

# Domain Driven Design

Mikael Sundberg  
Pingpong  
@msundb

# Why do we use DDD?

- Have a clear language.
- To highlight uncertainties.
- Challenge ourselves.

# Example

What makes a Course?

- Material to learn
- Questions to verify you understand the material
- Students

# Example

What makes a Course?

- Material to learn.
- ~~Questions to verify you understand the material~~ - Maybe, the evaluation probably not.
- ~~Students~~ - No, lets call it a Participation.

Can we Rename Course? Can we be more precise?

## Pågående




Butikssäljare på ...  
Som butikssäljare på Teknikm...

---



Fotboll utan fylla  
Alkoholprevension på idrottsa...

---




Internkontroll för ...  
Internkontroll och styrning är ...

---

Name and  
description

Progress

## Alla kurser och program



Odla i pallkrage  
Lär dig smarta sätt att få stor skör...

STARTA KURSEN



Basic Principles of ...  
Baloo Learning offers an introducti...

STARTA KURSEN



Grunderna i GDPR  
Baloo Learning bjuder på en kurs i ...

STARTA KURSEN

Is the course  
started?

# Course catalog

- Course participation
- Course progress
- Course information
- etc...

That sounds like many domains?

What domain is this? And in what context?

## Pågående



Butikssäljare på ...  
Som butikssäljare på Teknikm...



Fotboll utan fylla  
Alkoholprevention på idrottsa...



Internkontroll för ...  
Internkontroll och styrning är ...

## Alla kurser och program



Odla i pallkrage  
Lär dig smarta sätt att få stor skör...

STARTA KURSEN



Basic Principles of ...  
Baloo Learning offers an introducti...

STARTA KURSEN

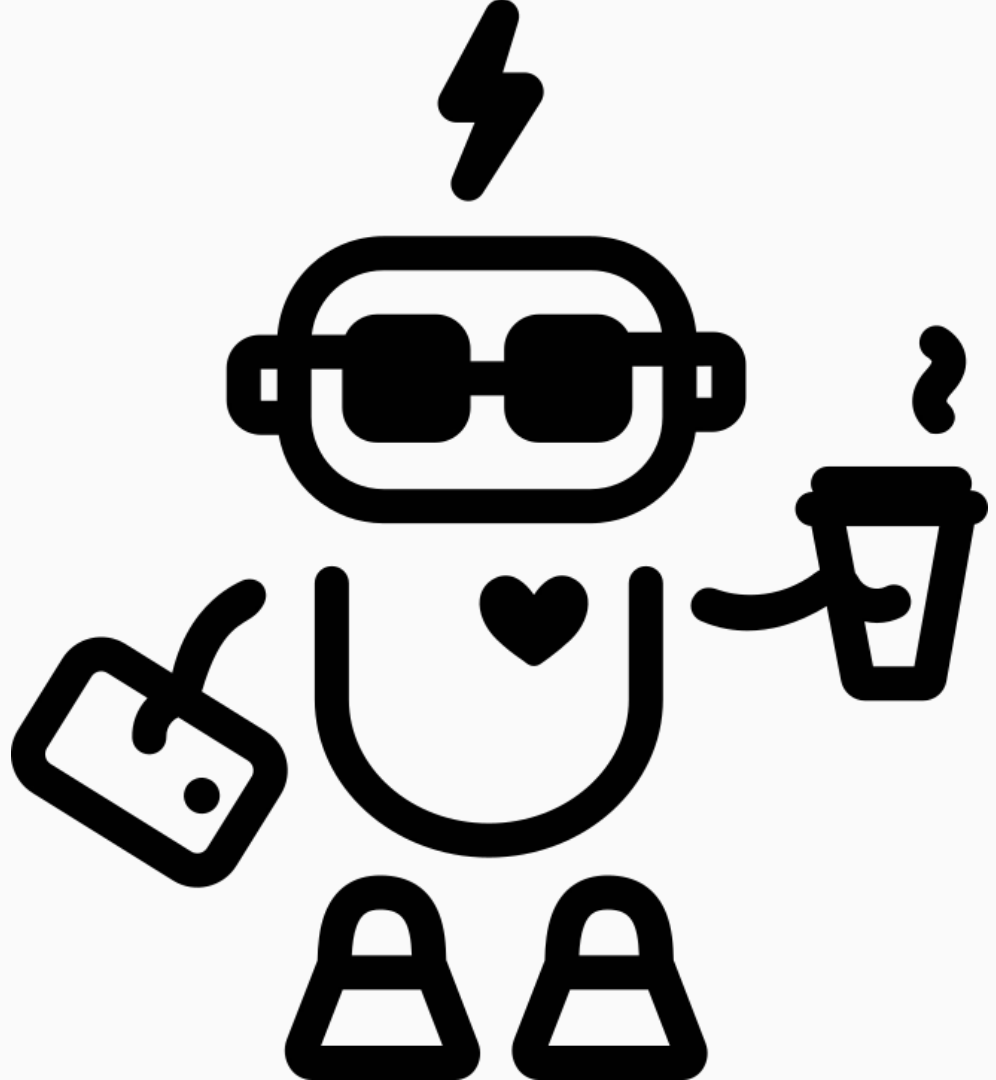


Grunderna i GDPR  
Baloo Learning bjuder på en kurs i ...

STARTA KURSEN

Baloo

One app/website  
many domains





## Welcome to Baloo Learning



Create



Learn



Analyse



Achievements



Plan

## Admin



App Access



Users & Groups



Configure



Each domain/context is a service. (not all services is a domain or context though)

## Services

Name

- ✓ [configure-app-api-service](#)
- ✓ [configure-app-ui-service](#)
- ✓ [course-designer-service](#)
- ✓ [course-evaluation-service](#)
- ✓ [course-info-service](#)
- ✓ [course-navigation-service](#)
- ✓ [course-participation-service](#)
- ✓ [course-progress-service](#)
- ✓ [course-report-catalog-service](#)
- ✓ [create-app-api-service](#)

```

▼ C Course
  ▶ E Status
    m Course()
    m Course(String, String, String, Long, String, String)
    m id(): String
    m name(): String
    m description(): String
    m lastUpdatedBy(): String
    m lastUpdatedAt(): Long
    m status(): Status
    m backgroundImageUrl(): String
    m addPart(Part): void
    m replaceParts(List<Part>): void
    m getParts(): List<Part>
    m updateLastUpdatedBy(String): void
    m updateTimestamp(Long): void
    m updateDescription(String): void
    m updateName(String): void
    m updateBackgroundImageUrl(String): void
    m publish(): void
    m createCopy(String): Course
    m addProgram(String): void
    m removeProgram(String): void
    m published(): boolean
    m markDeleted(): void
    m canBeDeletedInfo(): CanCourseBeDeletedInfo
    m deleted(): boolean

```

```

▼ C CourseAnswers
  m CourseAnswers()
  m CourseAnswers(String, String, List<PartAnswers>)
  m courseId(): String
  m partAnswers(): Collection<PartAnswers>
  m evaluateParticipation(CompletedParticipation): CourseParticipationResu
  m getPartAnswers(String): PartAnswers
  f courseId: String
  f name: String
  f partAnswers: Map<String, PartAnswers>

```

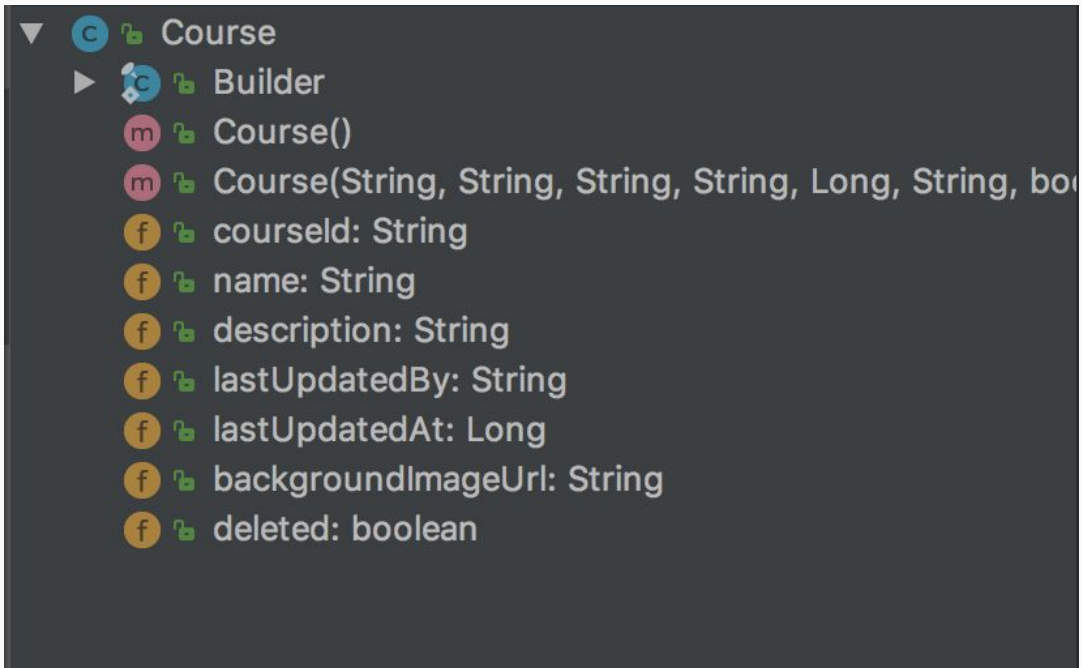
```

▼ C Course
  m Course()
  m Course(String, String)
  m assertStatusIsPublished(): void
  m publish(): Course
  m statusDraft(): boolean
  f courseId: String
  f status: String

```

# Domain objects or data bags

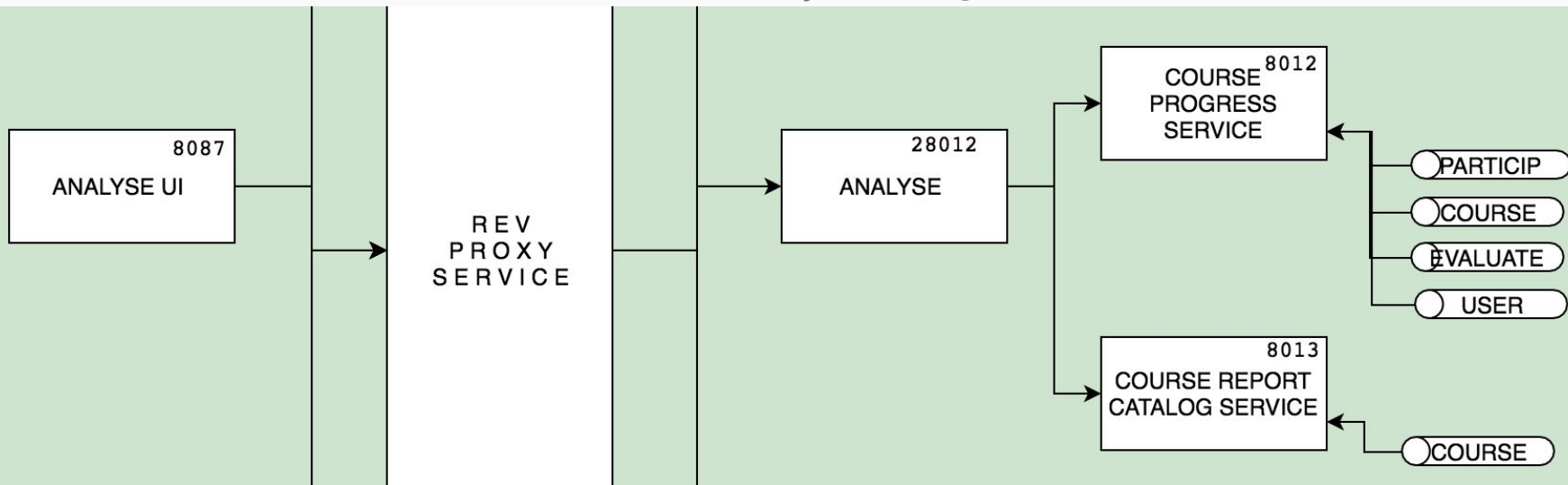
Not everything is a domain object.  
Sometimes you just have data.



Some things we live by

# Keep your own state

A service should not talk to other services. Listen to messages and keep your own state. Your Course is different from your neighbours course.



# Creating a new service

Bootstrap = do http call on spin up to fetch missing data

```
private fun spinUpService(programRepositoryFactory: RepositoryFactory<ProgramRepository>,<br>    val client = org.glassfish.jersey.client.JerseyClientBuilder.createClient()<br>    logger.info("Looking for customers")<br>    val response = client.target("http://customer-config-service/").path("customers").req<br>    if (response.statusInfo.family != Response.Status.Family.SUCCESSFUL) {<br>        throw RuntimeException("http call error")<br>    }<br>    val customers: List<String> = response.readEntity(List::class.java) as List<String><br>    logger.info("Found [{}] customers", customers.size)<br><br>    customers.forEach { customer -><br>        programRepositoryFactory.get(CustomerId(customer)).drop()<br>        val courseInfoResponse = client.target("http://program-info-service/").path("cust<br>        if (courseInfoResponse.statusInfo.family != Response.Status.Family.SUCCESSFUL) {<br>            throw RuntimeException("http call error")<br>        }<br>        val programsDto: ProgramsDto = objectMapper.readValue(courseInfoResponse.readEnti<br>        logger.info("Found [{}] programs", programsDto.programs.size)<br>        programsDto.programs.forEach { program -><br>            logger.info("Saving program [{}] in db", program.programId)<br>            programRepositoryFactory.get(CustomerId(customer)).saveOrUpdate(Program(program<br>        }<br>    }<br>}
```

- Revert "CORP-499 Spin up non deleted programs"
- CORP-499 Spin up non deleted programs

# Be extremely specific

Not being specific enough can cause your domains/bounded contexts to grow more than you want. And adds confusion.

If you add more info in service A, because downstream service B needs it, something could be fishy.

# Testing

```
▼ api
  CourseResourceTest
▼ integration
  CourseUpdateHandlerTest
```

0 tests using more than 1 service

Heavy focus on unit testing domain objects

Most logic should be in domain objects, and they are easy to test!

```
m test100PercentCorrectAnswerYieldsApprovedResult(): void
m test50PercentCorrectAnswerYieldsApprovedResult(): void
m testResponseToMissingQuestion(): void
m testIncorrectAnswerYieldsUnapprovedResult(): void
m testMultipleResponseQuestionWithAllAnswersSelectedShouldBeApproved(): void
m testMultipleResponseQuestionWithOneMissingAnswerShouldNotBeApproved(): void
m testMultipleResponseQuestionWithAllAnswersSelectedInWrongOrderShouldBeApproved(): void
m testMultipleResponseQuestionWithCorrectNumberOfAnswersButOnesWrongShouldNotBeApproved(): void
```



# Domain models can emit events

The domain object knows what happens.

```
logger.info("Creating plan for group [{}]", groupId);

Group group = repositoryFactory.get(customerId).load(groupId);
List<Event> events = group.plan(PlanRequest.toPlan(request));
repositoryFactory.get(customerId).saveOrUpdate(group);

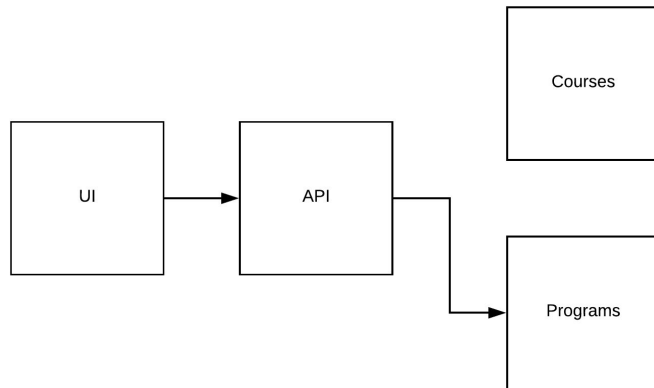
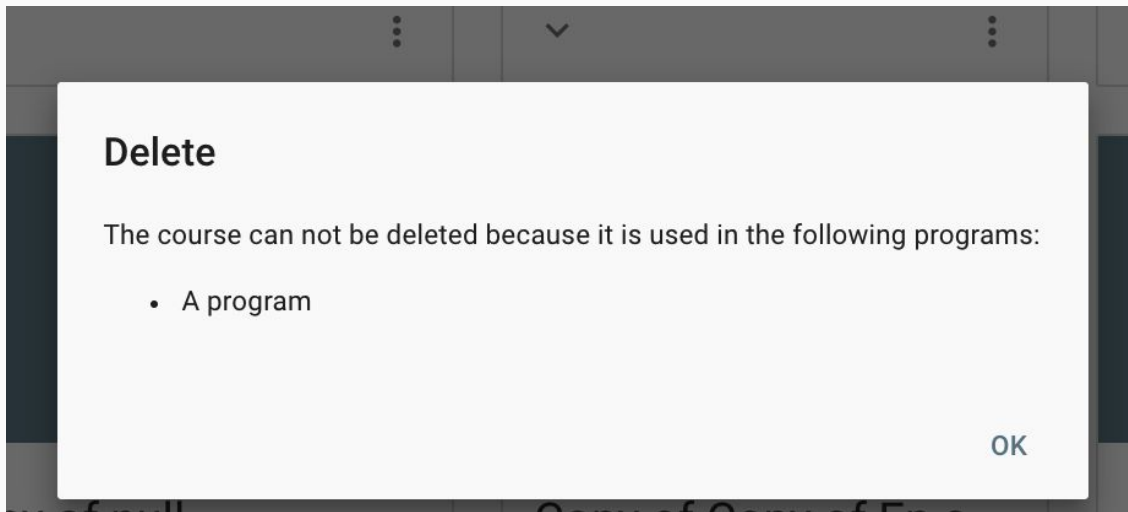
events.forEach(event -> publisher.publish(new PayloadMetaData(customerId, event.eventType), event));
```

# Examples

# Can a course be deleted?

Business rule: A course can be deleted, unless its is in a program.

We could ask program-service if it has any program with this course.

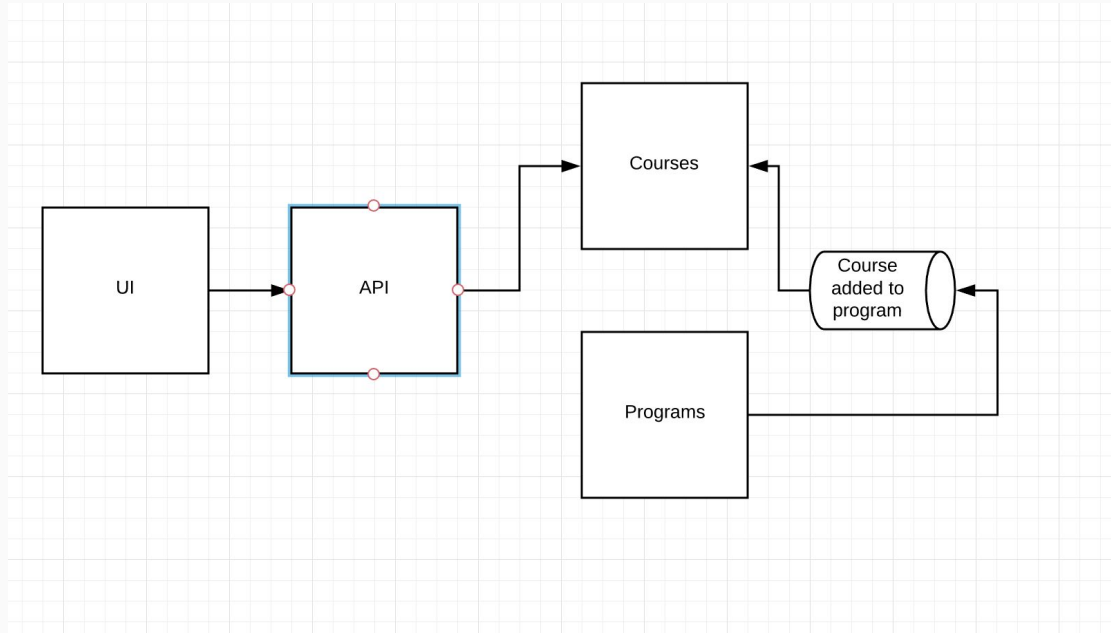
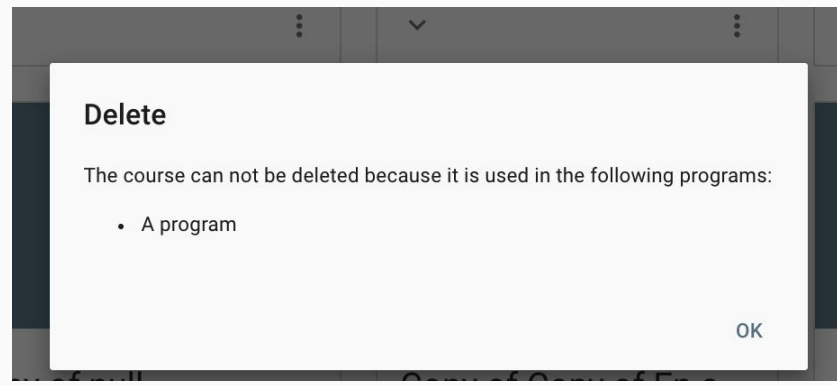


# Can a course be deleted?

Business rule: A course can be deleted, unless its is in a program.

We could ask program-service if it has any program with this course.

But, program service should not be responsible for if a course can be deleted.



# Can a course be deleted?

```
@GET
@Path("/{courseId}/can-be-deleted")
public Response canCourseBeDeleted(@PathParam("courseId") String courseId) {
    retrofit2.Response<CanCourseBeDeletedDto> canCourseBeDeletedDtoResponse = courseDesignerService.canCourseBeDeleted(
        extractCorrelationId(headers).toString(),
        extractCustomerId(headers).toString(),
        courseId)
        .blockingSingle();
}
```

```
💡 public CanCourseBeDeletedInfo canBeDeletedInfo() {
    return new CanCourseBeDeletedInfo(programs.isEmpty(), programs);
}
```

# Adding user to group

Mina kurser och program



## Grunderna i GDPR

Baloo Learning bjuder på en kurs i ...

STARTA KURSEN



## Checklistan som ...

Här har vi samlat 10 kurser dä...

VISA PROGRAM

When a user is added to a group, courses targeted to that group should be targeted to the user.

When listing courses. We could fetch the groups for the user, and then all targeted courses for those groups.

# Adding User to group

When a user is added to a group, courses targeted to that group should be targeted to the user.

```
List<Plan> plansInGroup = groupPlanRepositoryFactory.get(customerId).findPlans((event.groupId));
Coworker coworker = coworkerTargetedCoursesRepositoryFactory.get(customerId).find(event.userId).orElse(new Coworker(event.userId));

plansInGroup.forEach(plan -> {
    List<Event> events = coworker.addPlan(event.groupId, plan.planId, new Content(plan.content.contentId, plan.content.contentType));
    events.forEach(planAddedEvent -> publisher.publish(new PayloadMetaData(customerId, planAddedEvent.eventType), planAddedEvent));
});
```

# Adding User to group

When listing courses. We could fetch the groups for the user, and then all targeted courses for those groups.

```
@GET
@Path("/coworkers/{coworkerId}/planned-content")
public Response getPlannedContent(@PathParam("customerId") CustomerId customerId, @PathParam("coworkerId") String coworkerId) {
    logger.info("Getting planned-content for coworker [{}]", coworkerId);
    Optional<Coworker> coworkerTargetedCoursesOptional = coworkerRepository.get(customerId).find(coworkerId);
    ContentsDto contentsDto = ContentsDto.toDto(coworkerTargetedCoursesOptional.orElse(new Coworker(coworkerId)));
    logger.info("Returning published courses for coworker [{}]", coworkerId);
    return Response.ok(contentsDto).build();
}
```



# Conclusion

- Makes us think.
- Encourages involvement of non developers
- Improves separation between services
- Sometimes frustrating to find correct names

Thank you