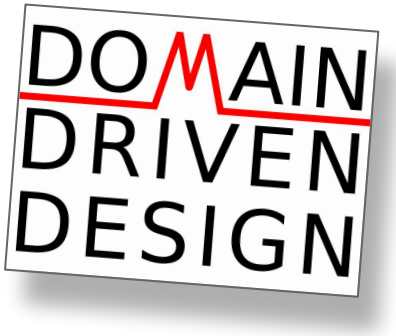
Refactoring Toward Deeper Insight

DDD Cologne Bonn Meetup / Jul 2, 2020

Christoph Baudson / @sustainablepace

REWE digital

Christoph Baudson



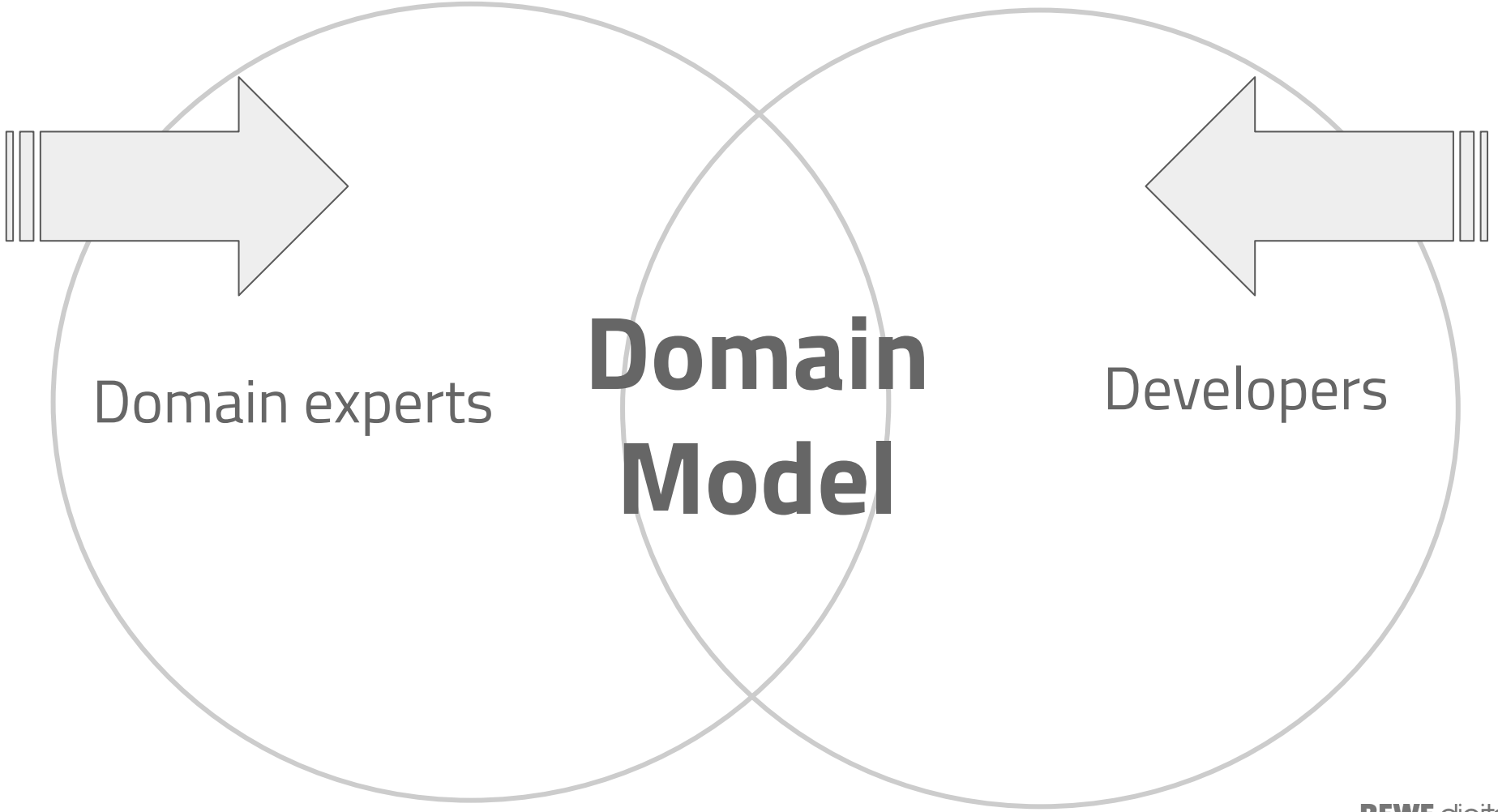
- **Software dev** at **REWE Digital** since 08/2015
- **Organizer** of the **DDD Meetup** Cologne
- @sustainablepace / sustainablepace.net

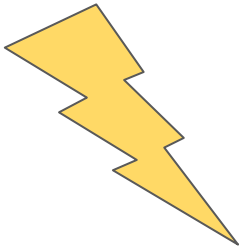
1. Some theory

2. Example

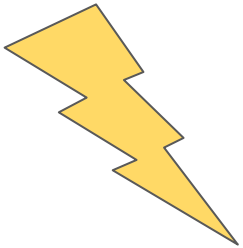
   a. The domain

   b. Let’s code!


Domain experts
Domain Model
Developers



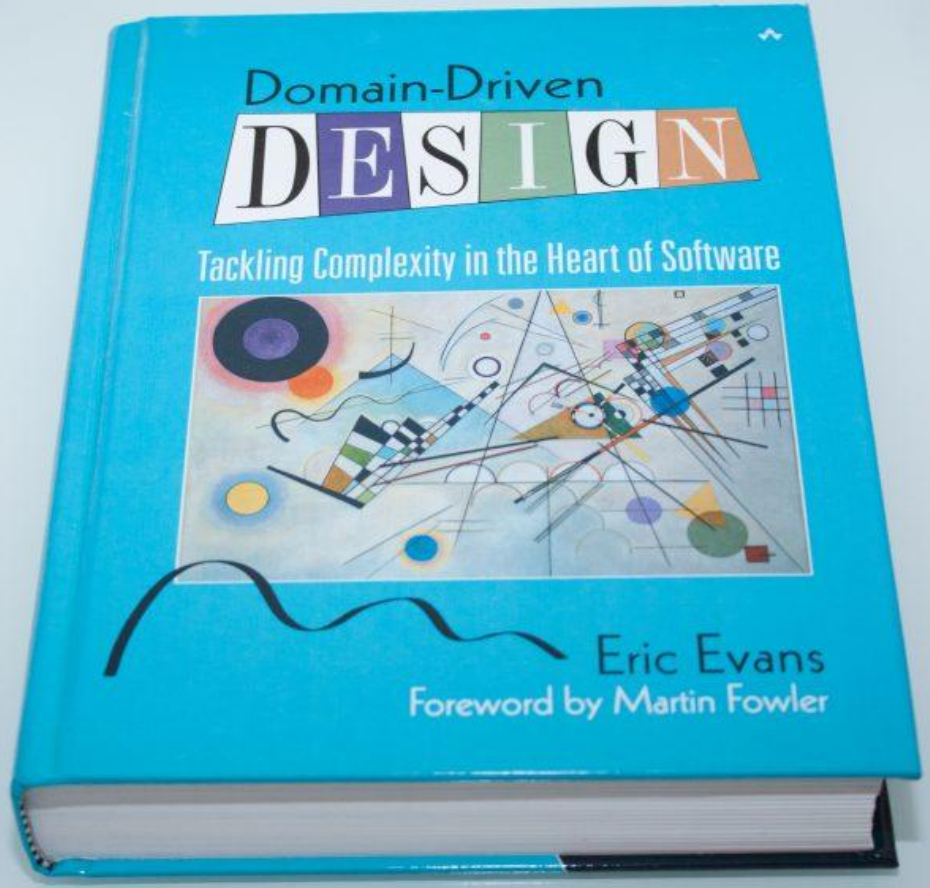
Misconception!

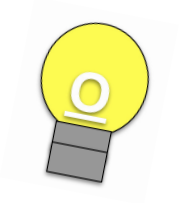


Modeling in Domain-Driven Design is a

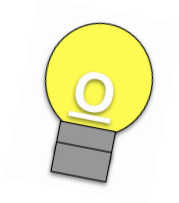
waterfall-like **big design up-front**,

and therefore **not agile**

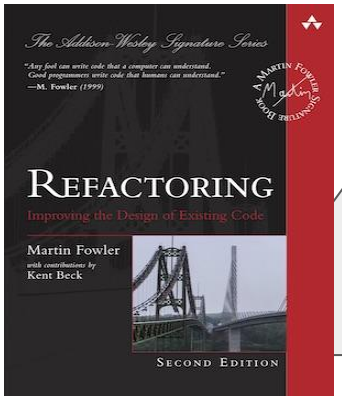

Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler



Insights



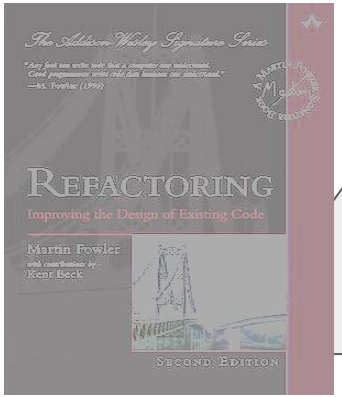
- No one gets the model correct the first time.
- The model changes over time.
- The model is never done.
- **The model needs to be refactored throughout the development lifecycle.**

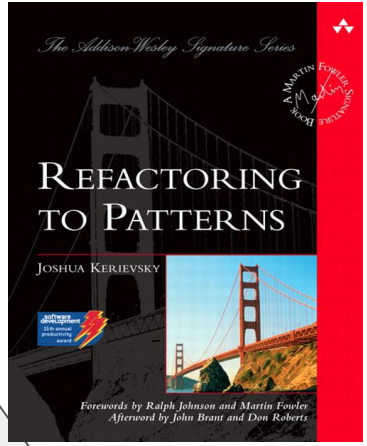


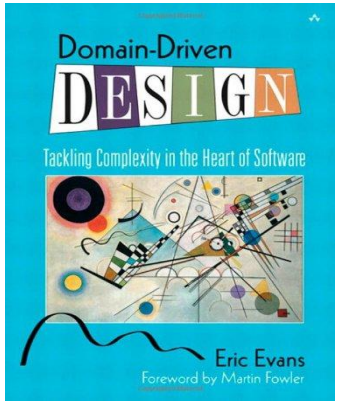
Micro-Refactorings

Refactoring to patterns

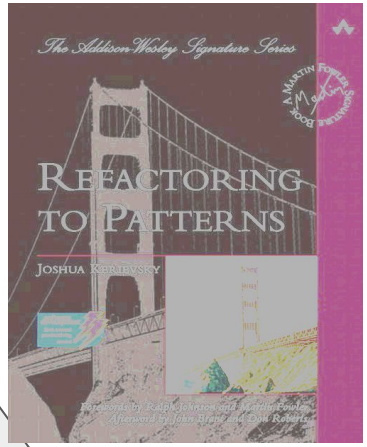
Micro-Refactorings



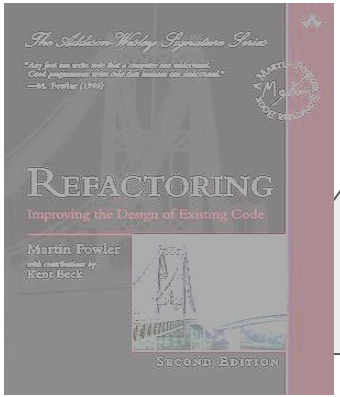




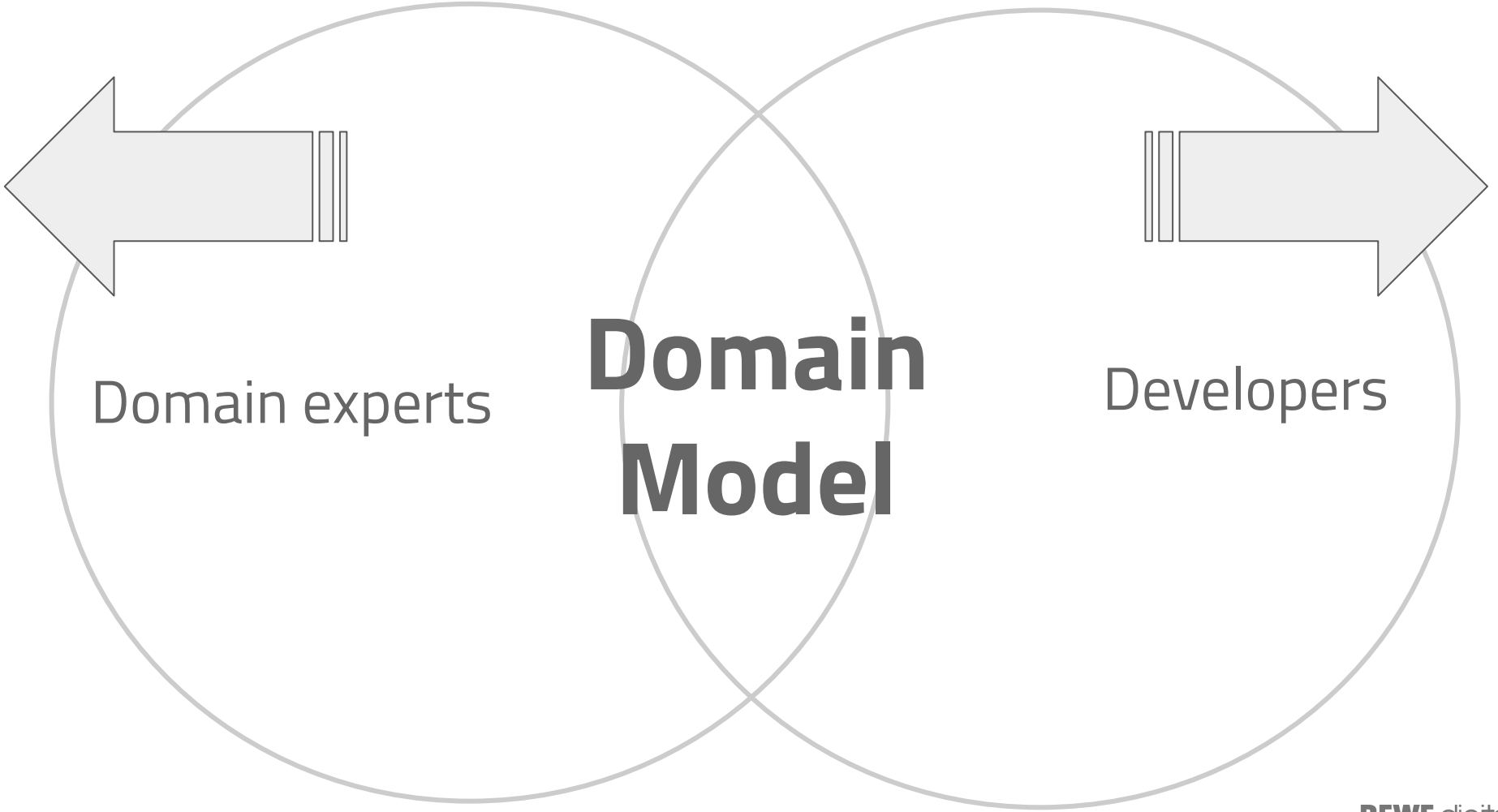
Refactoring to deeper insight

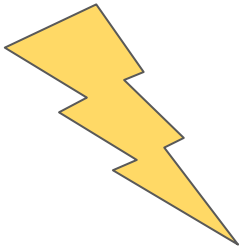


Refactoring to patterns

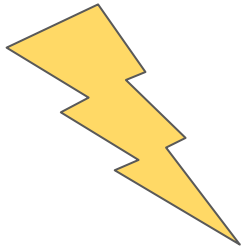


Micro-Refactorings


Domain experts
Domain Model
Developers

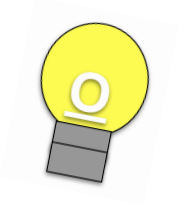


Misconception!

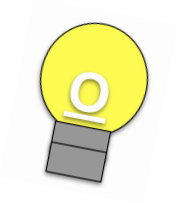


When doing **µServices**,

doing a **rewrite is easy**



Insights



- The domain knowledge is lost after rewriting
- All dialogue has to happen all over again.
- It’s hard to guarantee it will work as before.
- Doing the same thing twice and expecting improvements is foolish.
- **The domain model is an asset!**

~~1. Some theory~~

2. Example

a. The domain

b. Let’s code!

Einfache Pfandrückgabe

Den Pfandautomaten füllen wir für dich: Wenn du Getränke bei uns bestellst, kannst du dein Leergut dem Fahrer einfach wieder mitgeben.

0 KCAL

NO SUGAR

FSEKU

8734 2642

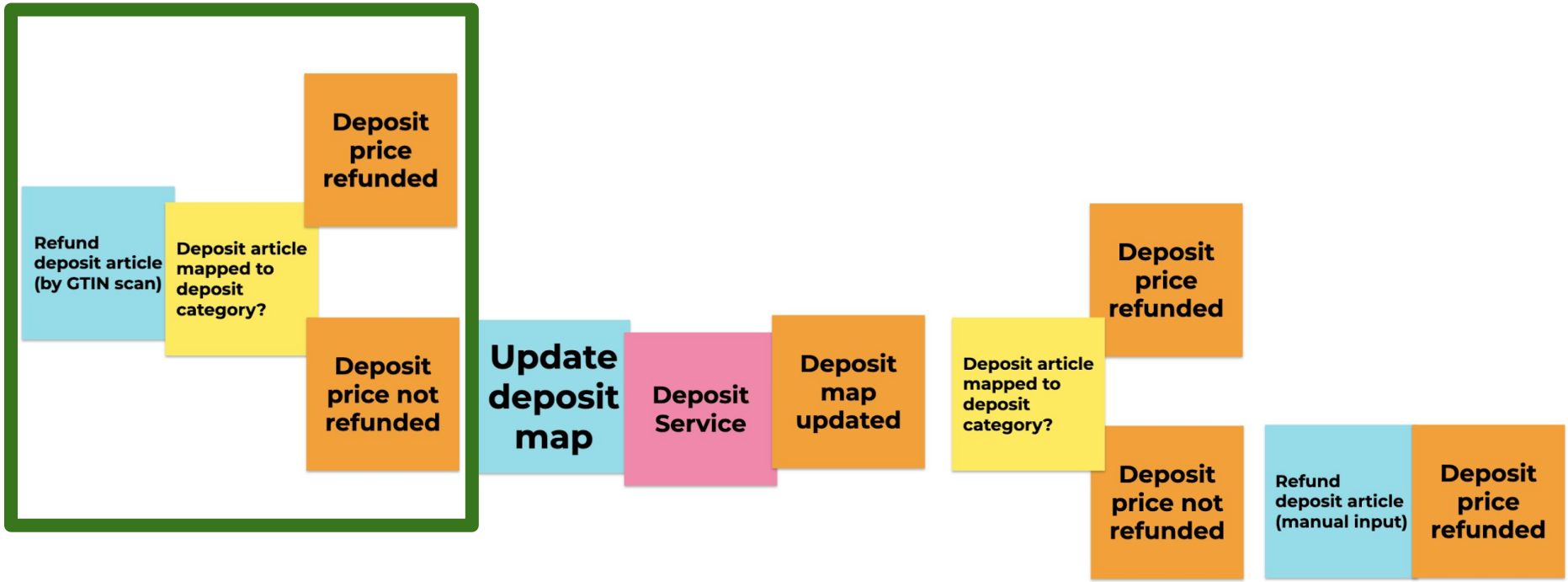
Pfand €0,25

(191)

100 g	0,5 %	*
1 mg	0,7 %	*
0,6 mg	0,1 %	*
0,6 mg	0,003 %	*
0,6 mg	0,1 %	*
100 mg	0,003 %	*
Nährstoffbezugswerte		

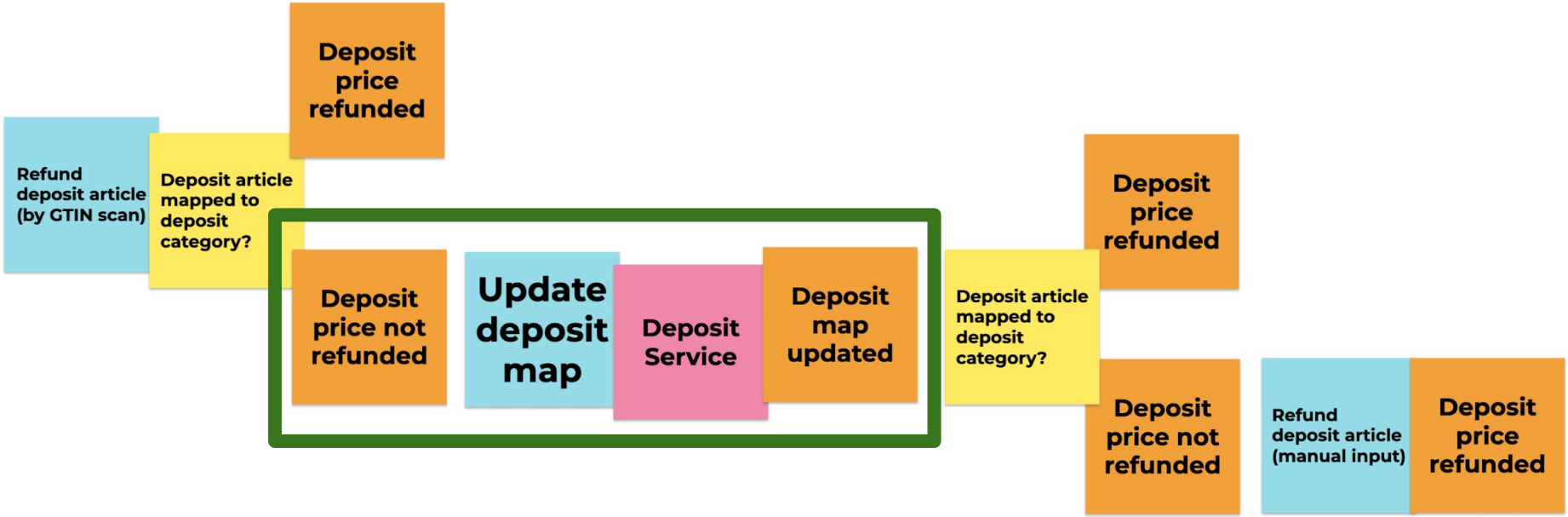
PER 100 ML:

TASTE AND


Refund deposit article (by GTIN scan)
Deposit article mapped to deposit category?
Deposit price refunded
Deposit price not refunded
Update deposit map
Deposit Service
Deposit map updated
Deposit article mapped to deposit category?
Deposit price refunded
Deposit price not refunded
Refund deposit article (manual input)
Deposit price refunded

Deposit Categories


```
[
  {
    "sortOrder": 1,
    "description": "Bierflasche, Glasflasche",
    "imageId": "1049143",
    "gtin": "21049149",
    "articleId": "5ee443fb-c28f-4052-a3ca-35ee34d48695",
    "refundPrice": 8,
    "unit": "Cent",
    "currency": "EUR",
    "reusableContainer": true,
    "vatCode": "V",
    "maxQuantity": 200
  },
  {"sortOrder": 2...},
  {"sortOrder": 3...},
  {"sortOrder": 4...},
  {"sortOrder": 5...},
  {"sortOrder": 6...},
  {"sortOrder": 7...},
  {"sortOrder": 8...},
  {"sortOrder": 9...}
]
```

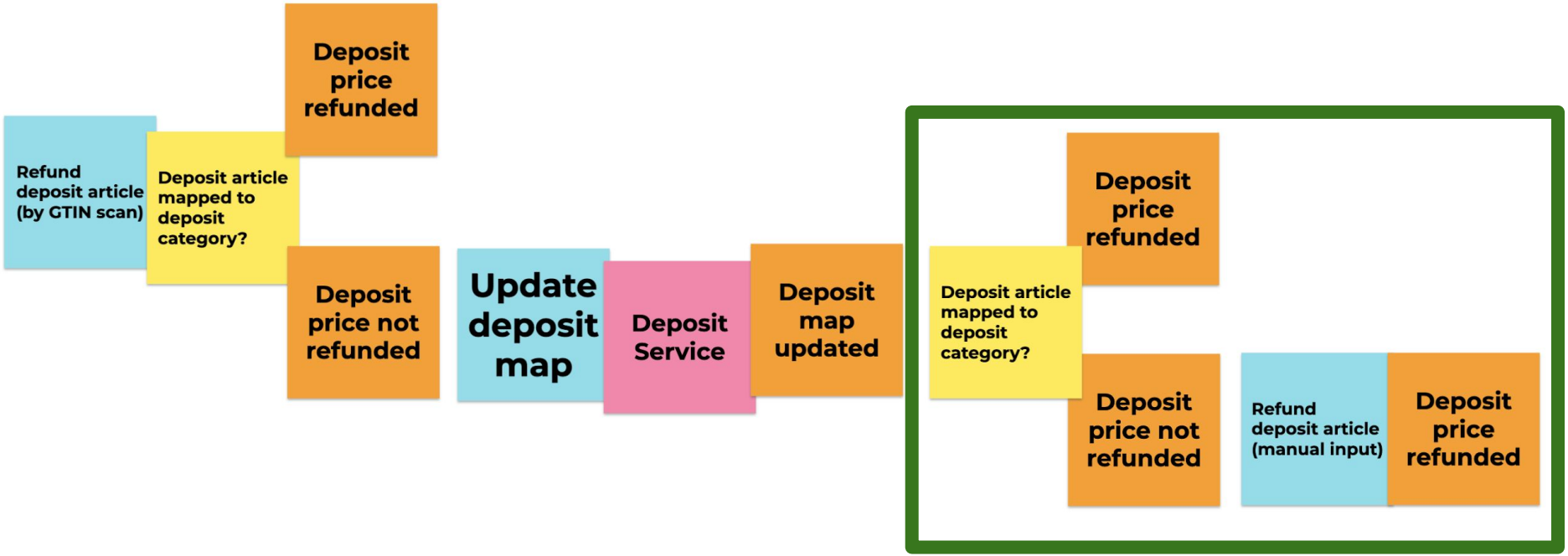

Refund deposit article (by GTIN scan)
Deposit article mapped to deposit category?
Deposit price refunded
Deposit price not refunded
Update deposit map
Deposit Service
Deposit map updated
Deposit article mapped to deposit category?
Deposit price refunded
Deposit price not refunded
Refund deposit article (manual input)
Deposit price refunded

Deposit Map

Updates after 2 weeks


```
{
  "dataFrom": "2020-07-01T06:20:58.541933Z",
  "dataDrop": false,
  "deposits": [
    {"id": "5ee443fb-c28f-4052-a3ca-35ee34d48695"...},
    {"id": "79eb1b84-ab64-48da-a800-0eed8b146dfd"...},
    {"id": "d1c3ba12-bee5-4e7f-af71-7156cbf6dec5"...},
    {"id": "6aaea08d-cfc2-4a39-a667-ad8915b34d55"...},
    {"id": "8de0b7a2-e509-4276-9b63-fd875c64324e"...},
    {"id": "203d3591-e38c-4bb1-81ae-1f0a2a797026"...},
    {"id": "fc6bc503-e51a-4dc4-814b-5b8a3f462c83"...},
    {"id": "ac992beb-6f14-4706-b3d4-20ae4137af83"...},
    {"id": "65708cc2-1fee-49b5-9a11-1f6ea347b5aa"...}
  ],
  "depositMappings": [
    {
      "depositId": "d1c3ba12-bee5-4e7f-af71-7156cbf6dec5",
      "gtins": [
        "5060608742332", "5060608742936", "5060608742967", "54493896", "3057640541483", "9008700201643", "5060608742905"
      ]
    },
    {
      "depositId": "ac992beb-6f14-4706-b3d4-20ae4137af83",
      "gtins": [
        "4104060029806", "28427834"
      ]
    },
    {
      "depositId": "5ee443fb-c28f-4052-a3ca-35ee34d48695",
      "gtins": [
        "4101120006685", "4069800005857", "41057759"
      ]
    }
  ]
}
```


REWE digital

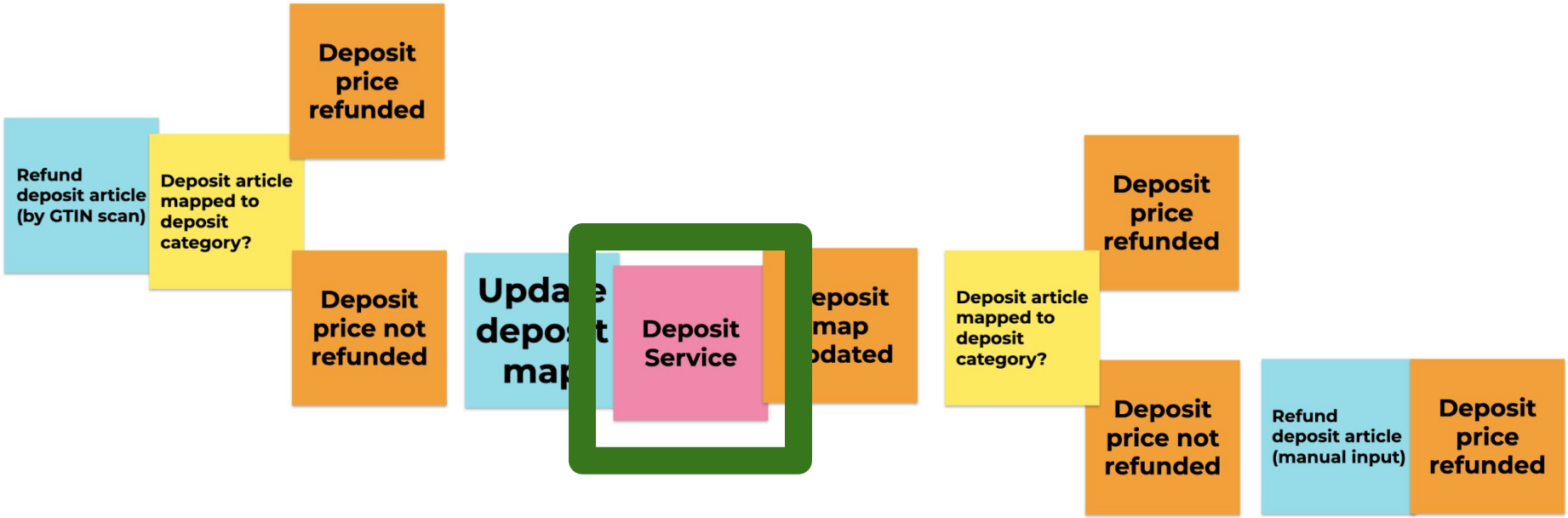

Refund deposit article (by GTIN scan)
Deposit article mapped to deposit category?
Deposit price refunded
Deposit price not refunded
Update deposit map
Deposit Service
Deposit map updated
Deposit article mapped to deposit category?
Deposit price refunded
Deposit price not refunded
Refund deposit article (manual input)
Deposit price refunded

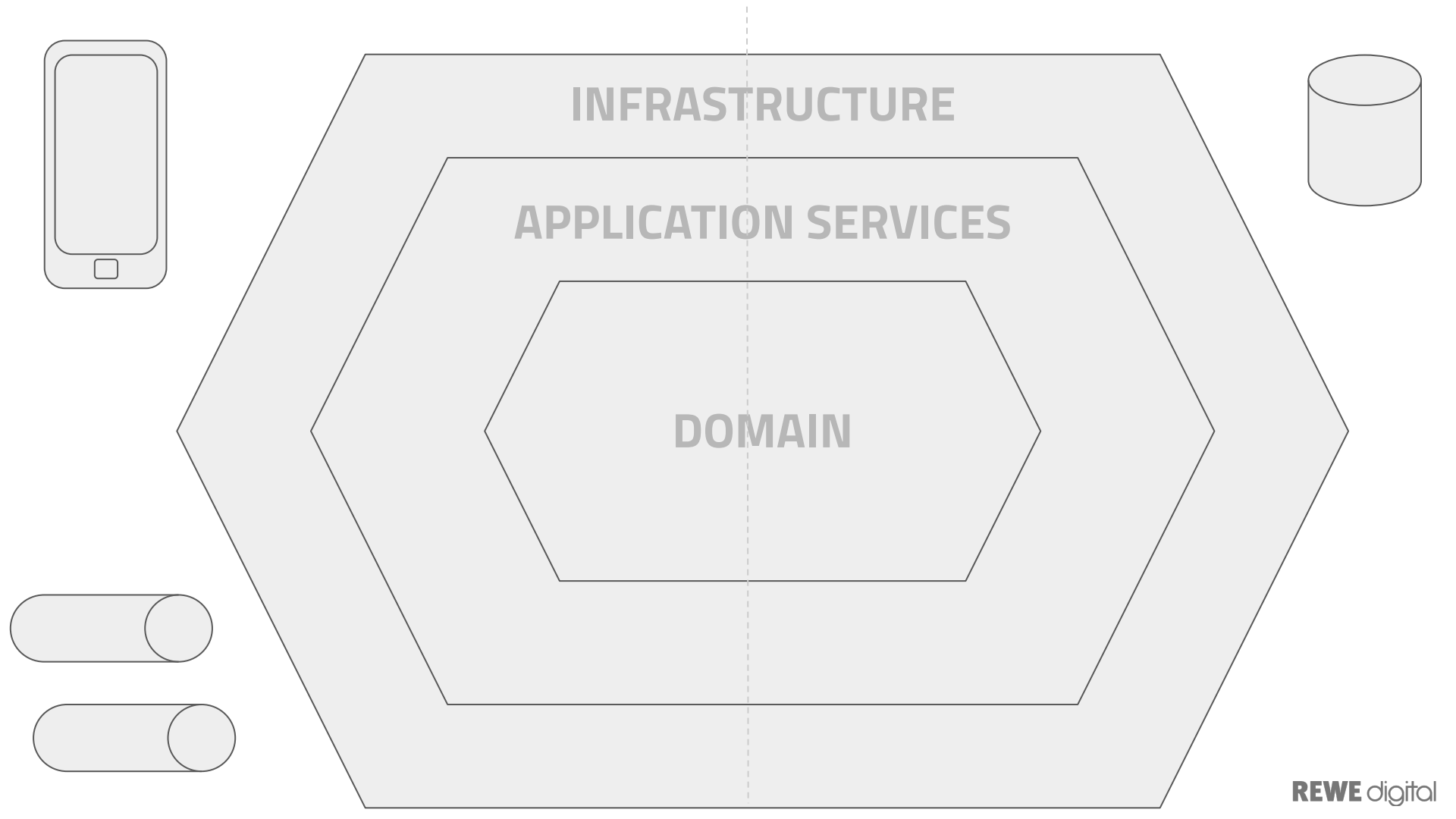
~~1. Some theory~~

2. Example

~~a. The domain~~

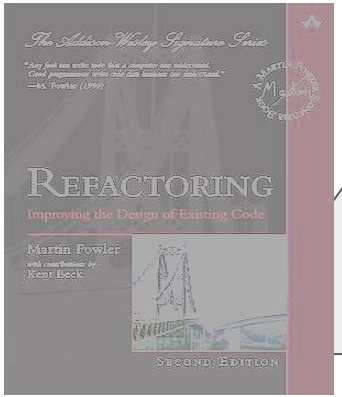
b. Let’s code!

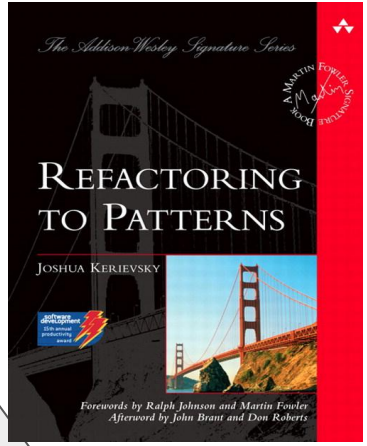

Refund deposit article (by GTIN scan)
Deposit article mapped to deposit category?
Deposit price refunded
Deposit price not refunded
Deposit Service
Deposit article mapped to deposit category?
Deposit price refunded
Deposit price not refunded
Refund deposit article (manual input)
Deposit price refunded

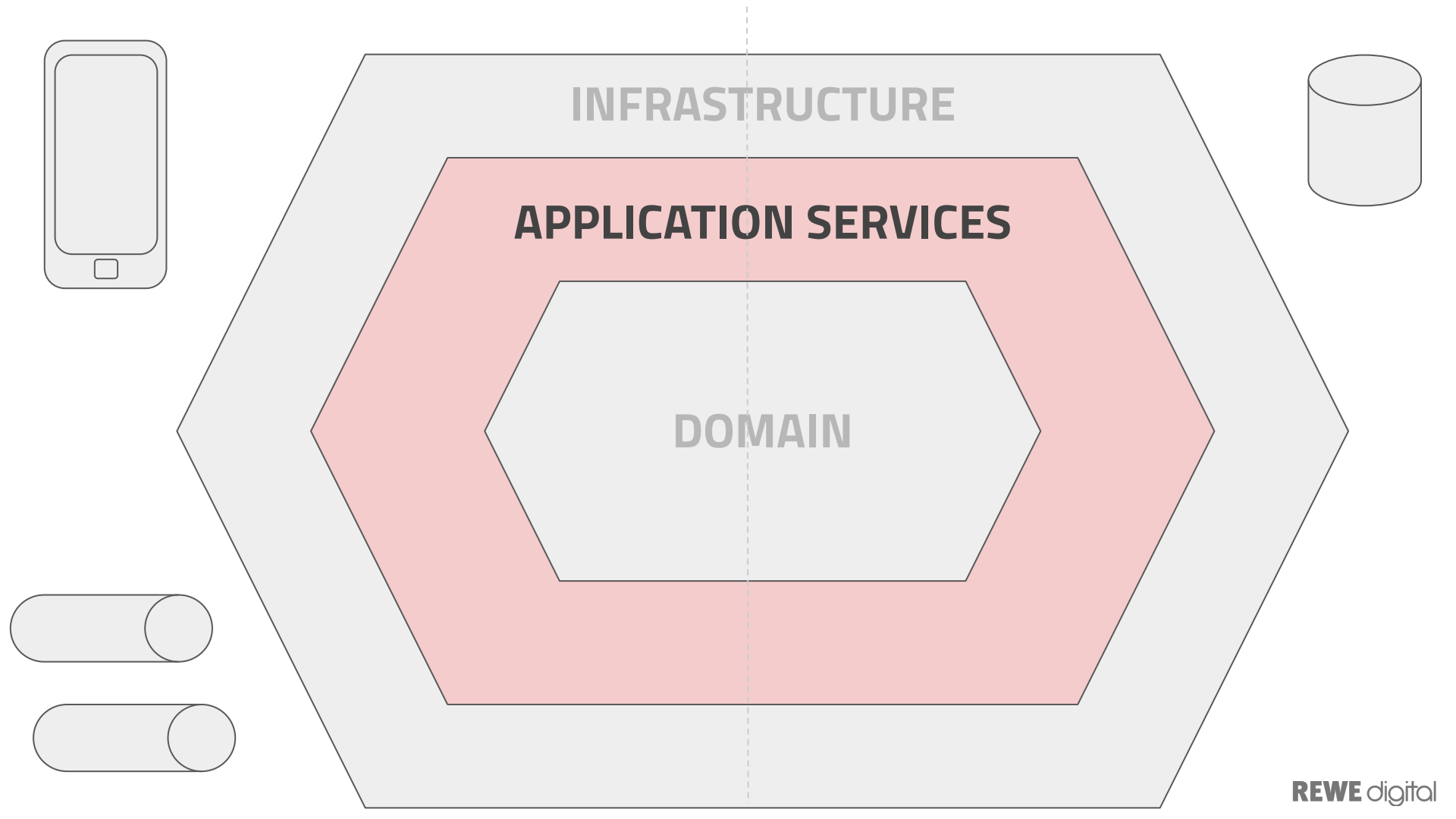

INFRASTRUCTURE
APPLICATION SERVICES
DOMAIN

Refactoring to patterns

Micro-Refactorings






INFRASTRUCTURE
APPLICATION SERVICES
DOMAIN

Creates the deposit map, based on the last update.

```
18      fun getMapping(fromTime: OffsetDateTime?): Mapping {
19          val depositCategories : List<DepositCategory> = depositCategoryRepository.findAllCategories()
20          val dataDrop : Boolean = depositCategories.any { it.createdAt >= fromTime ?: MIN_DATE_TIME }
21          val from : OffsetDateTime = if (fromTime == null || dataDrop) MIN_DATE_TIME else fromTime
22          val current : OffsetDateTime! = OffsetDateTime.now(clock)
23          val deposits : List<DepositCategoryDTO> = depositCategories.map(DepositCategory::toDTO)
24          val depositMappings : List<MappedArticleDTO> = mappedArticlesRepository
25              .getMappedArticles(from, current)
26              .groupBy(MappedArticle::depositCategoryId, MappedArticle::gtin)
27              .map { it: Map.Entry<DepositCategoryId, List<String>>
28                  MappedArticleDTO(it.key, it.value)
29              }
30
31          return Mapping(dataFrom = current, dataDrop = dataDrop, deposits = deposits, depositMappings = depositMappings)
32      }
```

Nullable?

```
18  fun getMapping(fromTime: OffsetDateTime?) : Mapping {
19      val depositCategories : List<DepositCategory> = depositCategoryRepository.findAllCategories()
20      val dataDrop : Boolean = depositCategories.any { it.createdAt >= fromTime ?: MIN_DATE_TIME }
21      val from : OffsetDateTime = if (fromTime == null || dataDrop) MIN_DATE_TIME else fromTime
22      val current : OffsetDateTime! = OffsetDateTime.now(clock)
23      val deposits : List<DepositCategoryDTO> = depositCategories.map(DepositCategory::toDTO)
24      val depositMappings : List<MappedArticleDTO> = mappedArticlesRepository
25          .getMappedArticles(from, current)
26          .groupBy(MappedArticle::depositCategoryId, MappedArticle::gtin)
27          .map { it: Map.Entry<DepositCategoryId, List<String>>
28              MappedArticleDTO(it.key, it.value)
29          }
30
31      return Mapping(dataFrom = current, dataDrop = dataDrop, deposits = deposits, depositMappings = depositMappings)
32  }
```

Nullable?

What does min date mean?

```
18  fun getMapping(fromTime: OffsetDateTime?): Mapping {
19      val depositCategories: List<DepositCategory> = depositCategoryRepository.findAllCategories()
20      val dataDrop: Boolean = depositCategories.any { it.createdAt >= fromTime ?: MIN_DATE_TIME }
21      val from: OffsetDateTime = if (fromTime == null || dataDrop) MIN_DATE_TIME else fromTime
22      val current: OffsetDateTime! = OffsetDateTime.now(clock)
23      val deposits: List<DepositCategoryDTO> = depositCategories.map(DepositCategory::toDTO)
24      val depositMappings: List<MappedArticleDTO> = mappedArticlesRepository
25          .getMappedArticles(from, current)
26          .groupBy(MappedArticle::depositCategoryId, MappedArticle::gtin)
27          .map { it: Map.Entry<DepositCategoryId, List<String>>
28              MappedArticleDTO(it.key, it.value)
29          }
30
31      return Mapping(dataFrom = current, dataDrop = dataDrop, deposits = deposits, depositMappings = depositMappings)
32  }
```

Nullable?

What does min date mean?

```
18  fun getMapping(fromTime: OffsetDateTime?): Mapping {
19      val depositCategories: List<DepositCategory> = depositCategoryRepository.findAllCategories()
20      val dataDrop: Boolean = depositCategories.any { it.createdAt >= fromTime ?: MIN_DATE_TIME }
21      val from: OffsetDateTime = if (fromTime == null || dataDrop) MIN_DATE_TIME else fromTime
22      val current: OffsetDateTime! = OffsetDateTime.now(clock)
23      val deposits: List<DepositCategoryDTO> = depositCategories.map(DepositCategory::toDTO)
24      val depositMappings: List<MappedArticleDTO> = mappedArticlesRepository
25          .getMappedArticles(from, current)
26          .groupBy(MappedArticle::depositCategoryId, MappedArticle::gtin)
27          .map { it: Map.Entry<DepositCategoryId, List<String>>
28              MappedArticleDTO(it.key, it.value)
29          }
30
31      return Mapping(dataFrom = current, dataDrop = dataDrop, deposits = deposits, depositMappings = depositMappings)
32  }
```

Another from date?

Nullable?

What does min date mean?

```
18  fun getMapping(fromTime: OffsetDateTime?) : Mapping {
19      val depositCategories : List<DepositCategory>  = depositCategoryRepository.findAllCategories()
20      val dataDrop : Boolean  = depositCategories.any { it.createdAt >= fromTime ?: MIN_DATE_TIME }
21      val from : OffsetDateTime  = if (fromTime == null || dataDrop) MIN_DATE_TIME else fromTime
22      val current : OffsetDateTime!  = OffsetDateTime.now(clock)
23      val deposits : List<DepositCategoryDTO>  = depositCategories.map(DepositCategory::toDTO)
24      val depositMappings : List<MappedArticleDTO>  = mappedArticlesRepository
25          .getMappedArticles(from, current)
26          .groupBy(MappedArticle::depositCategoryId, MappedArticle::gtin)
27          .map { it: Map.Entry<DepositCategoryId, List<String>>
28              MappedArticleDTO(it.key, it.value)
29          }
30
31      return Mapping(dataFrom = current, dataDrop = dataDrop, deposits = deposits, depositMappings = depositMappings)
32  }
```

Another from date?

-> Nullable input is leaking in from infrastructure

Why should a service assemble parts of the Map?

```
18    fun getMapping(fromTime: OffsetDateTime?): Mapping {
19        val depositCategories : List<DepositCategory>  = depositCategoryRepository.findAllCategories()
20        val dataDrop  Boolean  = depositCategories.any { it.createdAt >= fromTime ?: MIN_DATE_TIME }
21        val from : OffsetDateTime  = if (fromTime == null || dataDrop) MIN_DATE_TIME else fromTime
22        val current : OffsetDateTime!  = OffsetDateTime.now(clock)
23        val deposits  List<DepositCategoryDTO>  = depositCategories.map(DepositCategory::toDTO)
24        val depositMappings : List<MappedArticleDTO>  = mappedArticlesRepository
25            .getMappedArticles(from, current)
26            .groupBy(MappedArticle::depositCategoryId, MappedArticle::gtin)
27            .map { it: Map.Entry<DepositCategoryId, List<String>>
28                MappedArticleDTO(it.key, it.value)
29            }
30
31        return Mapping(dataFrom = current, dataDrop = dataDrop, deposits = deposits, depositMappings = depositMappings)
32    }
```

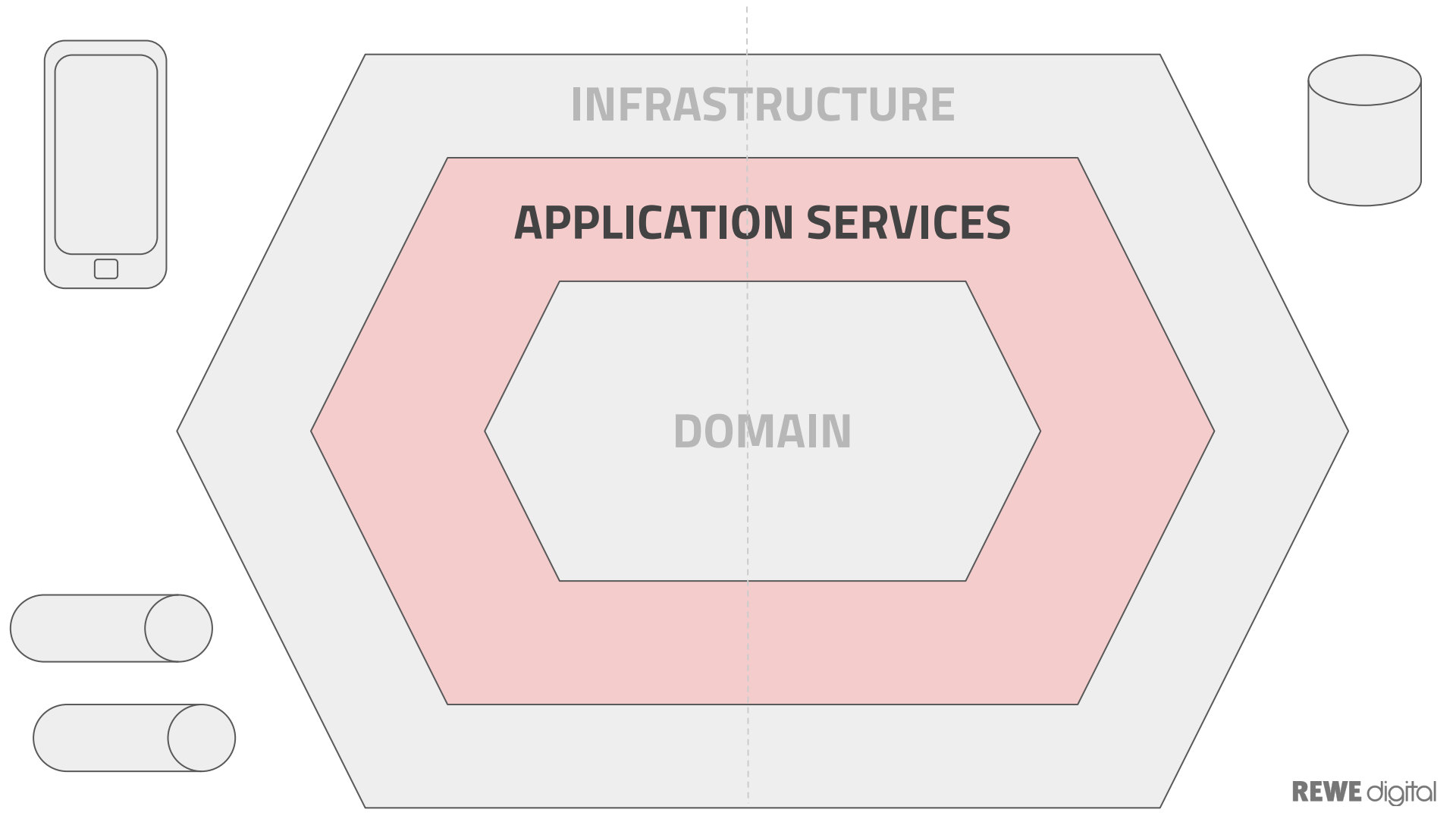
I need to think about this...

Why should a service assemble parts of the Map?

```
18    fun getMapping(fromTime: OffsetDateTime?): Mapping {
19        val depositCategories : List<DepositCategory> = depositCategoryRepository.findAllCategories()
20        val dataDrop : Boolean = depositCategories.any { it.createdAt >= fromTime ?: MIN_DATE_TIME }
21        val from : OffsetDateTime = if (fromTime == null || dataDrop) MIN_DATE_TIME else fromTime
22        val current : OffsetDateTime! = OffsetDateTime.now(clock)
23        val deposits : List<DepositCategoryDTO> = depositCategories.map(DepositCategory::toDTO)
24        val depositMappings : List<MappedArticleDTO> = mappedArticlesRepository
25            .getMappedArticles(from, current)
26            .groupBy(MappedArticle::depositCategoryId, MappedArticle::gtin)
27            .map { it: Map.Entry<DepositCategoryId, List<String>>
28                MappedArticleDTO(it.key, it.value)
29            }
30
31        return Mapping(dataFrom = current, dataDrop = dataDrop, deposits = deposits, depositMappings = depositMappings)
32    }
```

I need to think about this...

-> The deposit map should be build in the domain

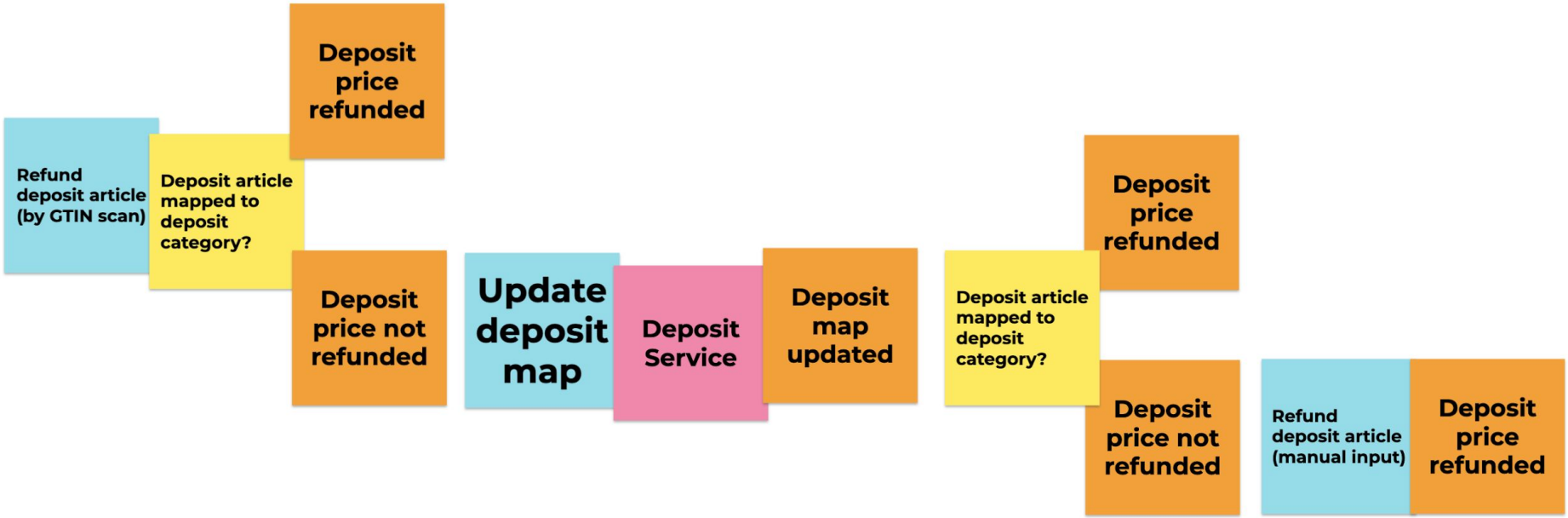

INFRASTRUCTURE
APPLICATION SERVICES
DOMAIN

The boundaries are not enforced.

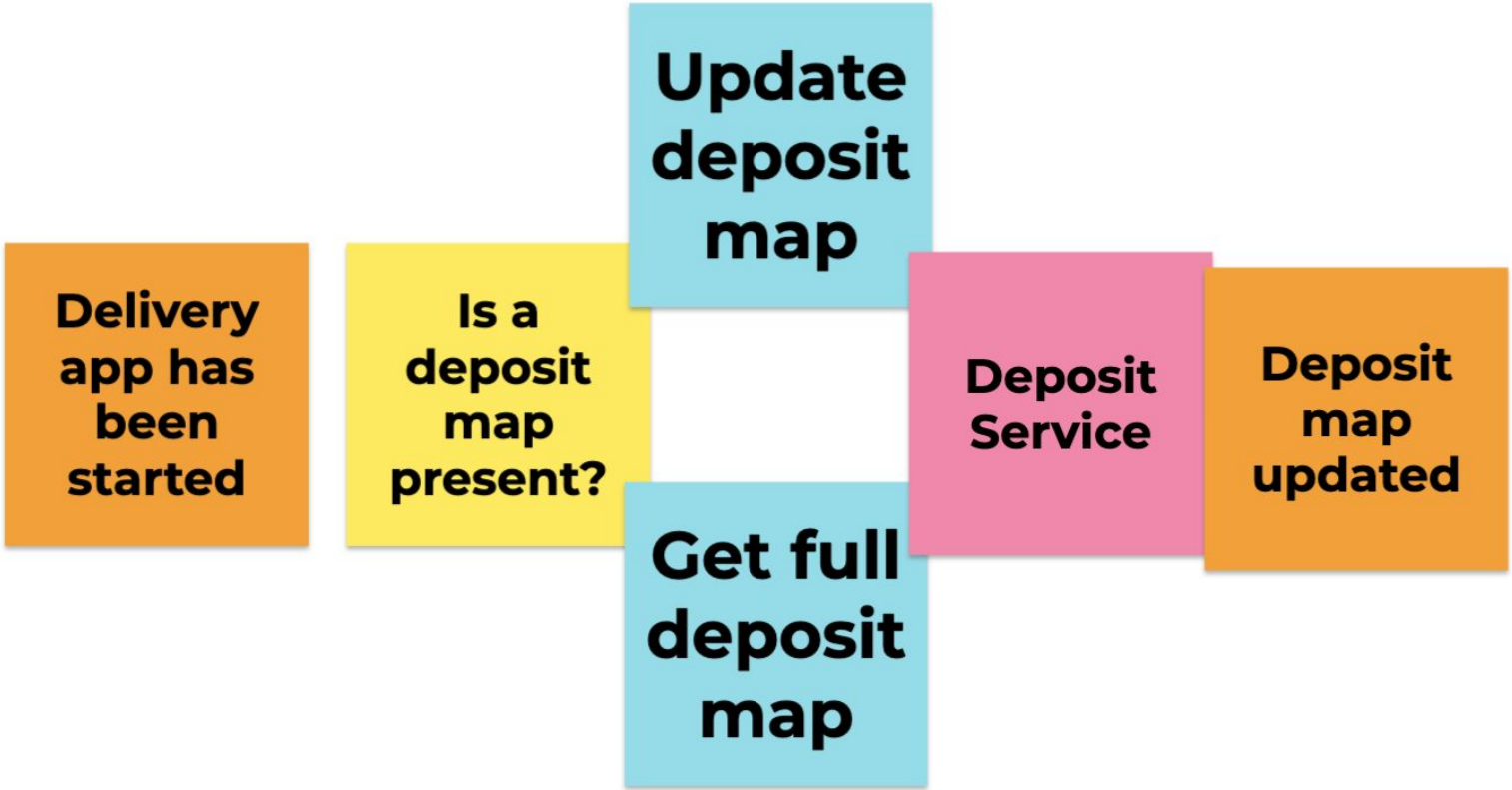
```
18    fun getMapping(fromTime: OffsetDateTime?): Mapping {
19        val depositCategories : List<DepositCategory>  = depositCategoryRepository.findAllCategories()
20        val dataDrop : Boolean  = depositCategories.any { it.createdAt >= fromTime ?: MIN_DATE_TIME }
21        val from : OffsetDateTime  = if (fromTime == null || dataDrop) MIN_DATE_TIME else fromTime
22        val current : OffsetDateTime!  = OffsetDateTime.now(clock)
23        val deposits : List<DepositCategoryDTO>  = depositCategories.map(DepositCategory::toDTO)
24        val depositMappings : List<MappedArticleDTO>  = mappedArticlesRepository
25            .getMappedArticles(from, current)
26            .groupBy(MappedArticle::depositCategoryId, MappedArticle::gtin)
27            .map { it: Map.Entry<DepositCategoryId, List<String>>
28                MappedArticleDTO(it.key, it.value)
29            }
30
31        return Mapping(dataFrom = current, dataDrop = dataDrop, deposits = deposits, depositMappings = depositMappings)
32    }
```

....and this obscures the business logic. Is there any?

Is there anything to be learned from the model?



YES! We forgot about another use case ;)



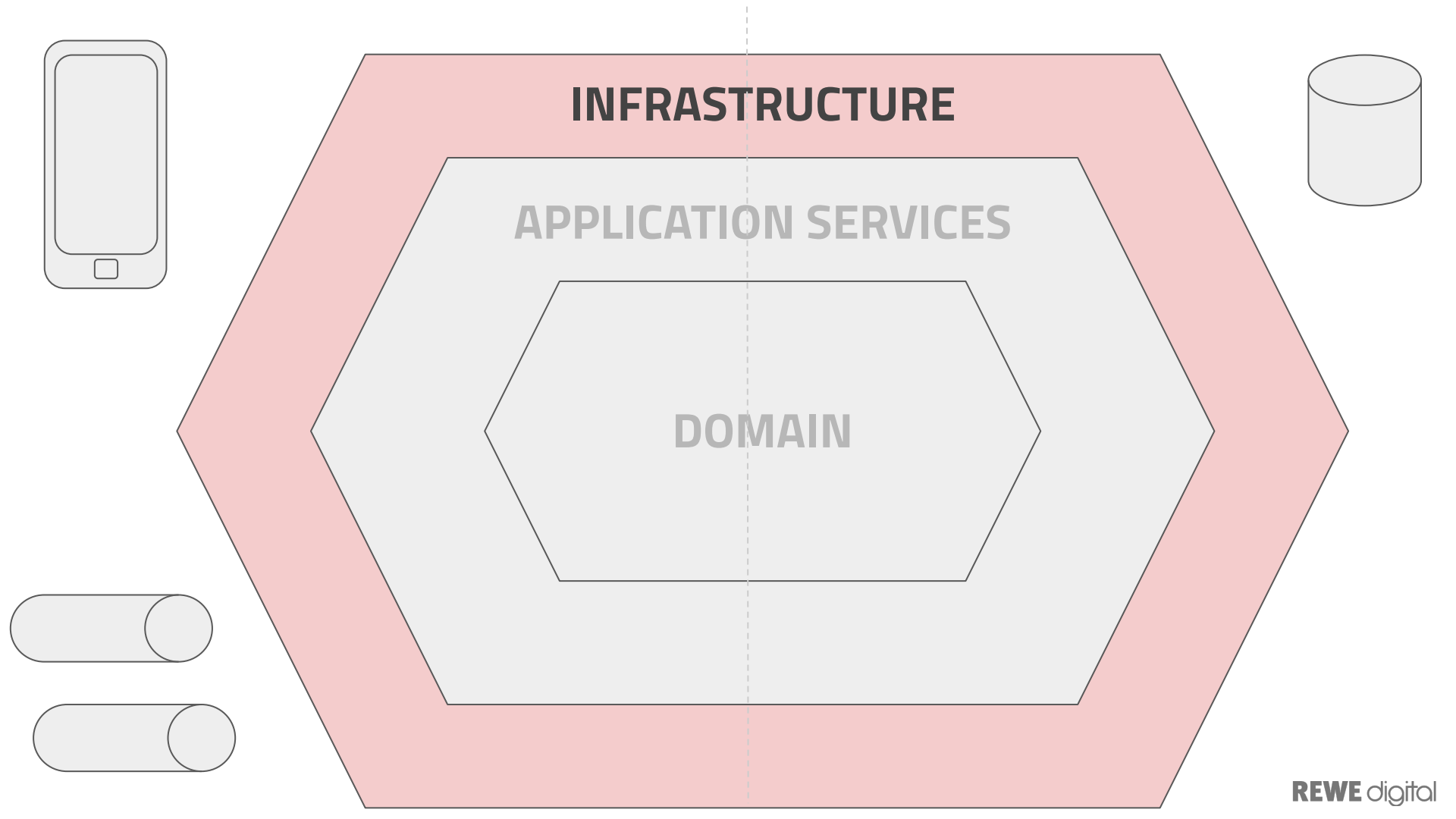
-> Let's add these concepts of the model to the code...

```
6  sealed class Query : Intent {
7      object GetDepositMap : Query()
8      data class GetDepositMapDiff(val timeOfPreviousCheck: OffsetDateTime) : Query()
9  }
```

-> Let's add these concepts of the model to the code...

```
6  sealed class Query : Intent {
7      object GetDepositMap : Query()
8      data class GetDepositMapDiff(val timeOfPreviousCheck: OffsetDateTime) : Query()
9  }
```

Input is no longer nullable!


INFRASTRUCTURE
APPLICATION SERVICES
DOMAIN

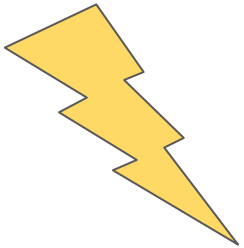
-> Null check is moved outwards to infrastructure

```
26      @GetMapping(TEMPLATE_PATH)
27      @PreAuthorize(HAS_CLIENT_ROLE_FETCH_DEPOSITS)
28      fun getMapping(@RequestParam( value: "dataFrom", required = false) @DateTimeFormat(iso = ISO.DATE_TIME) from: OffsetDateTime?) =
29          if(from == null) {
30              ok(getDepositMapService.process(GetDepositMap))
31          } else {
32              ok(getDepositMapDiffService.process(GetDepositMapDiff(from)))
33          }
```

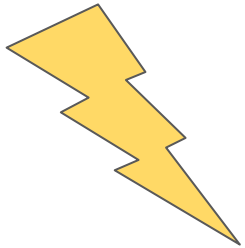
-> Null check is moved outwards to infrastructure

```
26      @GetMapping(TEMPLATE_PATH)
27      @PreAuthorize(HAS_CLIENT_ROLE_FETCH_DEPOSITS)
28      fun getMapping(@RequestParam( value: "dataFrom", required = false) @DateTimeFormat(iso = ISO.DATE_TIME) from: OffsetDateTime?) =
29          if(from == null) {
30              ok(getDepositMapService.process(GetDepositMap))
31          } else {
32              ok(getDepositMapDiffService.process(GetDepositMapDiff(from)))
33          }
```

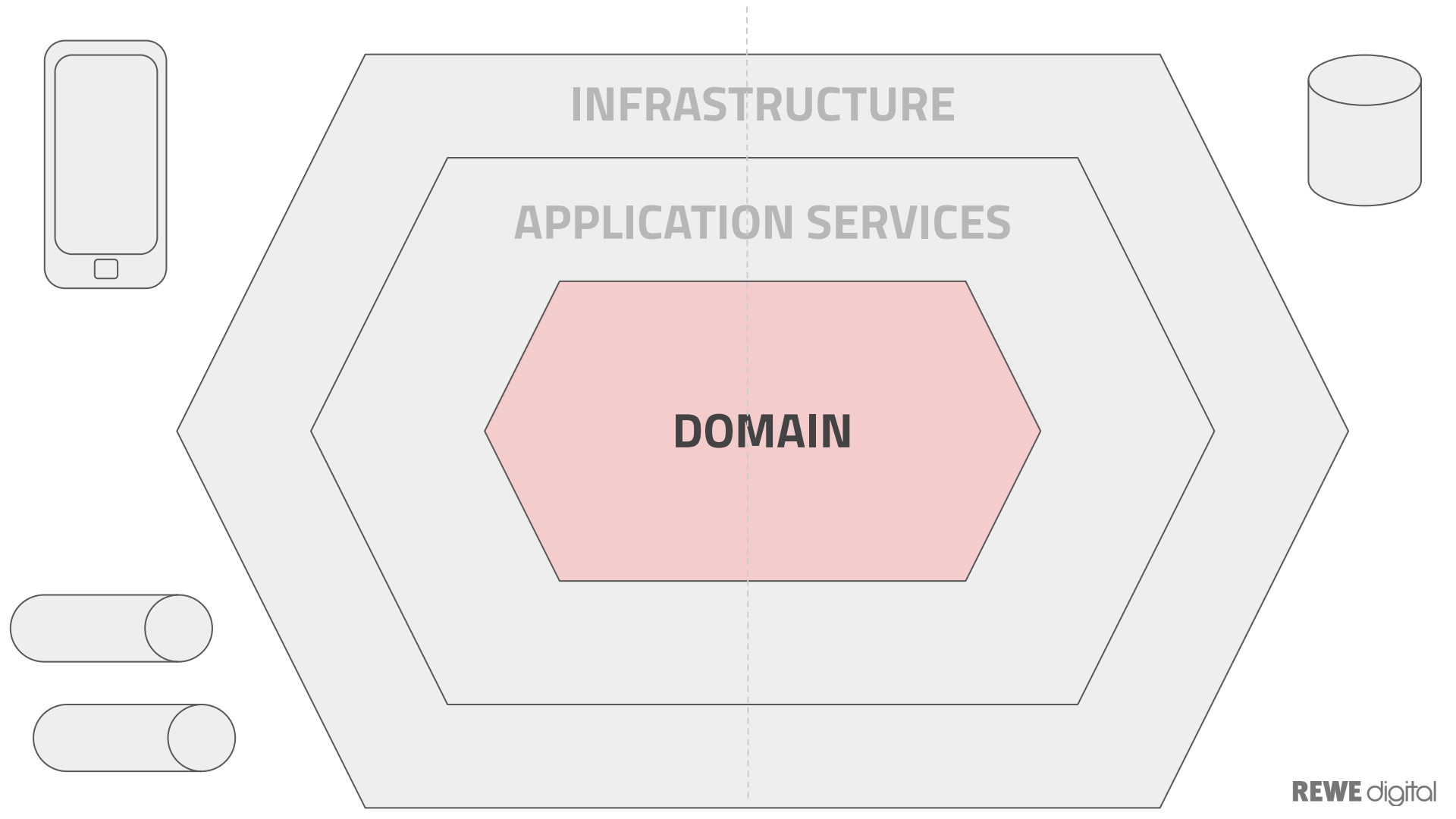
There is a service for each use case now...



Misconception!



Don’t repeat yourself!


INFRASTRUCTURE
APPLICATION SERVICES
DOMAIN

-> Let's move the deposit map creation to the domain

```
data class DepositCategoryMapping(
    val dataFrom: OffsetDateTime,
    val dataDrop: Boolean = false,
    val deposits: List<DepositCategoryDTO>,
    val depositMappings: List<MappedArticleDTO>
) {
    companion object {...}
}
```

```
15      fun createDiff(
16          now: OffsetDateTime,
17          depositCategories: List<DepositCategory>,
18          mappedArticles: List<MappedArticle>
19      ): DepositCategoryMapping = create(now, dropData: false, depositCategories, mappedArticles)
20
21      fun create(
22          now: OffsetDateTime,
23          depositCategories: List<DepositCategory>,
24          mappedArticles: List<MappedArticle>
25      ): DepositCategoryMapping = create(now, dropData: true, depositCategories, mappedArticles)
26
27      private fun create(
28          now: OffsetDateTime,
29          dropData: Boolean,
30          depositCategories: List<DepositCategory>,
31          mappedArticles: List<MappedArticle>
32      ): DepositCategoryMapping = DepositCategoryMapping(
33              dataFrom = now,
34              dataDrop = dropData,
35              deposits = depositCategories.map { DepositCategoryDTO.create(it) },
36              depositMappings = mappedArticles.let { it: List<MappedArticle>
37                  MappedArticleDTO.create(it)
38              }
39          )
```

Two different public factory methods

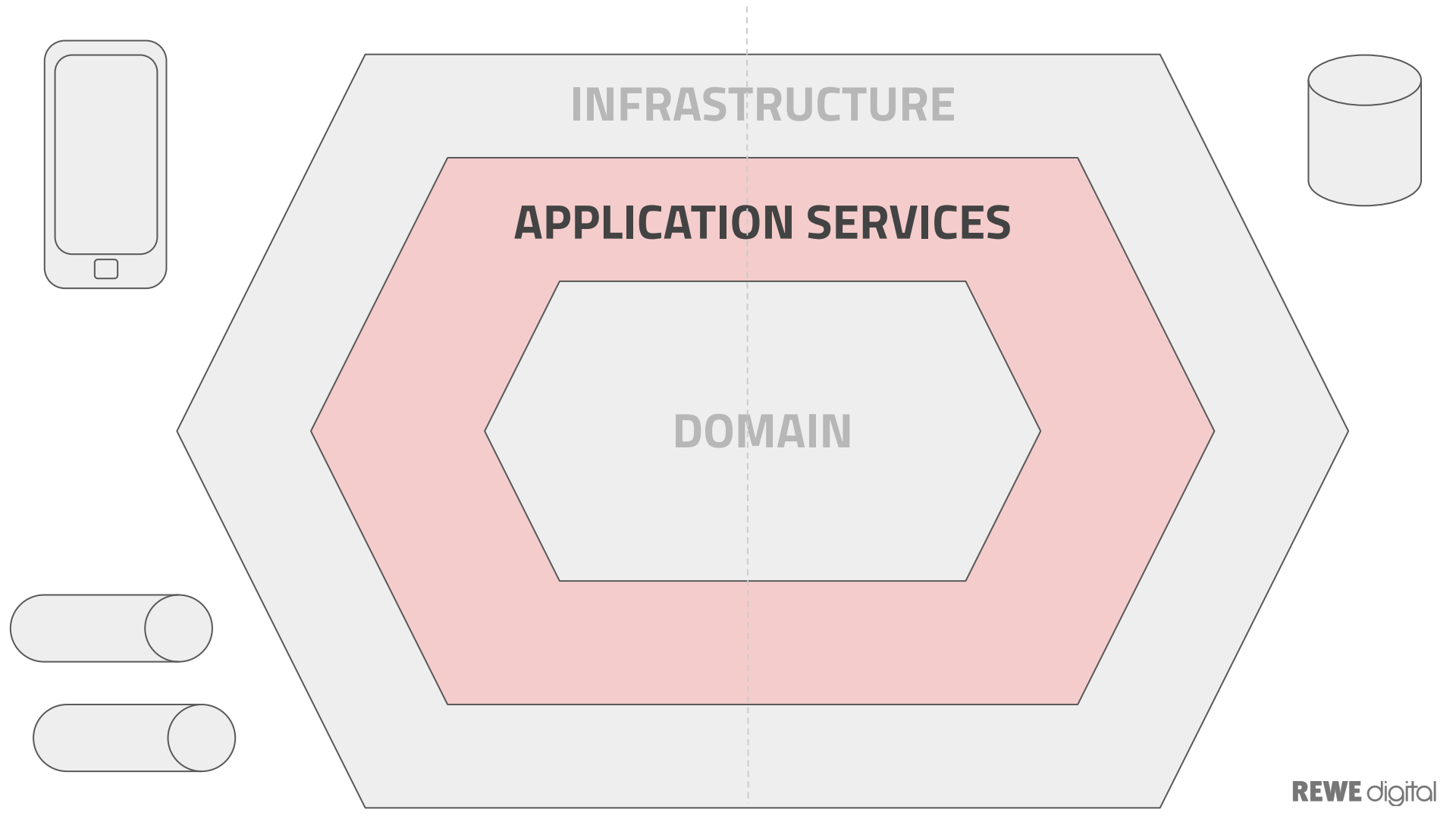
```
15      fun createDiff(
16          now: OffsetDateTime,
17          depositCategories: List<DepositCategory>,
18          mappedArticles: List<MappedArticle>
19      ): DepositCategoryMapping = create(now, dropData: false, depositCategories, mappedArticles)
20
21      fun create(
22          now: OffsetDateTime,
23          depositCategories: List<DepositCategory>,
24          mappedArticles: List<MappedArticle>
25      ): DepositCategoryMapping = create(now, dropData: true, depositCategories, mappedArticles)
26
27      private fun create(
28          now: OffsetDateTime,
29          dropData: Boolean,
30          depositCategories: List<DepositCategory>,
31          mappedArticles: List<MappedArticle>
32      ): DepositCategoryMapping = DepositCategoryMapping(
33              dataFrom = now,
34              dataDrop = dropData,
35              deposits = depositCategories.map { DepositCategoryDTO.create(it) },
36              depositMappings = mappedArticles.let { it: List<MappedArticle>
37                  MappedArticleDTO.create(it)
38              }
39      )
```

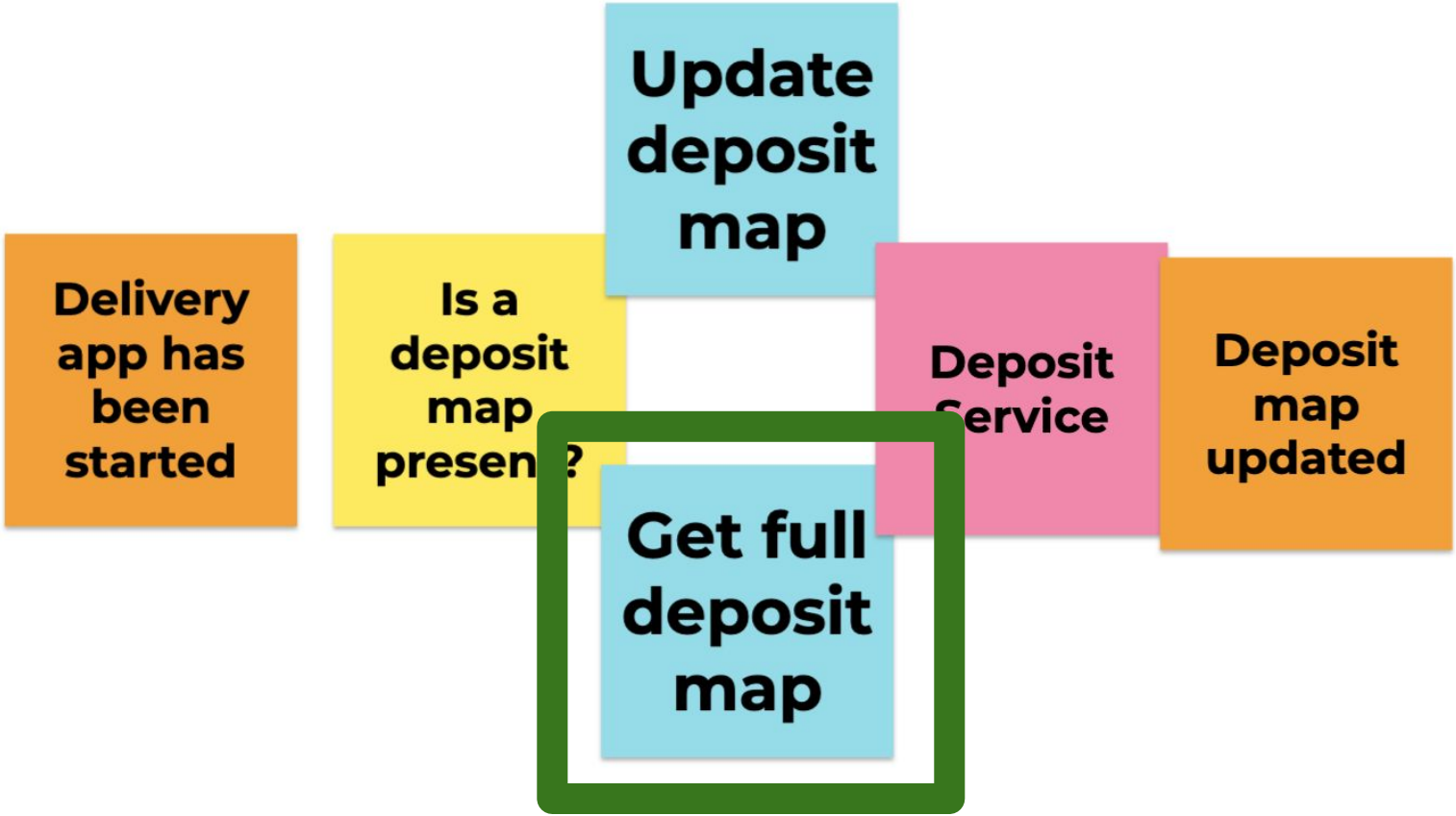
dataDrop is encapsulated in model

Naming could be improved...

```
15      fun createDiff(
16          now: OffsetDateTime,
17          depositCategories: List<DepositCategory>,
18          mappedArticles: List<MappedArticle>
19      ): DepositCategoryMapping = create(now, dropData: false, depositCategories, mappedArticles)
20
21      fun create(
22          now: OffsetDateTime,
23          depositCategories: List<DepositCategory>,
24          mappedArticles: List<MappedArticle>
25      ): DepositCategoryMapping = create(now, dropData: true, depositCategories, mappedArticles)
26
27      private fun create(
28          now: OffsetDateTime,
29          dropData: Boolean,
30          depositCategories: List<DepositCategory>,
31          mappedArticles: List<MappedArticle>
32      ): DepositCategoryMapping = DepositCategoryMapping(
33              dataFrom = now,
34              dataDrop = dropData,
35              deposits = depositCategories.map { DepositCategoryDTO.create(it) },
36              depositMappings = mappedArticles.let { it: List<MappedArticle>
37                  MappedArticleDTO.create(it)
38              }
39      )
```

Data conversion happens in the background


INFRASTRUCTURE
APPLICATION SERVICES
DOMAIN


Update deposit map
Delivery app has been started
Is a deposit map presen ?
Deposit Service
Deposit map updated
Get full deposit map

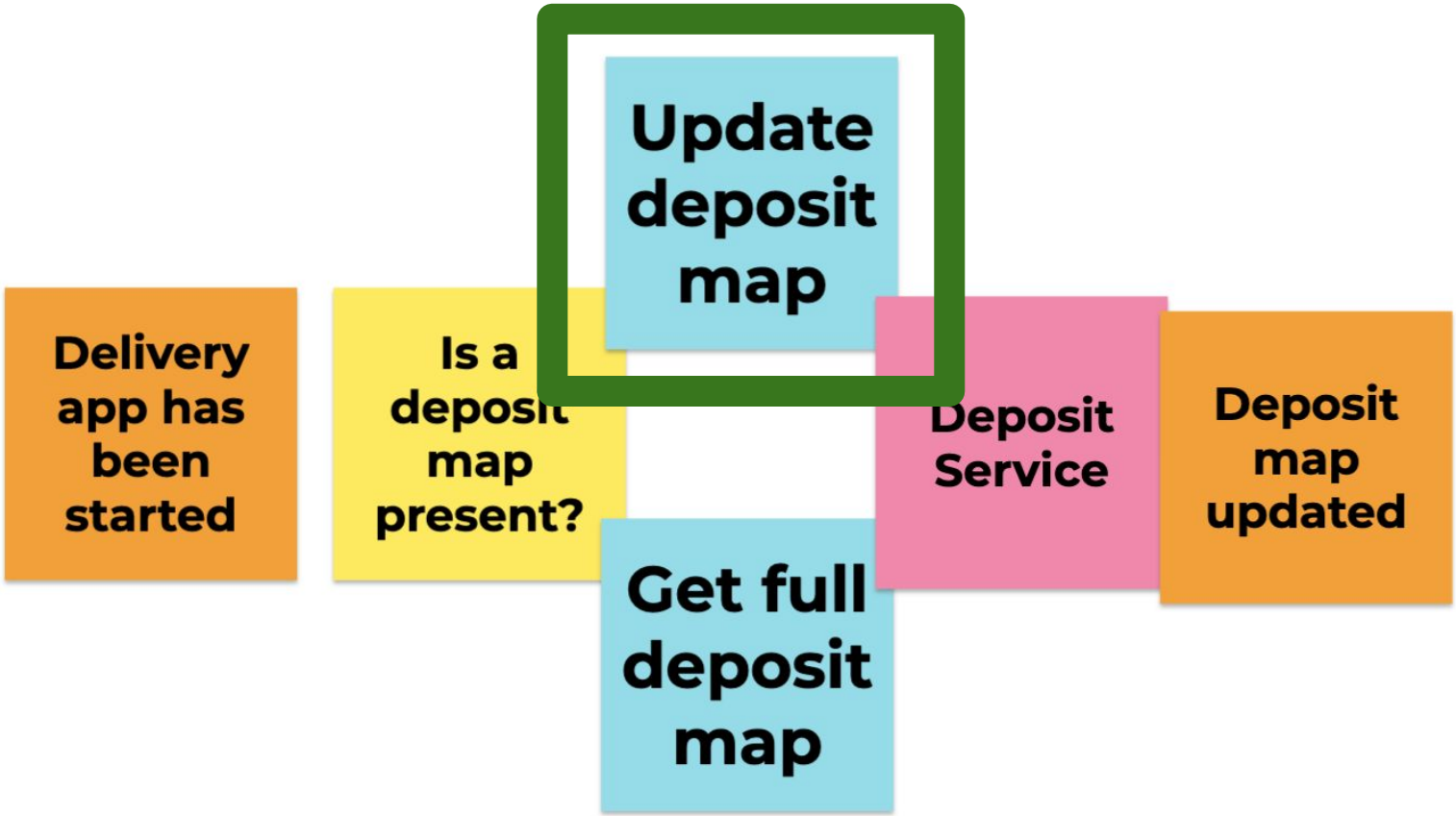
Apparently, there is no business logic...

```
19  override fun process(intent: GetDepositMap): DepositCategoryMapping = OffsetDateTime.now(clock).let { now ->
20      DepositCategoryMapping.create(
21          now,
22          depositCategoryRepository.findAllCategories(),
23          mappedArticlesRepository.getAllBefore(now)
24      )
25  }
```

Apparently, there is no business logic...

```
19  override fun process(intent: GetDepositMap): DepositCategoryMapping = OffsetDateTime.now(clock).let { now ->
20      DepositCategoryMapping.create(
21          now,
22          depositCategoryRepository.findAllCategories(),
23          mappedArticlesRepository.getAllBefore(now)
24      )
25  }
```

Added a new repository method to eliminate nullability...


Update deposit map
Delivery app has been started
Is a deposit map present?
Deposit Service
Deposit map updated
Get full deposit map

There is one business rule that was obscured before...

```
20  override fun process(intent: GetDepositMapDiff): DepositCategoryMapping = intent.run { this: GetDepositMapDiff
21      val timeOfCurrentCheck : OffsetDateTime! = OffsetDateTime.now(clock)
22      val depositCategories : List<DepositCategory> = depositCategoryRepository.findAllCategories()
23
24      if(depositCategories.haveBeenUpdatedSince(timeOfPreviousCheck)) {
25          DepositCategoryMapping.create(
26              timeOfCurrentCheck,
27              depositCategories,
28              mappedArticlesRepository.getAllBefore(timeOfCurrentCheck)
29          ) ^run
30      } else {
31          DepositCategoryMapping.createDiff(
32              timeOfCurrentCheck,
33              depositCategories,
34              mappedArticlesRepository.getAllBetween(timeOfPreviousCheck, timeOfCurrentCheck)
35          ) ^run
36      }
37  }
```

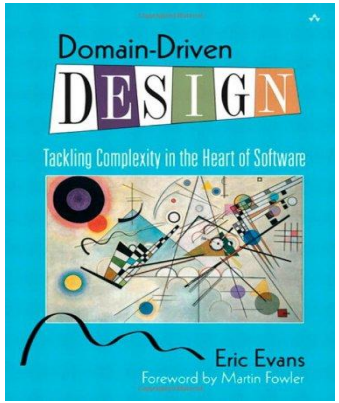
There is one business rule that was obscured before...

```
20  override fun process(intent: GetDepositMapDiff): DepositCategoryMapping = intent.run { this: GetDepositMapDiff
21      val timeOfCurrentCheck : OffsetDateTime! = OffsetDateTime.now(clock)
22      val depositCategories : List<DepositCategory> = depositCategoryRepository.findAllCategories()
23
24      if(depositCategories.haveBeenUpdatedSince(timeOfPreviousCheck)) {
25          DepositCategoryMapping.create(
26              timeOfCurrentCheck,
27              depositCategories,
28              mappedArticlesRepository.getAllBefore(timeOfCurrentCheck)
29          ) ^run
30      } else {
31          DepositCategoryMapping.createDiff(
32              timeOfCurrentCheck,
33              depositCategories,
34              mappedArticlesRepository.getAllBetween(timeOfPreviousCheck, timeOfCurrentCheck)
35          ) ^run
36      }
37  }
```

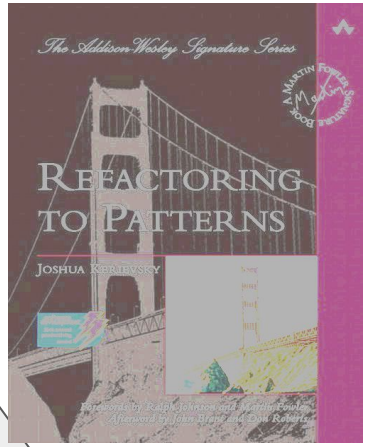
OK! We send a full ma
if the categories have
been updated :)

```
20  override fun process(intent: GetDepositMapDiff): DepositCategoryMapping = intent.run { this: GetDepositMapDiff
21      val timeOfCurrentCheck : OffsetDateTime! = OffsetDateTime.now(clock)
22      val depositCategories : List<DepositCategory> = depositCategoryRepository.findAllCategories()
23
24      if(depositCategories.haveBeenUpdatedSince(timeOfPreviousCheck)) {
25          DepositCategoryMapping.create(
26              timeOfCurrentCheck,
27              depositCategories,
28              mappedArticlesRepository getAllBefore(timeOfCurrentCheck)
29          ) ^run
30      } else {
31          DepositCategoryMapping.createDiff(
32              timeOfCurrentCheck,
33              depositCategories,
34              mappedArticlesRepository getAllBetween(timeOfPreviousCheck, timeOfCurrentCheck)
35          ) ^run
36      }
37  }
```

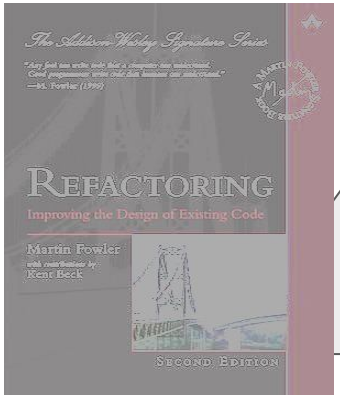
Parameters are not null. Min Date has been eliminated.



Refactoring to deeper insight



Refactoring to patterns



Micro-Refactorings

~~1. Some theory~~

~~2. Example~~

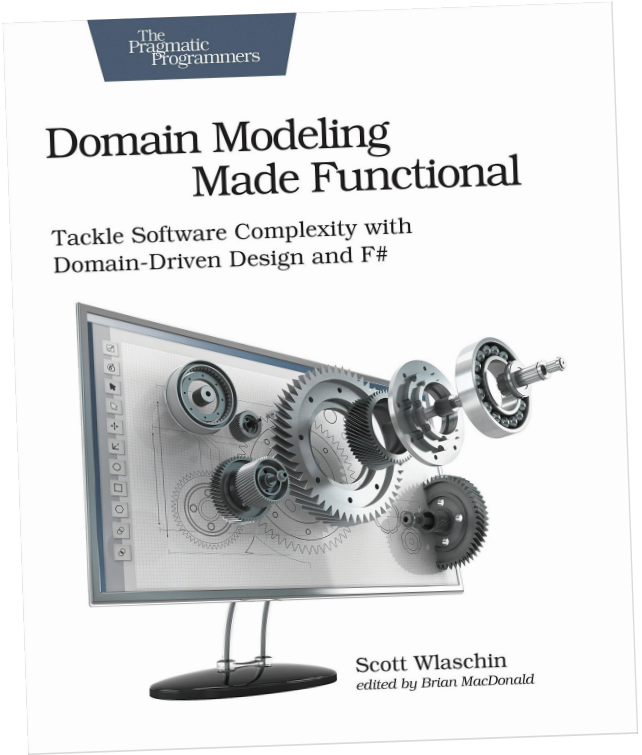
~~a. The domain~~

~~b. Let's code!~~

Afterthoughts

- It might not be the best example
- We were merely reminded about domain concepts we didn’t translate to code
- It was not a **breakthrough**
- But the code is now aligned again with the domain model! Hexagon ftw!

Recommendations



Let’s learn more about tactical design. Contact me!

Questions?

Christoph Baudson / @sustainablepace