NIEMOPEN

NIEM 6.0 Technical Architecture: What's New?

8 December 2022

DRAFT

OUTLINE: NIEM 6.0 IN 2023

- An OASIS Open Project
- New modeling formalism
- Convertible message serializations
- Support for ontology and knowledge graphs
- New message specification (IEPDs)
- Benefits for message implementers

NIEMOPEN

NIEM OPEN: AN OASIS OPEN PROJECT

- **NIEM is transitioning to OASIS in 2022 as an Open Project**
  - Community development of standards and software
  - Through a defined and managed process that is open, balanced, consensus-based
  - Producing new OASIS standards (like SAML, STIX, etc.)
  - With a path to international recognition (ISO)
- **The NTAC anticipates two OASIS standards for NIEM 6**
  - NIEM model (core and domains) – a shared semantic reference model for the NIEM community
  - NIEM technical framework – usable by anyone to specify data content and meaning

NIEM COMMON MODEL FORMAT (CMF)

- **NIEM has always used XML Schema as its data modeling language**
  - Model semantics formally defined via mappings to Resource Description Framework (RDF)
  - Mappings defined in the Naming and Design Rules
  - Convenient for designers and developers implementing XML-based data exchange

```
<xs:complexType name="VehicleType">
 <xs:annotation>
  <xs:documentation>
    A data type for a conveyance designed
    to carry an operator, passengers and/or
    cargo, over land.
  </xs:documentation>
 </xs:annotation>
<xs:complexContent>
 <xs:extension base="nc:ConveyanceType">
  <xs:sequence>
```

```
nc:VehicleType a rdf:Class ;
    rdfs:subclassOf nc:ConveyanceType ;
    rdfs:comment "A data type for a
      conveyance designed to carry an
      operator, passengers and/or
      cargo, over land." .
```

XML Schema like this ——— *entails* ———→ RDF like this

NIEMOPEN

NIEM COMMON MODEL FORMAT (CMF)

- **NIEM has always used XML Schema as its data modeling language**
  - Model semantics formally defined via mappings to Resource Description Framework (RDF)
  - Mappings defined in the Naming and Design Rules
  - Convenient for designers and developers implementing XML-based data exchange
- **NIEM now supports developers who aren't using XML**
  - XML Schema is not convenient for them
- **For NIEM 6, the NTAC created the NIEM metamodel and Common Model Format**
  - Metamodel: A conceptual data model for the things we want to know about data models
  - CMF: A NIEM-based implementation of the metamodel… Just like any IEPD

NIEM COMMON MODEL FORMAT (CMF)

- **NIEM 6 will have two data modeling languages: XSD and CMF**
  - The NTAC is providing free and open-source tools to convert between them

```
<xs:complexType name="VehicleType">
 <xs:annotation>
  <xs:documentation>
    A data type for a conveyance designed
    to carry an operator, passengers and/or
    cargo, over land.
  </xs:documentation>
 </xs:annotation>
<xs:complexContent>
 <xs:extension base="nc:ConveyanceType">
  <xs:sequence>
   <xs:element ref="VehicleDoorQuantity"
   <xs:element ref="VehicleIdentification"
```

```
<cmf:Class>
 <cmf:Name>VehicleType</cmf:Name>
 <cmf:Namespace s:ref="nc"/>
 <cmf:DescriptionText>
  A data type for a conveyance designed
  to carry an operator, passengers and/or
  cargo, over land.
 </cmf:DescriptionText>
 <cmf:ExtensionOfClass s:ref="nc:ConveyanceT
 <cmf:HasProperty>
  <cmf:Property s:ref="nc_VehicleDoorQuantit
  <cmf:MinOccursQuantity>1</cmf:MinOccursQua
  <cmf:MaxOccursQuantity>unbounded</cmf:MaxO
```

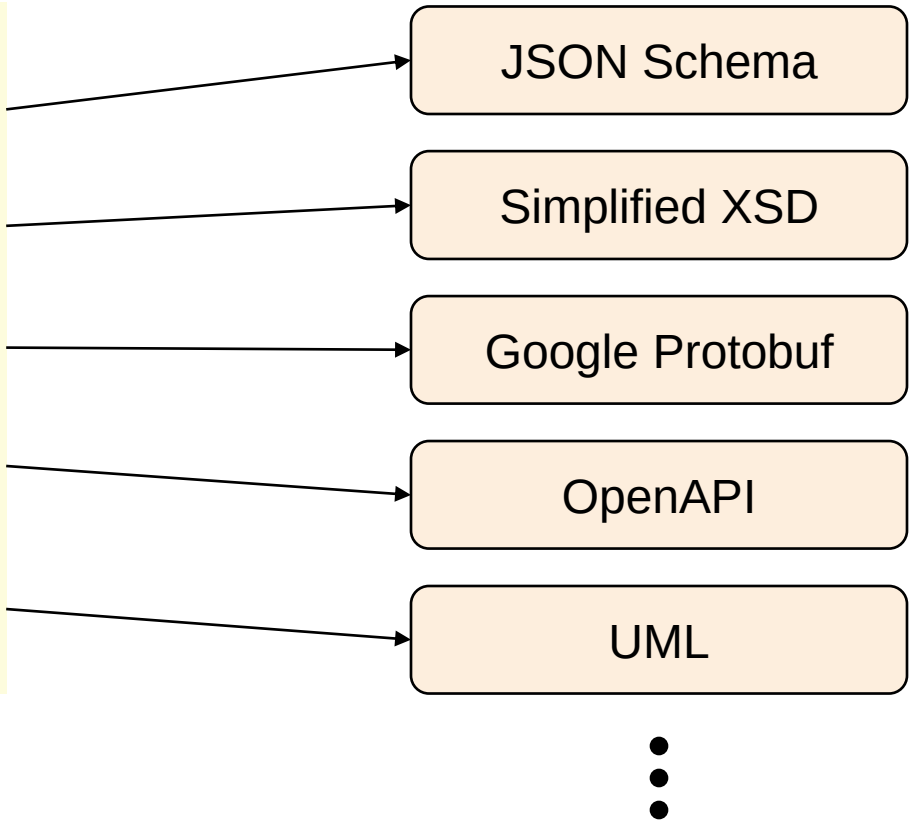
XML Schema like this ⟵ *exactly equals* ⟶ CMF like this

NIEMOPEN

TECHNOLOGY-INDEPENDENT DATA MODELS

- **CMF can be converted into developer artifacts for many technologies**
  - The NTAC is providing free and open-source tools to generate these artifacts from CMF

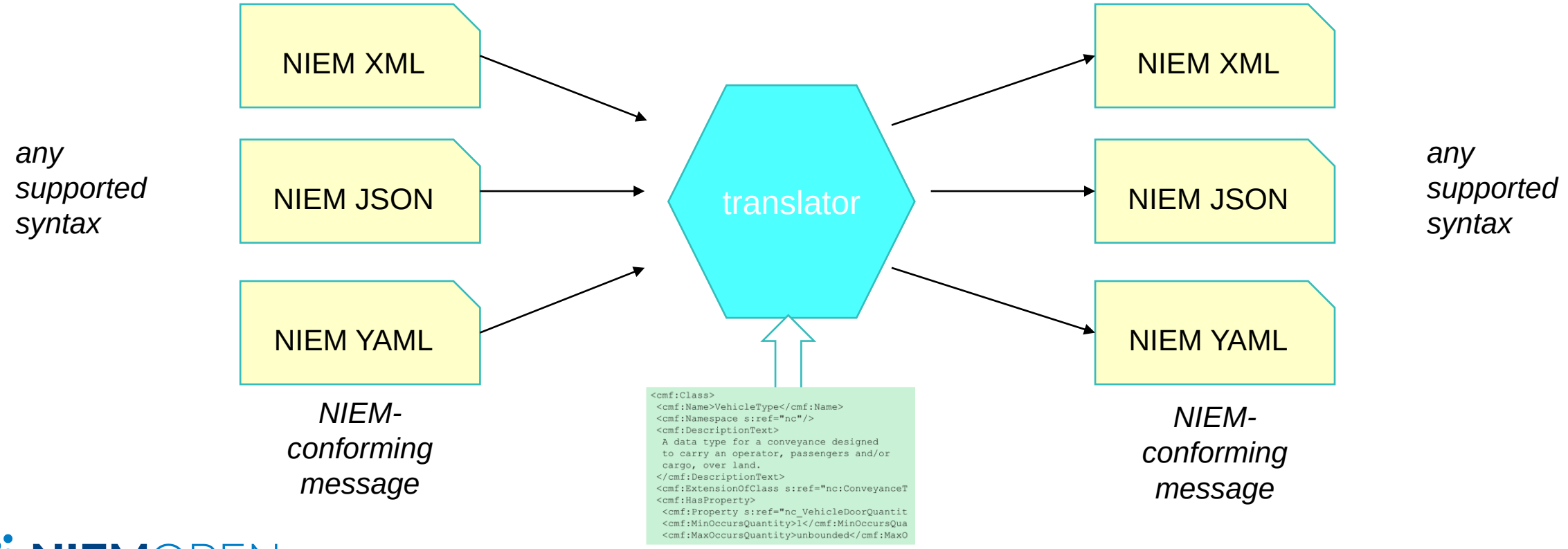
```
<cmf:Class>
 <cmf:Name>VehicleType</cmf:Name>
 <cmf:Namespace s:ref="nc"/>
 <cmf:DescriptionText>
  A data type for a conveyance designed
  to carry an operator, passengers and/or
  cargo, over land.
 </cmf:DescriptionText>
 <cmf:ExtensionOfClass s:ref="nc:ConveyanceT
 <cmf:HasProperty>
  <cmf:Property s:ref="nc_VehicleDoorQuantit
  <cmf:MinOccursQuantity>1</cmf:MinOccursQua
  <cmf:MaxOccursQuantity>unbounded</cmf:MaxO
```



NIEMOPEN

CONVERTIBLE MESSAGE SERIALIZATIONS

- **In NIEM 6, the same message information can have different message serializations**
  - The CMF model has everything needed to drive syntax conversions (XML→JSON, JSON→XML, etc.)
  - The NTAC is providing free and open-source tools to convert NIEM messages



NIEMOPEN

KNOWLEDGE GRAPH SUPPORT

- **CMF message models and runtime NIEM messages can both be converted into RDF**
  - The NTAC is providing free and open-source tools for these conversions

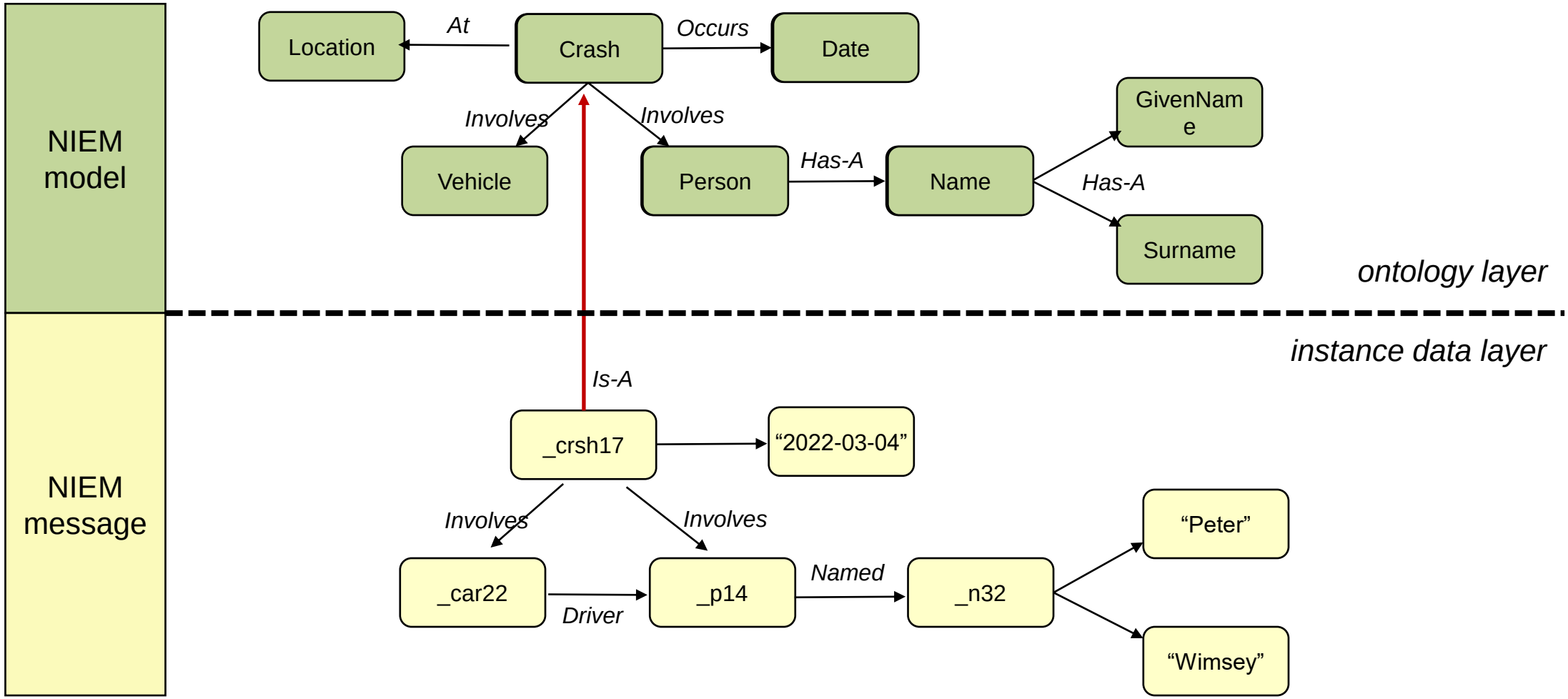
```
<ex:CrashDriverInfo>
 <j:Crash>
  <j:CrashVehicle>
   <j:CrashDriver>
    <nc:RoleOfPerson s:id="P1">
     <nc:PersonBirthDate>
      <nc:Date>1890-05-04</nc:Date>
     </nc:PersonBirthDate>
     <nc:PersonName>
      <nc:PersonGivenName>Peter</nc:PersonGiv
      <nc:PersonMiddleName>Death</nc:PersonMi
```

```
_:n0 a j:CrashType ;
     j:CrashVehicle _:n1 .
_:n1 a j:CrashVehicleType ;
     j:CrashDriver _:n2 .
_:n2 a j:CrashDriverType ;
     nc:RoleOfPerson _:P1 ;
_:P1 a nc:PersonType ;
     nc:PersonBirthDate _:n3 ;
     nc:PersonName _:n4 .
_:n3 a nc:DateType ;
     nc:Date "1890-05-04" .
```

NIEM-based message in XML ⟵ *equivalent to* ⟶ NIEM-based message in RDF

NIEMOPEN

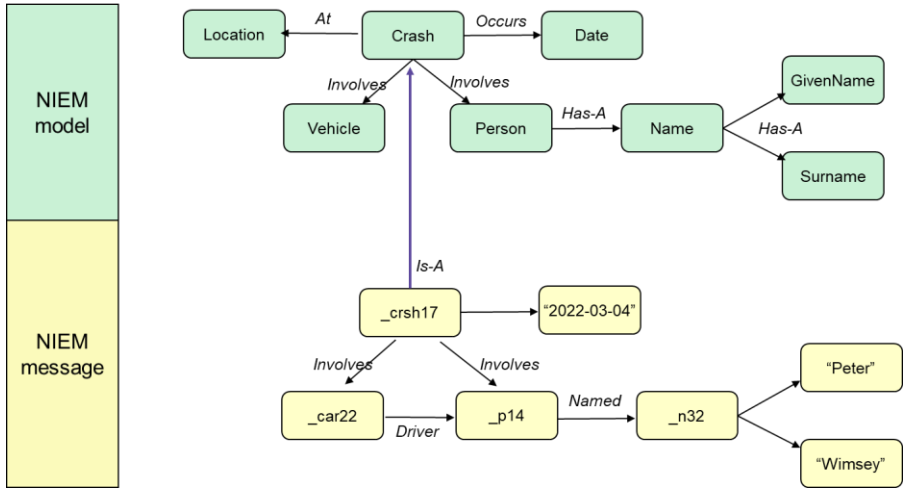
MODEL + MESSAGE = KNOWLEDGE GRAPH

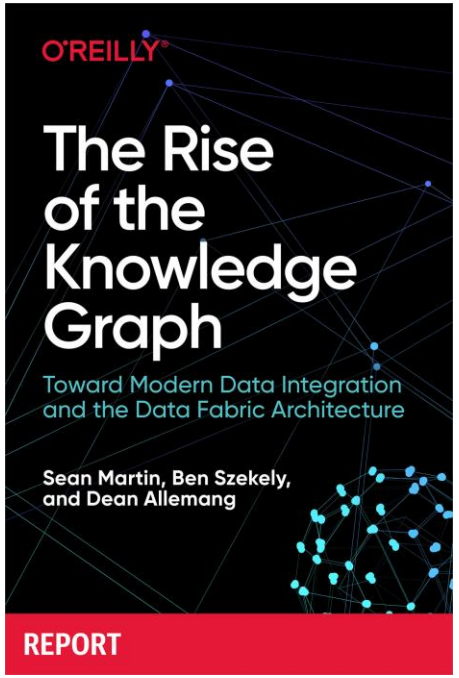


NIEMOPEN

KNOWLEDGE GRAPHS: NEXT BIG THING?

- **An important topic in data management and symbolic AI**
  - There are commercial vendors and impressive industry applications
- **NIEM 6 offers an easy "on-ramp" for developers using NIEM**
  - Most of whom know nothing of knowledge graphs, and care less
  - Their NIEM-conforming data will be accessible as a knowledge graph





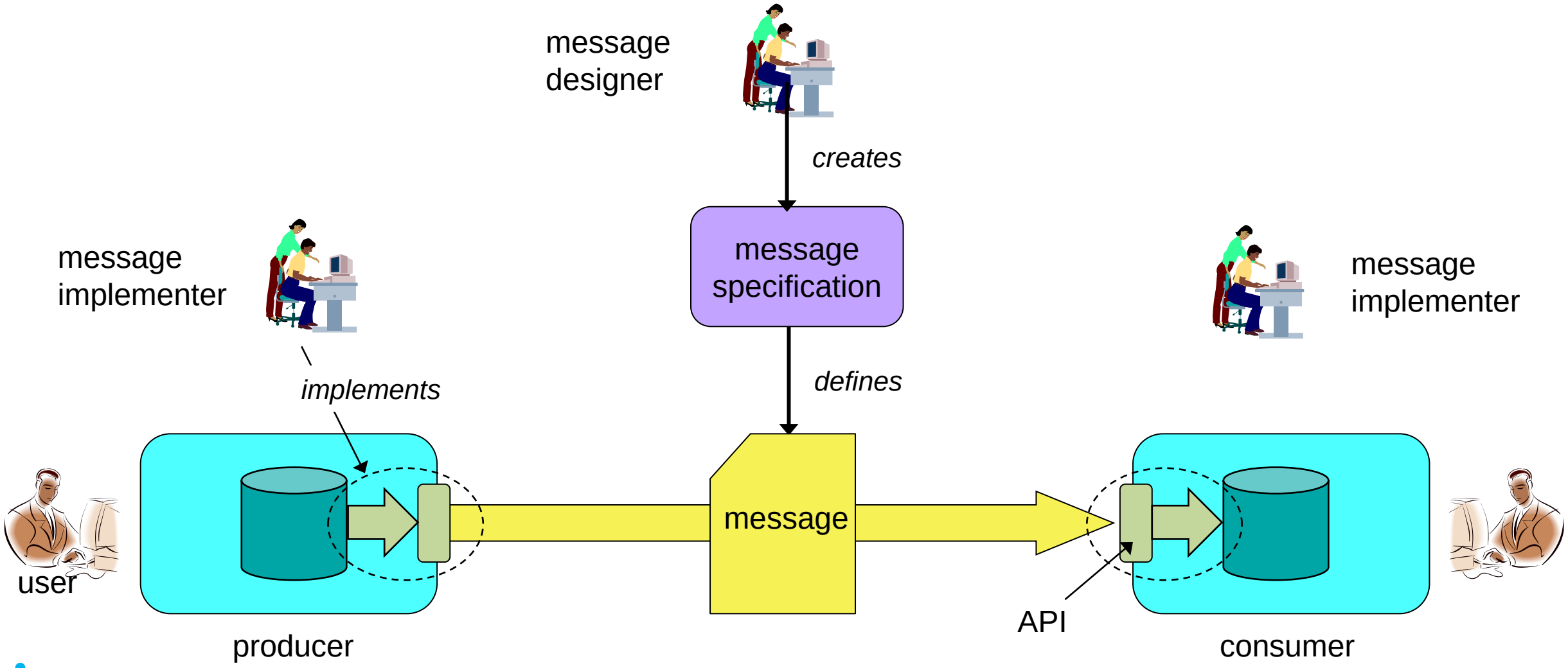
NIEMOPEN

MESSAGE SPECIFICATIONS IN NIEM 6

- **A new way to specify NIEM messages == a new IEPD format**
- **Goal is to make simple specifications easy; complex specifications possible**
- **Support for specifications that**
  - Define a single class of message (e.g. TACELINT), or more than one (e.g. request / response)
  - Define a single serialization for a message class (XML, JSON), or more than one
  - Use canonical property names (`nc:PersonSurName`), or simple properties (`lname`)
- **Convention over configuration**
  - Use the default values and your configuration file can be very small
- **We will develop free and open-source tools to work with new message specifications**
  - Test messages for conformance
  - Validate message specifications
  - Generate implementation artifacts for developers (e.g. OpenAPI document components)

NIEMOPEN

BENEFITS FOR MESSAGE IMPLEMENTERS



NIEMOPEN

IMPLEMENTER OBJECTIONS TO NIEM XSD

1. **Many namespaces and many schema documents; schema assembly is complicated**
2. **Many global elements and attributes**
   - Awkward in some integrated development environments (IDE)
   - Not possible to determine the message element from the schema
   - Extra bits in EXI encoding
3. **ISO 11179 element names and upper camel case**
   - Many programming languages have a lower camel case convention
   - Many developers prefer shorter names specialized for their particular application
4. **Data binding tools (JAXB, .NET) work poorly, or not at all**
   - Substitution groups don't work as well as `xs:choice`
   - Complex types for simple content are inefficient, awkward
5. **NIEM XSD typically not usable for XML validation in classification-domain guards**

IMPLEMENTER SUPPORT IN NIEM 6.0

- **Simple NIEM XSD: Tool support for schemas now built by hand**
  - Replace substitution with xs:choice
  - Fewer global declarations
  - Replace long type derivation chains
  - Additional constraints in XSD used for CDS validation
  - These simplified XML schemas can be generated from CMF
  - This will be in NIEM 6.0
- **Simple NIEM XML: Messages that are easier to create and consume**
  - One namespace and one global element for the message format
  - Simple property names (`lname`) instead of canonical names (`nc:PersonSurName`)
  - More work for NTAC, more benefit for developers
  - This may be in NIEM 6.0

SIMPLE NIEM XML

Canonical NIEM XML

```
<exch:CrashDriverInfo
  xmlns:exch="http://example.com/CrashDriver/1.1/"
  xmlns:j="http://release.niem.gov/niem/domains/jxdm/7.0/"
  xmlns:nc="http://release.niem.gov/niem/niem-core/5.0/">
  <j:CrashVehicle>
    <j:CrashDriver>
      <nc:Person>
        <nc:PersonBirthDate>
          <nc:Date>1890-05-04</nc:Date>
        </nc:PersonBirthDate>
```

Simple NIEM XML

```
<CrashDriverInfo
  xmlns="http://example.com/CrashDriver/1.1/simple/">
  <vehicle>
    <driver>
      <person>
        <bday>
          <date>1890-05-04</date>
```

This works because the *namespace URI* identifies a message specification with mappings from the simple property names to the canonical names

NIEMOPEN

SUMMARY

- **Transition to OASIS is next month**
- **NIEM 6.0 release is October 2023**
  - Next version of the NIEM model (core + domains)
  - New modeling formalism (CMF and XSD supported)
  - Convertible message serializations (XML, JSON, others)
  - Support for ontology and knowledge graphs
  - New message specifications (aka IEPDs)
  - Simple XML and other benefits for message implementers