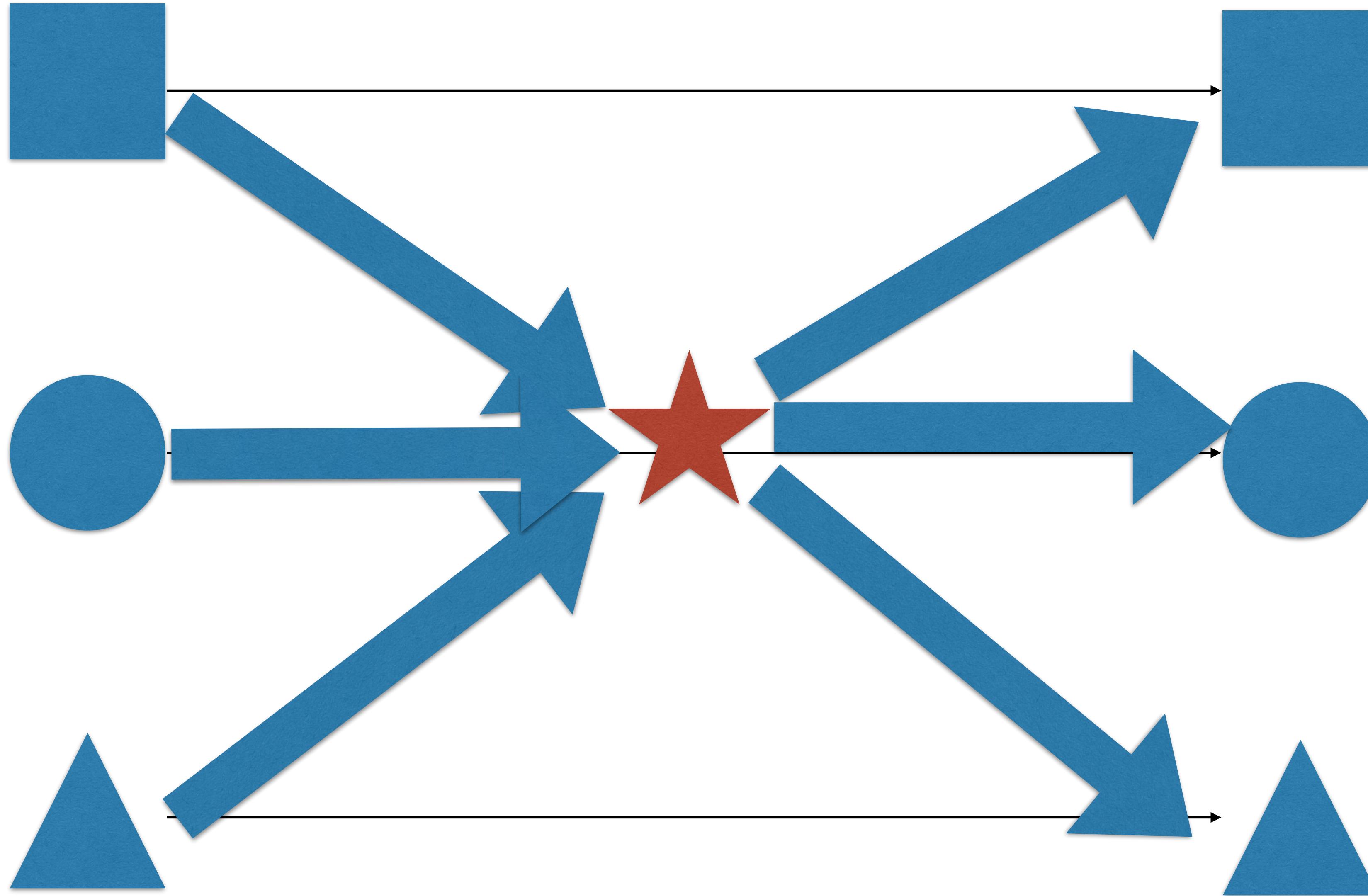


Derivation and Instantiation: Two dimensions of model evolution

T2TRG: Thing-to-Thing Research Group Interim meeting,
January 30th, 2025, Online
Carsten Bormann

SDF: $(n-1)^2 \rightarrow 2n$



Information and Data Modeling, Interaction Modeling

- An SDF model describes interaction opportunities for a **class** of devices
- Devices and their models **evolve** (revisions, changing functionality)
- **Variants** of a device model are needed, without replacing base model
Ecosystem-specific information needs to be added
→ All while base model evolves, too
- Device classes are **instantiated** (based on a class, i.e., model-**controlled**),
adding information beyond interaction opportunities (affordances),
often **ecosystem**-specific
- **Messages** (data from/independent of interaction) need to be modeled

Derivation

Instantiation

SDF: Class-level information about digital interaction opportunities (affordances)

- **Things = digital side, physical side**
SDF models provide an **Interaction model**
for the **digital side** of a **class** of Things, in terms of:
 - **Named Interaction Opportunities (Affordances)**
 - Supplying **Information Models** for input and output data
 - **Named data**
 - *Some semantic* information (e.g., units)
- SDF models have no specific information about device **instances**

Evolving SDF models

- Versioning
 - No rules for how versions of the same lineage have to be related (Compare YANG's elaborate BC/NBC rules)
 - No way to reference a particular subrange of versions
 - Who controls a lineage?
- Derivation
 - sdfRef: Structural, not semantic
 - Relationship of base and derived unclear

sdfRef: Structural Derivation

- Reference (parts of) base model:
`"sdfRef": "foo:#/sdfData/temperatureData"`
- Override by a “merge-patch” (RFC 7396) process:

```
  "sdfObject": {  
    "SafetySwitch": {  
      "sdfRef": "cap:#/sdfObject/Switch",  
      "sdfAction": {  
        "toggle": null,  
        "emergencyStop": ...,  
      }  
    }  
  }  
}
```

sdfRef: Structural Derivation

- 😊 Single, very **simple** mechanism can cover 80 % of objectives
- 😞 Base model has no way to control what is derived from it
(Or support that derivation in any way)
- 😞 Innocuous changes in base model can break derived models
- 😞 Hard to process other than “literally”

Is SDF Derivation ~ Subclassing?

- Subclasses in Programming Languages are **semantic** derivation
- Governed by rules such as Liskov Substitution Principle (LSP)
Object of subclass can be substituted for object of base class
(behavioral subtyping)
- E.g., can I use a dimmer in place of a switch (A: mostly, but...)
(Is that a question about digital or physical side?)

Use cases for derivation

- Variants of a device
 - Parameters (e.g., voltage range)
 - Features
- Revisions of a model
 - Fix bugs (should work in place!)
 - Modernize interface (e.g., ASCII → Unicode)
 - Add features
- Adaptations
 - E.g., to ecosystem (BlueTooth vs. Matter vs. ...)

Picking up base model evolution

1. Reference **specific** revision of base model

- No “automatic” pickup of bug fixes
- Can be fully controlled

2. Reference **any** revision of base model

- Automatic, but dangerous

3. Specify **subrange** of base model revisions

- Can be mostly controlled (semver...)

Instance Information

Information about:

- **Identity** of the thing (on Thing's **digital side**)
+ any Supplementary Identity Info (with **twin**/asset management/...)
- **Purpose** in Life, links, ...
- Information about **physical side**
(location, manual electrical settings, ...)
— lives with twin, *possibly* with thing itself

Instantiation is an Affordance (!)

- Interaction model may want to describe instantiation
- What parameters are needed (e.g., IP address)
- What interactions are performed during instantiation process
- What information is produced during instantiation process
- (Physical Device probably needs to exist beforehand)

Cheating: Describing Instance via Derived Model

- Make an instance by:
 - Referencing base model (sdfRef)
 - Inserting instance information as constant properties into derived model
- Problems:
 - No control
 - Instance information may not be constant → need to update model
 - Not all instance information is on device (or should be obtained there)

Describing messages ("data in flight")

- SDF describes **information model** of messages that go with interactions (inputdata/outputdata)
- **Ecosystem** may or may not provide rules mapping message IM to message **data model** and/or **serialization**
 - (Also applies to structural aspects of interaction)
- Possibly add ecosystem-specific information about message mapping
 - (Add by derivation? How does a class find a subclass?)