BLOG POST

Data Mesh to Go: How to Get the Data Product

Using DDD artifacts to get a data product for data mesh

December 09, 2021

You know what a data mesh is? You understand its basic principles? But you don't know how on earth to get the data product? Then I will show you how to extract your data product from your Domain-driven Design (DDD) artifacts.

9 minutes reading time

PHILIPP BEYERLEIN

CONTENT

↓ *Learn from the Real World*

↓ *What Have We Learned?*

When you are looking around in the data mesh ecosystem, you will often read that having a data product defined in your core domain is one basic requirement. You can find a good general definition of a data product in the post "Data Mesh Principles and Logical Architecture" by Zhamak Dehghani. Now you might be at the point where you know the theory about it, but working with it in your project is quite hard. You have a lot of design artifacts which could be used as the source for a data product. You may be thinking about adding some analytics skills to your knowledge board before going any further. This might help you in the future, but I guess you won't need it now. So let's try to get a strategy to define a good data product based on domain-driven design artifacts.

A domain-driven design environment follows the modern design principles of today's application landscape. How can you check if your environment qualifies? Tick the following points that apply to you:

- A complete event storming map with domain events, aggregates, etc. exists
- Bounded contexts have been defined based on that map
- The team has a clear understanding of your core business domain and possible subdomains
- The team understands the business of the system and speaks a ubiquitous language
- The team has detailed knowledge about the aggregates and domain events

Are you unhappy that you do not have a design which follows the principles of DDD? That's not really a problem as long as you have a deep knowledge about your environment's core domain and its design. In the following section I use a lot of wording from the DDD domain. If you want to familiarize yourself with those terms, I recommend the following reading: DDD Condensed

Learn from the Real World

To make it easier to understand, we will compare physical products with digital products. We will start with the example of a physical product, which will be one from the fashion industry. There will be a fashion company, which has different types of product streams: footwear, jackets, and so on. Each of these product streams represents a subdomain of the whole fashion company domain. As our example we take shoes as a physical product of the footwear domain.

You might know the product attributes of shoes that help you to definitely identify if something is a shoe, not a jacket or something else.

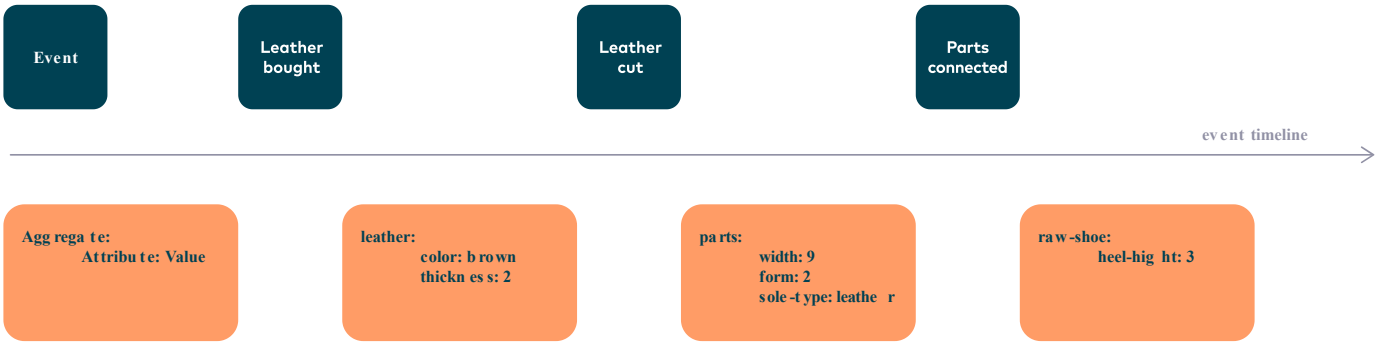
A shoe can have the following attributes and example values:

```
shoe:
 form: high heel
```

```
width: 9
color: brown
heel height: 3
sole type: leather
```

So you can see some attributes like color, which not only identify a shoe, but also differentiate between instances of the same shoe. So the combination of all these attributes gives you a good picture of which type of shoe you have. Let's see where the fashion company gets these values and attributes.

First, we should take a look at the event stream of the shoes' production:



Event-Stream-Shoe

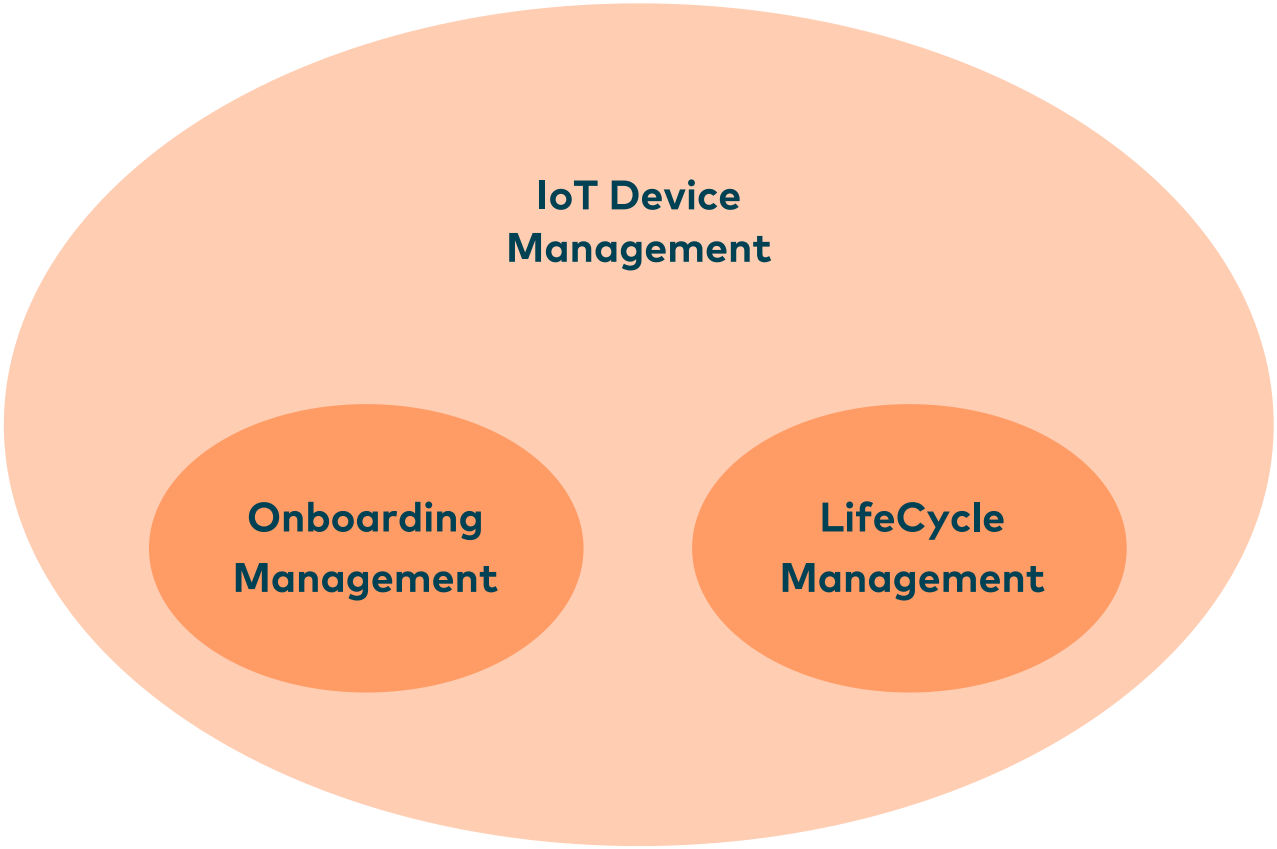
As you can see, the different values and attributes has been spread over all aggregates. Most of them you can just discover when the correlated business events happen.

So far, so good – but how does that help us to develop a data product? What the fashion company knows for sure is that they have produced a brown high heel based on the aggregates of the production process.

In your case, the aggregates of your business events contain the attributes of your data product. You think it's really easy by adding all aggregate attributes to a data product, that's it; then you reduce the quality of your data. The data you can obtain will contain a lot of information that is not relevant to measuring the value of your business. In the example above you see the thickness attribute. This information is necessary for the leather cutter to know which tool to use. That attribute might be helpful in identifying the shoe, but a metric "sold shoes in relation to the leather thickness" might not say anything about the success of that shoe. If it is a rough working shoe it might be different, but this is a high heel made with as little leather as possible.

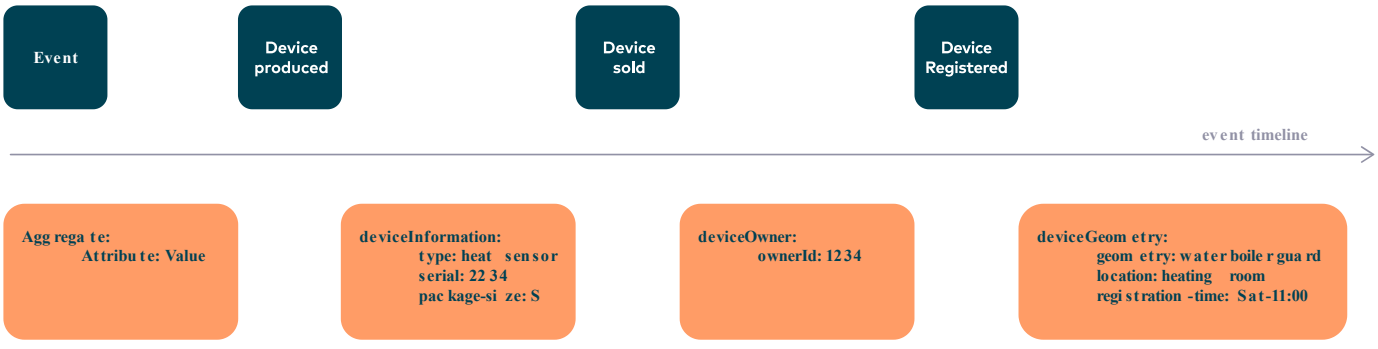
So keep in mind for your data product: In the context of your ubiquitous language, find the attributes which are necessary to identify and measure your business value.

Now it is time to take this logic to the example of a digital product. We will use an IoT device management system as the domain for design artifacts to get a data product out. What could be the data product? You might guess it will be device? We will see. Let's go deeper into this domain. The following context map shows the subdomains of an IoT device management domain:



IoT-Device-Domain

As you can see, the IoT device management would be split into two different subdomains. I guess you could find more if you ask the right business expert. For our example two are enough. Each of these domains produces its own data product, which is defined by the business value of the subdomains. Let us dive into the onboarding domain to find the data product. To get it, we need the event stream with aggregates:



Event-Stream-IoT

Now we have a brief insight of the business of that device's onboarding. To get the data product, we should find an aggregate which has the attributes necessary to identify and measure the core domain's business value. In this case the business expert might want to know: How many devices have been registered or not, grouped by owner and type? Only the business value of the subdomain can give the answer. The aggregate the expert asks for is a type of registration, as it carries that information. The data product which the onboarding subdomain might produce is a registration with the following attributes and values:

```
registration:
 serial: $deviceInformation.serial
 type: $deviceInformation.type (heat sensor)
 ownerId: $deviceOwner.ownerId (1234)
 registered: $deviceGeometry.registration-time exists ? yes : no (yes)
```

All attributes are references to different values from the aggregates. You might be wondering about the serial attribute, because the shoes do not have an ID. A finished physical product is unique by nature. For example the leather can differ slightly in color, so each shoe will be different. In the case of our fashion company it might not add any benefits to put a serial number on the shoes. The only important number to measure their business might be the total number of produced pairs of shoes. However, with the IoT devices onboarding it is required to have its serial in the data product, since the registration can be updated. Thus, it is necessary to uniquely identify each single registration.

Now we have the basic structure of the data product for the onboarding management in our IoT device management domain that can be given to the analytics consumer of the data mesh ecosystem.

What Have We Learned?

As you can see, it is not a data analytics black art to find data products. The central question you have to ask is: Which information is necessary to measure the business value of the core domain of a single system? You should get the answer to this question from your business experts.

The easiest way to work on this answer is to think about your data products as part of the event storming sessions. At these sessions you talk with business experts, build the core of your ubiquitous language, define aggregates and events. When all of these design elements are on your event storming timeline, you are really deep inside your core domain. Finding the name and attributes of your data product is only one small step further during the design creation process.

Now you have your data product, but what's next? You should start to think about how to distribute it. This will be the topic of the next "Data Mesh to Go" post.

TAGS

Data Mesh Software Architecture Domain-driven Design

PHILIPP BEYERLEIN

Senior Consultant

Philipp Beyerlein works as a Senior Consultant at INNOQ. He is an AWS specialist and fan of simple, efficient and modern software development. He likes to challenge current trends to find the real need for technology to get the best solutions to solve problems. His technology focus is on developing and running applications for the cloud (AWS).

COMMENTS

Share on

RECOMMENDED

BLOG POST

The language of maths is not the language of your business

Abstractions from category theory can be powerful. But there are reasons why you may want to keep your domain model free of them.

DANIEL WESTHEIDE

BLOG POST

Domain Events vs. Event Sourcing

Why domain events and event sourcing should not be mixed up.

CHRISTIAN STETTLER

BLOG POST

Is Domain-driven Design overrated?

STEFAN TILKOV

Get in touch

Use our contact form or send us an email to info@innoq.com.

You can send us encrypted emails, too. Just use our S/MIME certificates (.cer, .p7b, .pem) or our public PGP key.

Name

Email

Message

Anti-Robot Verification

Click to start verification

FriendlyCaptcha

Send

Links

Blog & Articles

Talks

Podcasts

Technology Lunch

Primers

Books

Management

socreatory – The Software Creators' Academy

Contact

Privacy

Legal Notice

Services

Strategy and Technology Consulting

Digital Product Development

Software Architecture and Development

Digital Platforms and Infrastructures

Knowledge Transfer, Coaching and Trainings

Offices

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Tel (+49) 2173 3366 0

DIRECTIONS

c/o Design Offices
Königstorgraben 11
90402 Nürnberg

DIRECTIONS

Ludwigstr. 180 E
63067 Offenbach

DIRECTIONS

Kreuzstr. 16
80331 München

DIRECTIONS

Wendenstraße 130
20537 Hamburg

DIRECTIONS

Ohlauer Str. 43
10999 Berlin

DIRECTIONS

innoQ Schweiz GmbH

Schutzengelstr. 57
6340 Baar
Tel (+41) 41 743 01 11

DIRECTIONS

Hardturmstrasse 253
8005 Zürich

DIRECTIONS