Messages canonical representation

An answer to some typical messaging problems

Table of content

What is canonicalization ?

Standardise, normalise, simplify

"In computer science, canonicalization is a process for converting data that has more than one possible representation into a "standard", "normal", or canonical form. This can be done to compare different representations for equivalence, to count the number of distinct data structures, to improve the efficiency of various algorithms by eliminating repeated calculations, or to make it possible to impose a meaningful sorting order."

Wikipedia

Canonicalization can also be useful to gain more control on message structures designed by the external world

Typical messages problematic

Lack of control, high impact

- No control on the design of the business messages handled: Message structures are imposed by the external world
- No control on the business message design lifecycle e.g. why, how and when they change.
- Difficulty to anticipate message structure changes
- And these changes impacting all the company applications

Besides, messages are often poorly designed from an IT stand point:

- Designed by non-technical persons (e.g. business actors, lawyers)
- Not designed with automation in mind (by software systems):
- No transactional support (correlation algorithm)
- ...

Advantage of a canonical form

Isolate from external world and gain control

A way to gain control, improve flexibility and reactivity to changes and to minimize the impact on applications is:

- Push the complexity and specificity in interfaces (message format adapter) at the boundaries of the company "environment"
- Internally only handle and manage the canonical form of messages (applications would know only the canonical form)
- Canonical form will compensate for message structure lacks:
  - Transactional support
  - Unique identification
  - Business correspondent support
  - …

So when the format of the message changes, only the interface is impacted, not the applications (provided that the message logic does not change)

Driven by models

A central specification point

The usage of models provides a central point for definition of messages structure from which implementation details are derived: XML, XSD, database layout, validation rules, …

Roadmap:

- Start from current business messages
- Derive atomic messages (data) elements
- Derive canonical (normalized) message structures in a model
- Add transactional support
- Derive syntactic and semantic validations
  - Wellformedness
  - Logical coherence
  - …
- Derive XML structures and associated schemas
- Derive a logical data model (basis for database design)
- Derive message format adapter algorithm (specific ↔ canonical)

Message building blocks

Data elements

- Proper definition of data elements is key
- They can be regrouped for database modelling
- They can be regrouped in messages
- If this is done properly, this will simplify software design and software/database interfaces:
  - No heavy data transformation
  - One-to-one data element mapping from message canonical format and database fields
  - …

All data elements handled in the company must be identified and included in an EA model. They will then drive the design of the business canonical format and the logical data model from which database implementation will be derived

Proof of concept

Start with a business domain

- Select business messages from a specific business domain
- Identify and model all data elements
- Model messages
- Model the transactions
- Produce associated XML and schemas (automation)

And:

- Data element level validation rules
- Message level validation rules
- Transaction level validation rules (state machines)
- The logic to map the specific to canonical message format

But also:

- Produce the associated business objects
  - Logical data model (objects properties e.g. person name, last name, …)
  - Elementary business functions (object methods e.g. create, update, delete person)

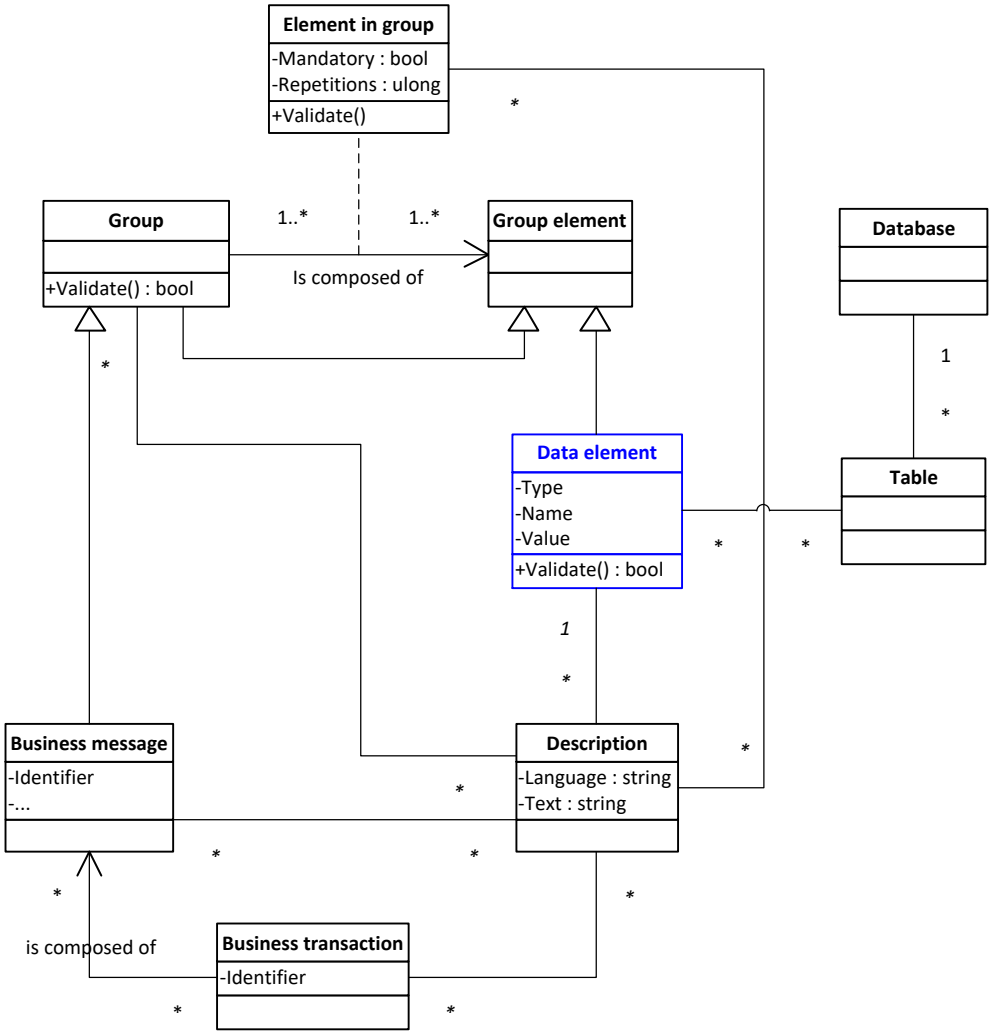
End of presentation

Spare slides

- Message modelling

Message modelling

And reuse in database design



Validating a message consists in invoking the *Validate* method of a message which in turn invokes recursively the group and data elements *Validate* methods

The validation of a data element is defined once for all and reused everywhere i.e. for database design

Model drives implementation