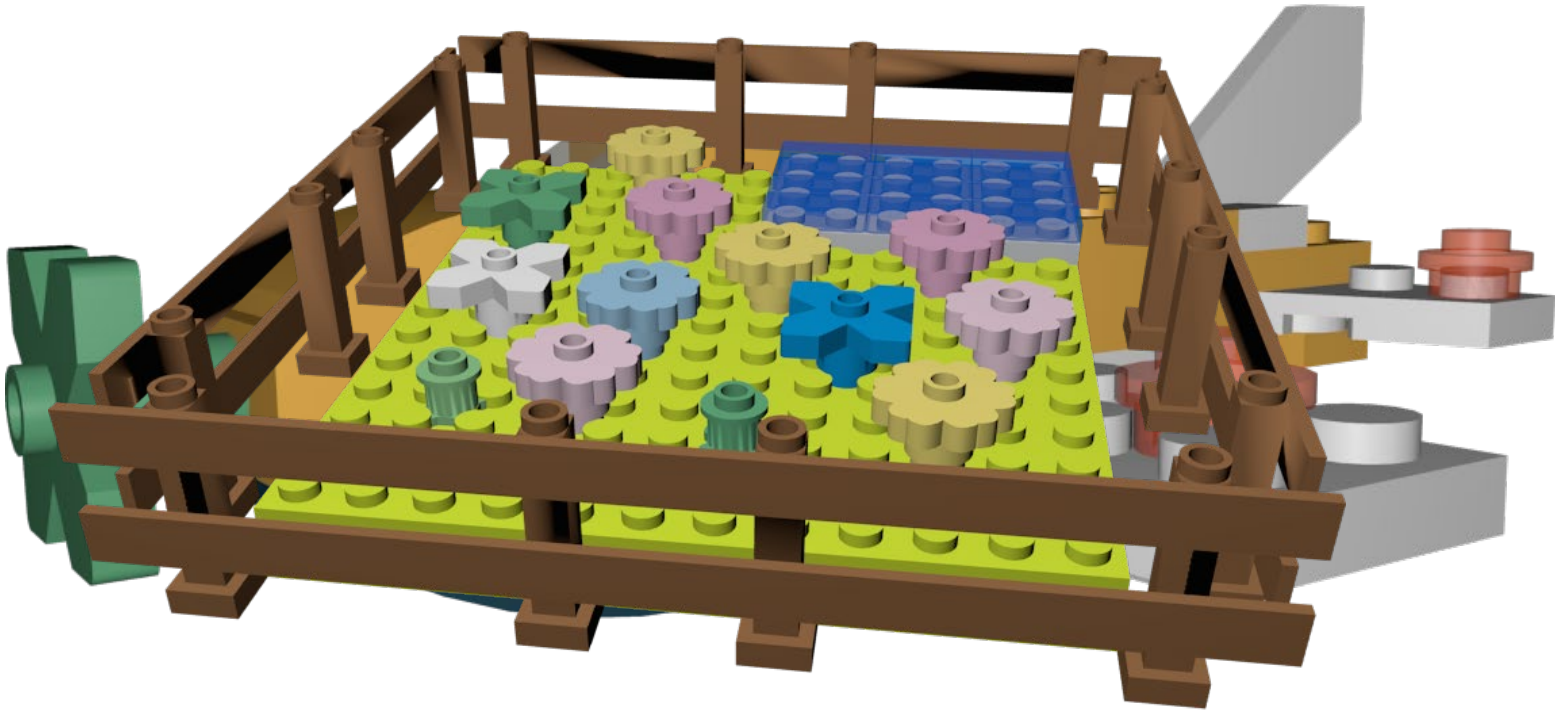
Building Blocks

Separate syntax from semantics



Building Blocks

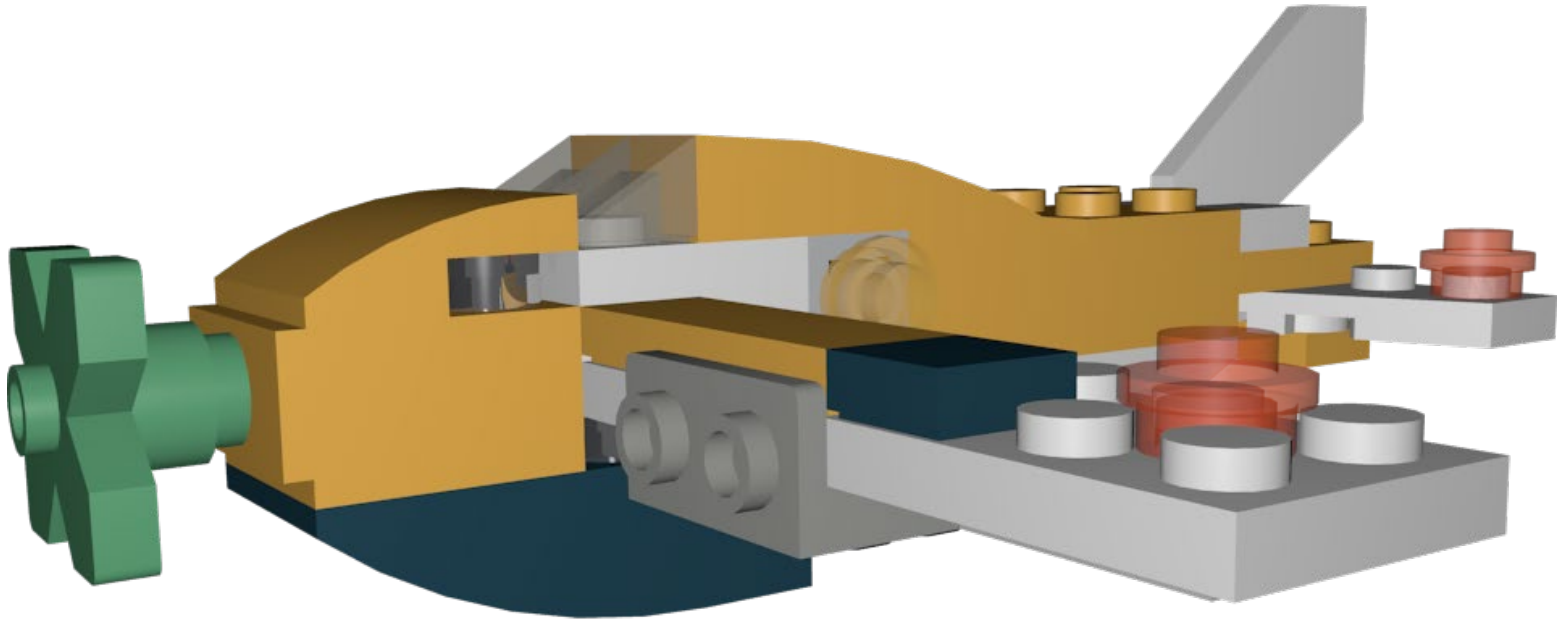
Separate syntax from semantics



What something is doesn't always define how I can use it!

Building Blocks

Separate syntax from semantics

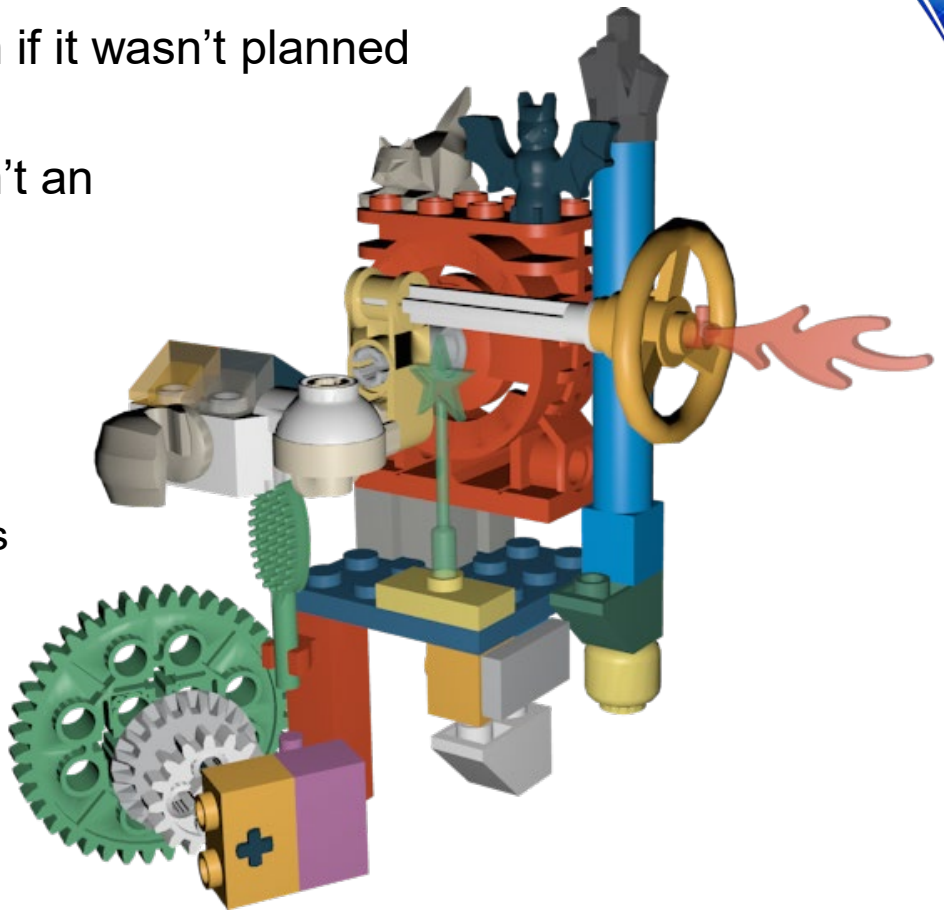


What something is doesn't always define how I can use it!

Don't panic!

Many existing products are already based on Building Blocks

- Building blocks are all around us, even if it wasn't planned
  - *So we didn't document them that way*
- Refactoring product documentation isn't an all-or-nothing proposition
  - *Be content if we have a lot of ugly, single-use Building Blocks*
  - *Define Building Blocks as time allows*
    - We don't have to do it all at once!
    - Best case, Building Block descriptions are sufficient enough to start deleting existing text
    - Worst case, new diagrams improve clarity of existing specifications

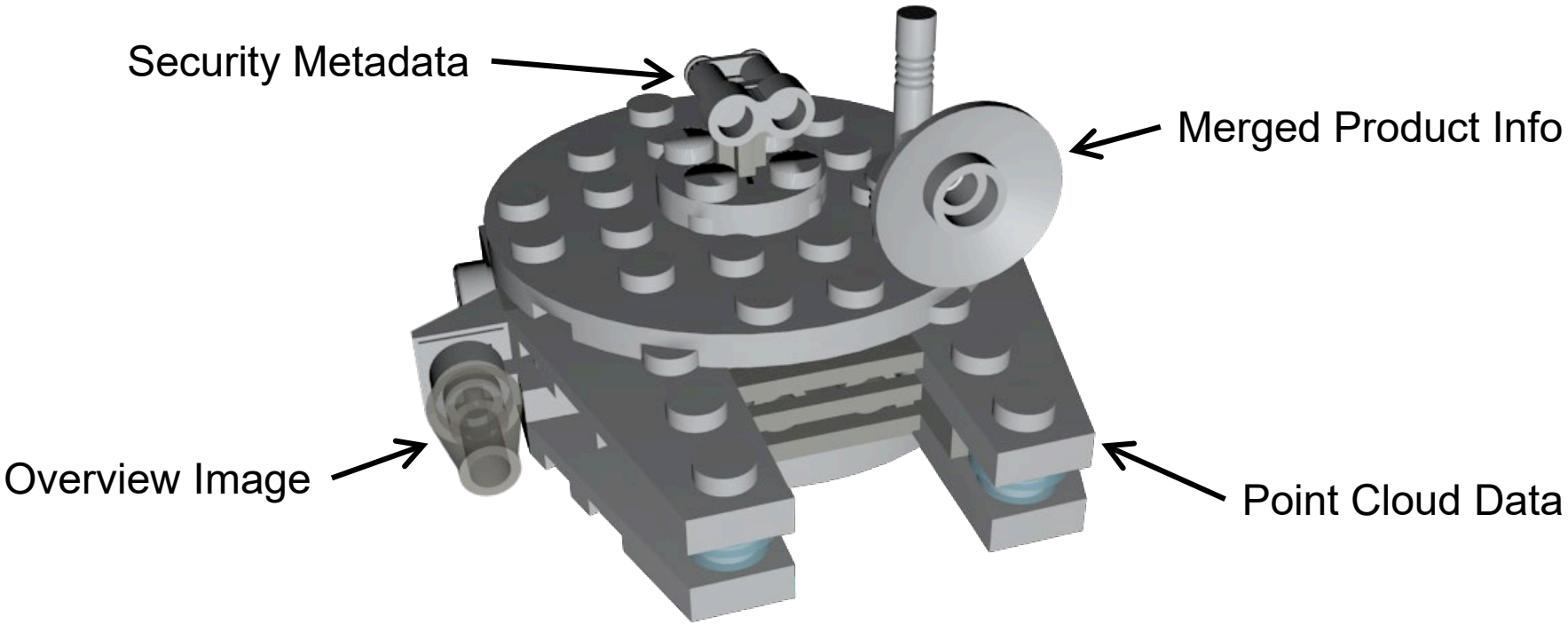


Implementing Building Block ≠ Changing the existing products

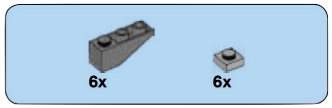
Example Building Block-based product format

Generic EO Point Cloud (EPC) Product Format for delivery to the library

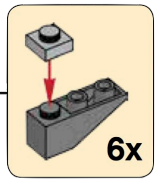
- NITF file with embedded point cloud data in externally defined formats (e.g., Binary Point File, Sensor-Independent Point Cloud), security and merged product metadata, and an optional 2D visualization to aid search and discovery



Building Block definitions can be as normative as necessary



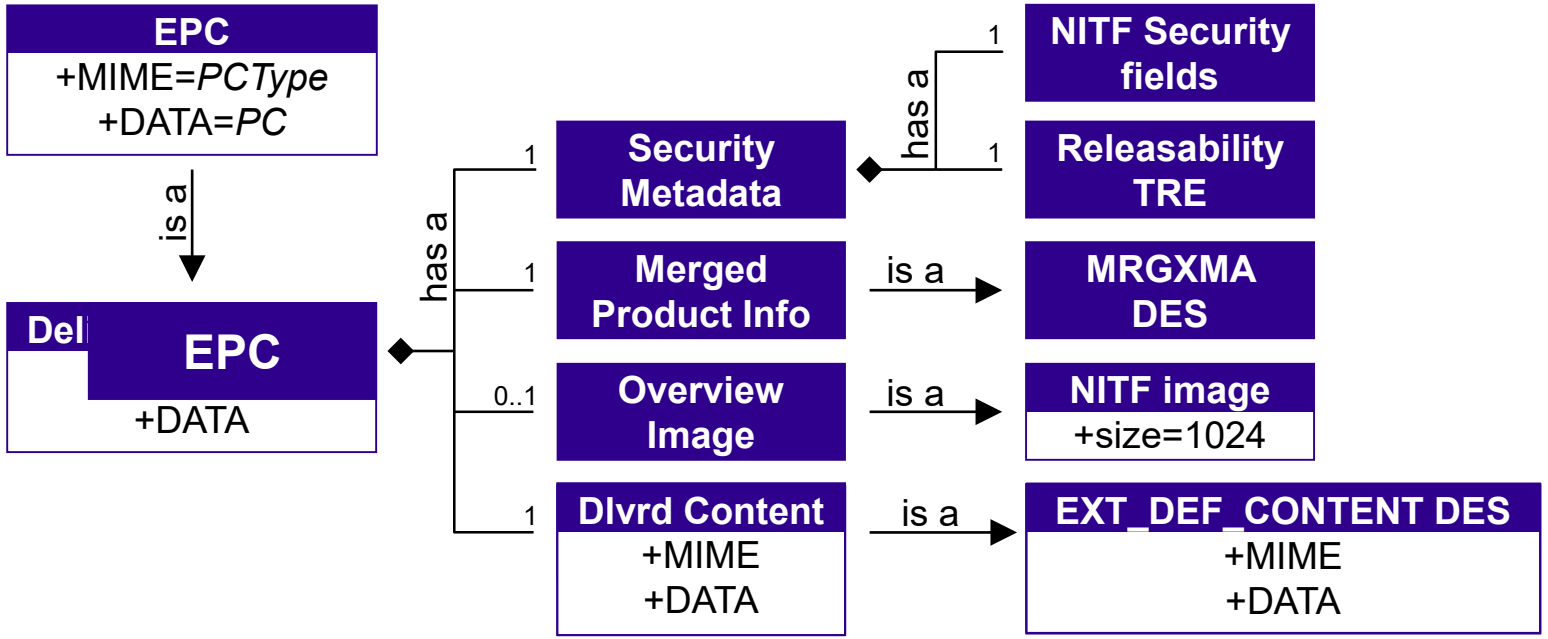
68



Example Building Block-based product format

Generic EO Point Cloud (EPC) Product Format for delivery to the library

- NITF file with embedded point cloud data in externally defined formats(e.g., Binary Point File, Sensor-Independent Point Cloud), security and merged product metadata, and an optional 2D visualization to aid search and discovery



Scalable, flexible definitions that provide just the right amount of information at once

Enables just-in-time format design & development

High reusability, information communicated clearly

- Product format designers can focus initially on the high level contents of the product, in human-centric words
  - *e.g., Generic change detection product format defines a mechanism to store a 2D georeferenced grid of values covering the compared region. Values specify whether change was or wasn't detected at that point, and if so, specify the type of change (e.g., the location was hotter)*

- Completely specifies product at the level necessary to satisfy most users of the document! Including management!
- The hard work in product development can begin unimpeded to develop the algorithms (e.g., actually build the point cloud or perform change detection)
- The structure of the actual bits on disk can come later if necessary
  - *Final formatting of data is smallest and simplest subtask*
- Building blocks can be readily reused across multiple programs because we kept the program specifics out of them

Reusable, interoperable, scalable, understandable


Questions?