Formal Rules to Produce Object Notation for EXPRESS Schema-Driven Data

Georgii Semenov

ITMO University, Saint Petersburg

Agenda

1. **Introduction. PDM systems**
2. **STEP (ISO 10303)**
3. **Product data serialization: JSON vs XML**
4. **EXPRESS to JSON mapping issues**
5. **Formal rules to produce JSON notation**
6. **Examples**
7. **Conclusions**

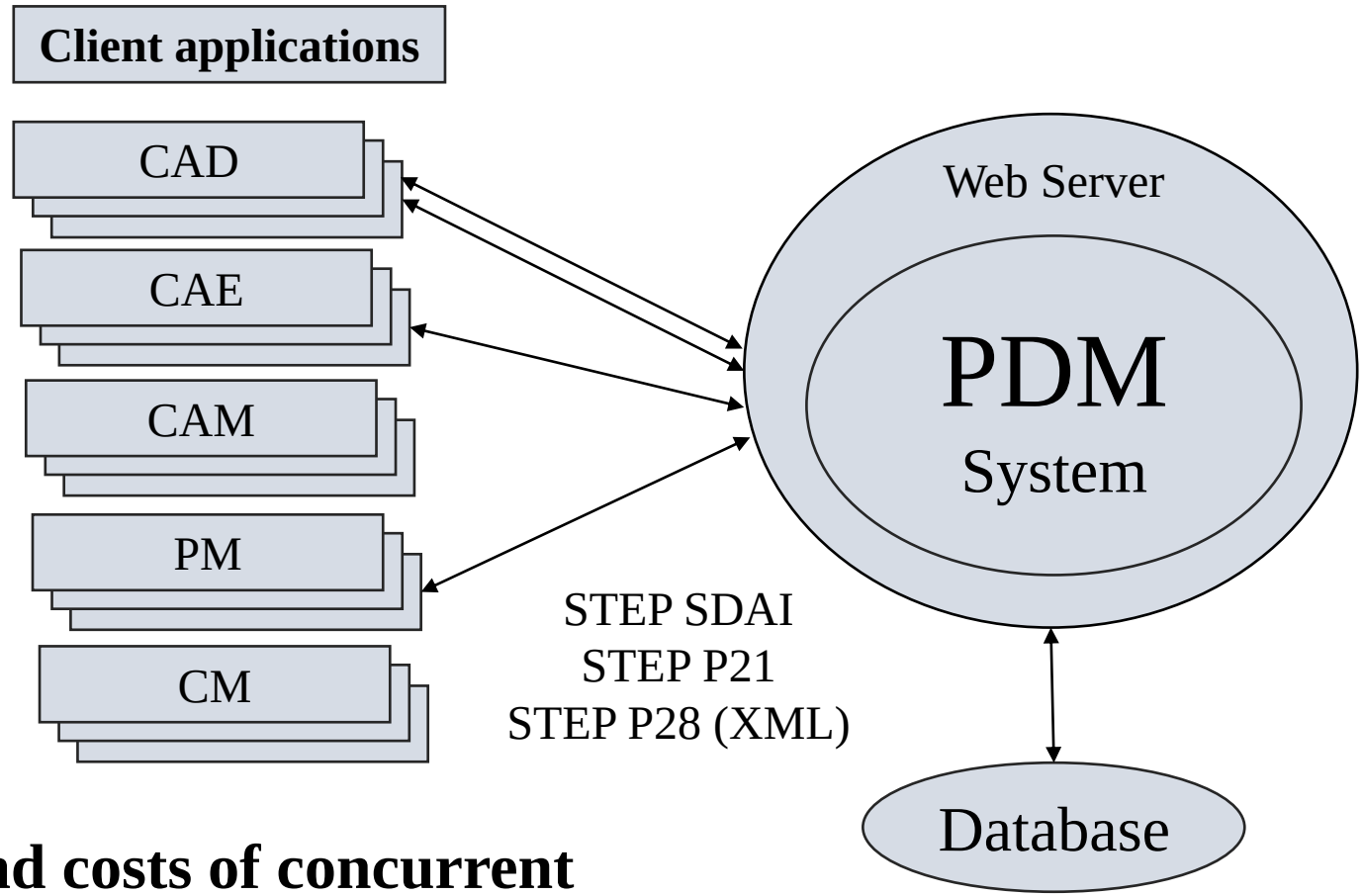
Product Data Management Systems

Popular PDM systems:

- ProjectWise
- Windchill
- Teamcenter
- Enovia

Application domains:

- Aerospace
- Defense
- Shipbuilding
- Electronics & etc.



PDM systems reduce the time and costs of concurrent design and engineering.

STEP standard family

STEP — Standard for the computer-interpretable representation and Exchange of Product data (ISO 10303)

STEP standard:

Part 1, 11, 14. Descriptions methods.
Part 21-28. Implementation methods.
Parts 41-61. Integrated generic resources.
Parts 101-112. Integrated application resources.
Parts 201-242. Application Protocols.
Parts 501-523. Application interpreted constructs.

Schema	P11. EXPRESS language		
Data formats	P21. SPF	P28. XML	**other**

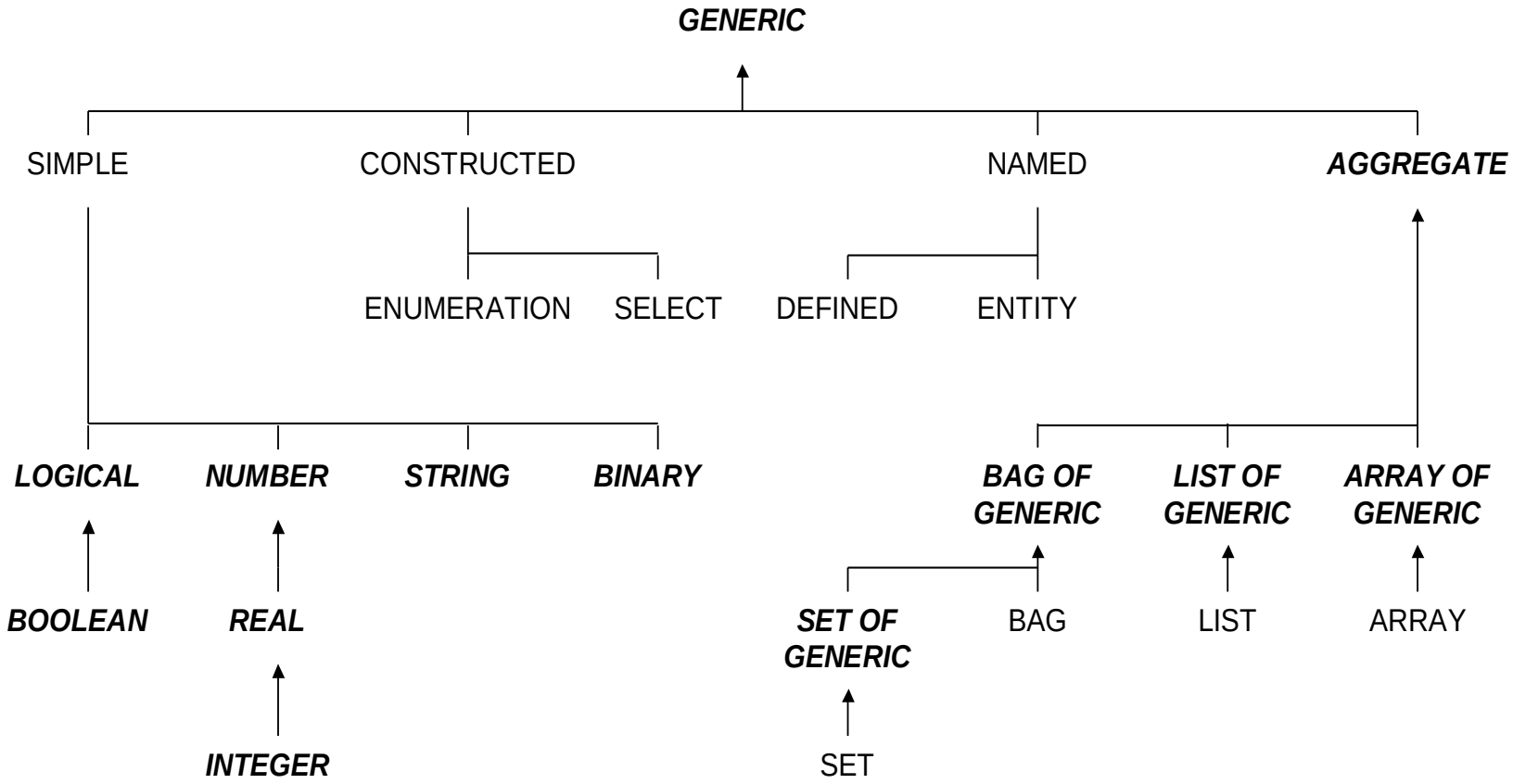
Application protocols:

- AP242 (ISO 10303-242)
- AP238 (ISO 14649)
- IFC (ISO 16739)
- P-LIB (ISO 13584)
- CIS/2
- POSC/CAESAR (ISO 15926)
- PSL (ISO 18629) & etc.

EXPRESS language

EXPRESS — declarative object-oriented data modelling language (ISO 10303-11).

- Datatypes:
  - Simple
  - Aggregate
  - Enumeration
  - Selection
  - Entity
- Constraints
  - Where rules
  - Unique rules
  - Global rules

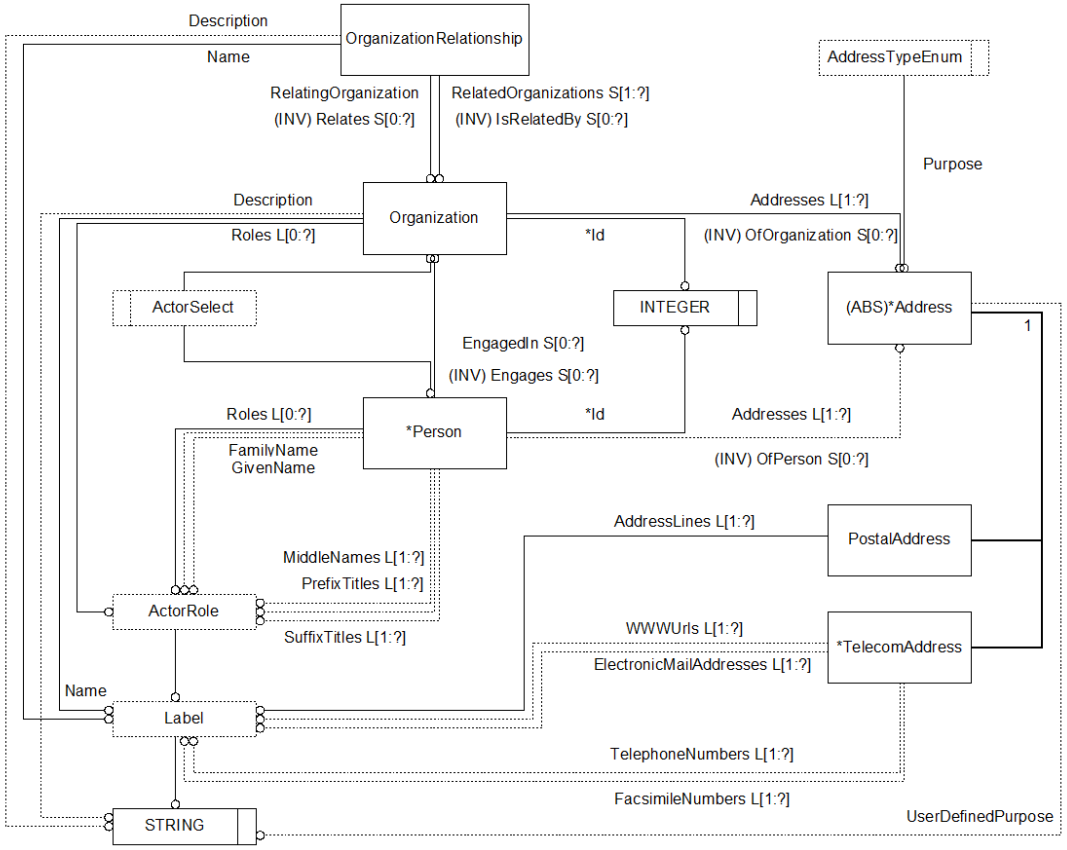


Example of EXPRESS schema

EXPRESS schema:

```
ENTITY Person;
        Id             : INTEGER;
       FamilyName     : OPTIONAL Label;
       GivenName      : OPTIONAL Label;
       MiddleNames    : OPTIONAL LIST [1:?] OF Label;
       PrefixTitles   : OPTIONAL LIST [1:?] OF Label;
       SuffixTitles   : OPTIONAL LIST [1:?] OF Label;
       Roles          : LIST [0:?] OF UNIQUE ActorRole;
       Addresses      : LIST [1:?] OF UNIQUE Address;
       EngagedIn      : SET OF Organization;
     UNIQUE
       UR1 : Id;
     WHERE
       WR1 : EXISTS(FamilyName) OR EXISTS(GivenName);
 END_ENTITY;
```

EXPRESS-G diagram:



Sample data set

SPF (Part 21):

```
#61 = Person(901, 'Pringle', 'Andrew', $,
('Mr'), $, ('Supply Chain Manager',
'Executive Manager'), (#34), (#11));
```

XML (Part 28):

```
<Person id="61">
<Id>901</Id>
<FamilyName>Pringle</FamilyName>
<GivenName>Andrew</GivenName>
<PrefixTitles><Label>Mr</Label>
</PrefixTitles>
<Roles><ActorRole>Supply Chain Manager
</ActorRole><ActorRole>Executive Manager
</ActorRole></Roles>
<Addresses><Address ref="34"/></Addresses>
<EngagedIn><Organization ref="11"/></EngagedIn>
</Person>
```

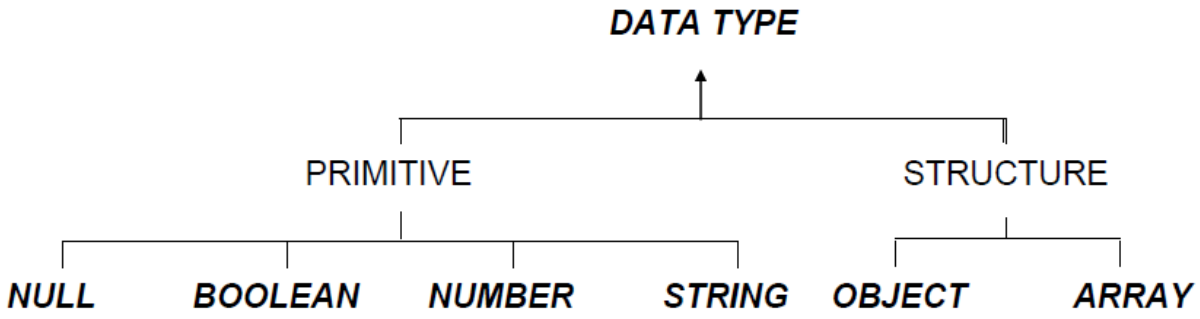
JSON representation

```
[{    "_oid":"#61",
      "type":"Person",
      "id":901,
      "familyName":"Pringle",
      "givenName":"Andrew",
      "middleNames":null,
      "prefixTitles":["Mr"],
      "suffixTitles":null,
      "roles": ["Supply Chain Manager",
                "Executive Manager"],
      "addresses":["#34"],
      "engagedIn":["#11"]            }]
```

JSON language

JSON — JavaScript Object Notation language. (RFC 7159, ISO 21778)

- Derived from the object JavaScript literals
- JSON & XML are standard Web data formats



JSON data types

- JSON document root is a JSON object or JSON array
- JSON object attribute name is a JSON string

```
{
    "null": null,
    "boolean": true,
    "number": -1.9,
    "string": "string",
    "object": {
        "a": null,
        "c": "d"
    },
    "array":[
        1,
        "a"
    ]
}
```

Valid JSON text

Problem statement

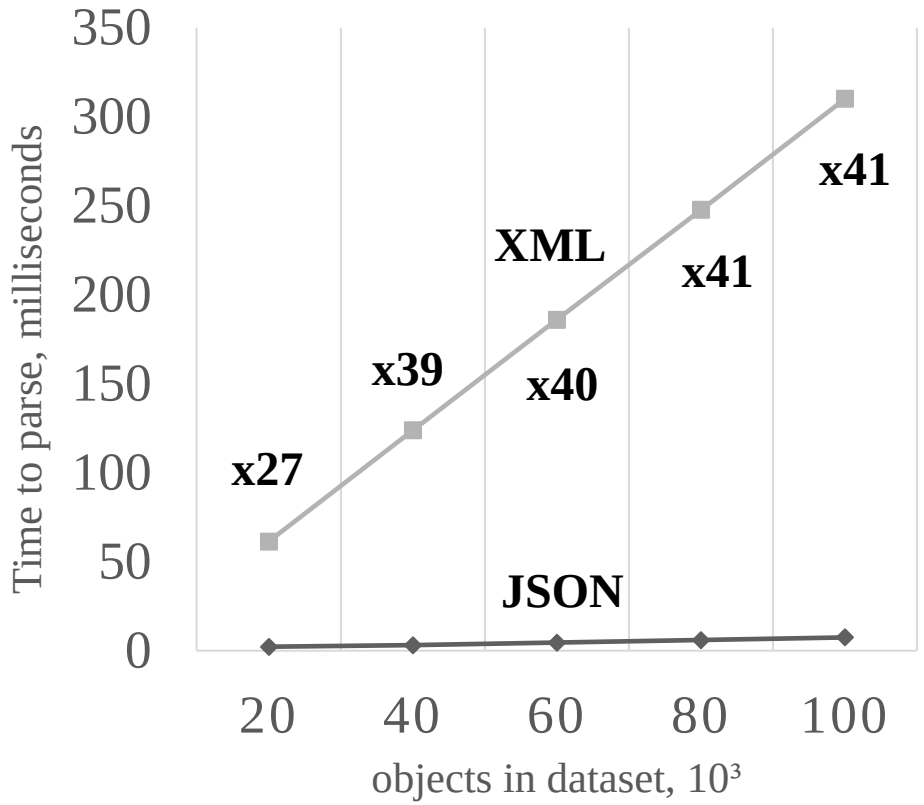
Formal rules to represent EXPRESS schema-driven data in JSON

Motivation:

- JSON is the default format for Web services
- JSON is parsed faster than XML

Requirements:

- Universality
- Unambiguousness
- Non-redundancy
- Feasibility



Time spent to parse equivalent JSON and XML datasets *

* N. Nurseitov, M. Paulson, R. Reynolds, C. Izurieta. Comparison of JSON and XML Data Interchange Formats: A Case Study. Proceedings of the ISCA 22nd International Conference on Computer Applications in Industry and Engineering, 2009.

Mapping rules for EXPRESS datatypes 1

EXPRESS datatype category	EXPRESS datatype	JSON datatype
SIMPLE	INTEGER	number
	REAL	number
	NUMBER	number
	BOOLEAN	boolean
	LOGICAL	string (“true”, “false”, “unknown”)
	STRING	string
	BINARY	string (base64)
AGGREGATE	BAG, SET, LIST, ARRAY	array
CONSTRUCTED	ENUMERATION	string (lowercase): • “enumeration_item_k”
	SELECT	object with attributes: • “type”: string (qualified select_type_k) • “value”: value of qualified select_type_k

Mapping rules for EXPRESS datatypes 2

EXPRESS datatype	JSON datatype
ENTITY	object with members: • "_oid": unique string • "type": ENTITY name • members for ENTITY attributes
ENTITY with inheritance constraints (ONEOF, ANDOR, AND)	object with standard members and: • "_prototype": array of nested objects
ENTITY attribute	string: • "_oid" of object in the document
ENTITY attribute (INVERSE SET [1:1] OF)	array of nested objects
OPTIONAL attribute	null (INDETERMINATE value) or typed value

JSON examples: simple types

EXPRESS schema:

```
ENTITY Person;
       Id           : INTEGER;
       FamilyName   : OPTIONAL Label;
       GivenName    : OPTIONAL Label;
       MiddleNames  : OPTIONAL LIST [1:?] OF Label;
       PrefixTitles : OPTIONAL LIST [1:?] OF Label;
       SuffixTitles : OPTIONAL LIST [1:?] OF Label;
       Roles        : LIST [0:?] OF UNIQUE ActorRole;
       Addresses    : LIST [1:?] OF UNIQUE Address;
       EngagedIn    : SET OF Organization;
     UNIQUE
       UR1 : Id;
     WHERE
       WR1 : EXISTS(FamilyName) OR EXISTS(GivenName);
 END_ENTITY;
```

JSON representation

```
[{    "_oid":"#61",
      "type":"Person",
      "id":901,
      "familyName":"Pringle",
      "givenName":"Andrew",
      "middleNames":null,
      "prefixTitles":["Mr"],
      "suffixTitles":null,
      "roles": ["Supply Chain Manager",
                "Executive Manager"],
      "addresses":["#34"],
      "engagedIn":["#11"]            }]
```

JSON examples: constructed types

EXPRESS schema:

```
TYPE ColorChannel = INTEGER;
WHERE
WR1: (0 <= SELF) AND (SELF <= 255);
END_TYPE;

TYPE ChannelColor = ARRAY [3:3] OF ColorChannel;
END_TYPE;

TYPE StandardColor = ENUMERATION OF (CYAN, YELLOW,
MAGENTA, GREEN, ...);
END_TYPE;

TYPE Color = SELECT (ChannelColor, StandardColor);

ENTITY Palette:
   colors: ARRAY [1:?] OF Color;
END_ENTITY;
```

JSON representation

```
[{    "_oid":"#81",
      "type":"Palette",
      "colors": [
          {"type": "ChannelColor",
           "value": [255, 255, 255]
          },
          {"type": "ChannelColor",
           "value": [0, 255, 0]
          },
          {"type": "StandardColor",
           "value": “yellow"
          }
      ]
}]
```

JSON examples: entity types

EXPRESS schema:

```
ENTITY Person
SUPERTYPE OF (ONEOF(Male, Female) AND
ONEOF(Citizen, Alien));
    FamilyName   : Label;
    GivenName    : OPTIONAL Label;
    MiddleNames  : OPTIONAL LIST [1:?] OF Label;
END_ENTITY;

ENTITY Male SUBTYPE OF (Person); ...
   married: LOGICAL;
END_ENTITY;

ENTITY Female SUBTYPE OF (Person); ...; END_ENTITY;

ENTITY Citizen SUBTYPE OF (Person); ...
    country: STRING;
END_ENTITY;

ENTITY Alien SUBTYPE OF (Person); ...; END_ENTITY;
```

JSON representation

```
[{    "_oid":"#61",
      "type":"Person",
       "familyName":"Pringle",
      "givenName":"Andrew",
      "middleNames":null,
      "_prototype": [
          { "type": “Male",
            “married":true
          },
          { "type": “Citizen",
            "country":"Russia"
          }
      ]
}]
```

Conclusions

Thus, the general rules for producing JSON notation for EXPRESS schema-driven data are formulated and presented with explanatory examples. **The rules allow to produce a universal, unambiguous, non-redundant data representation in the JSON file format.** The universality of the rules in relation to arbitrary schemas formally specified in the EXPRESS language facilitates their widespread adoption in Web services dedicated to manage product information data in various industries.

The directions of future research involve practical aspects of the implementation of the proposed rules as a part of the existing and emerging PDM systems to accelerate exchange, interpretation and processing of complex engineering data.