

Modeling Problems in Software



Steve Smith

Force Multiplier for
Dev Teams

@ardalis | ardalisc.com

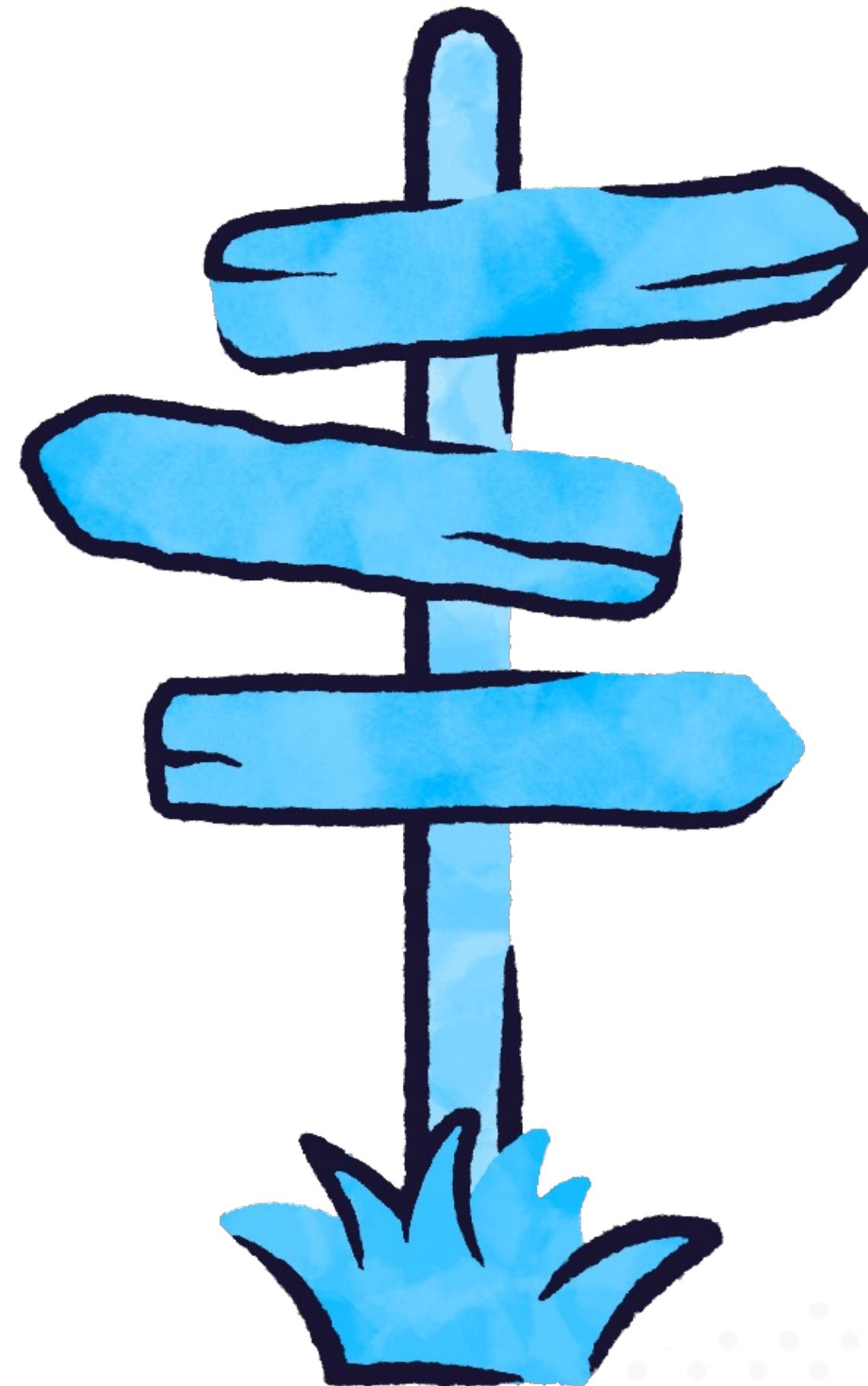


Julie Lerman

Software Coach,
DDD Champion

@julielerman | thedatafarm.com

Module Overview



Breaking up the veterinary office domain

The importance of the domain experts

A play! (Discovering the domain)

Core elements of a domain model

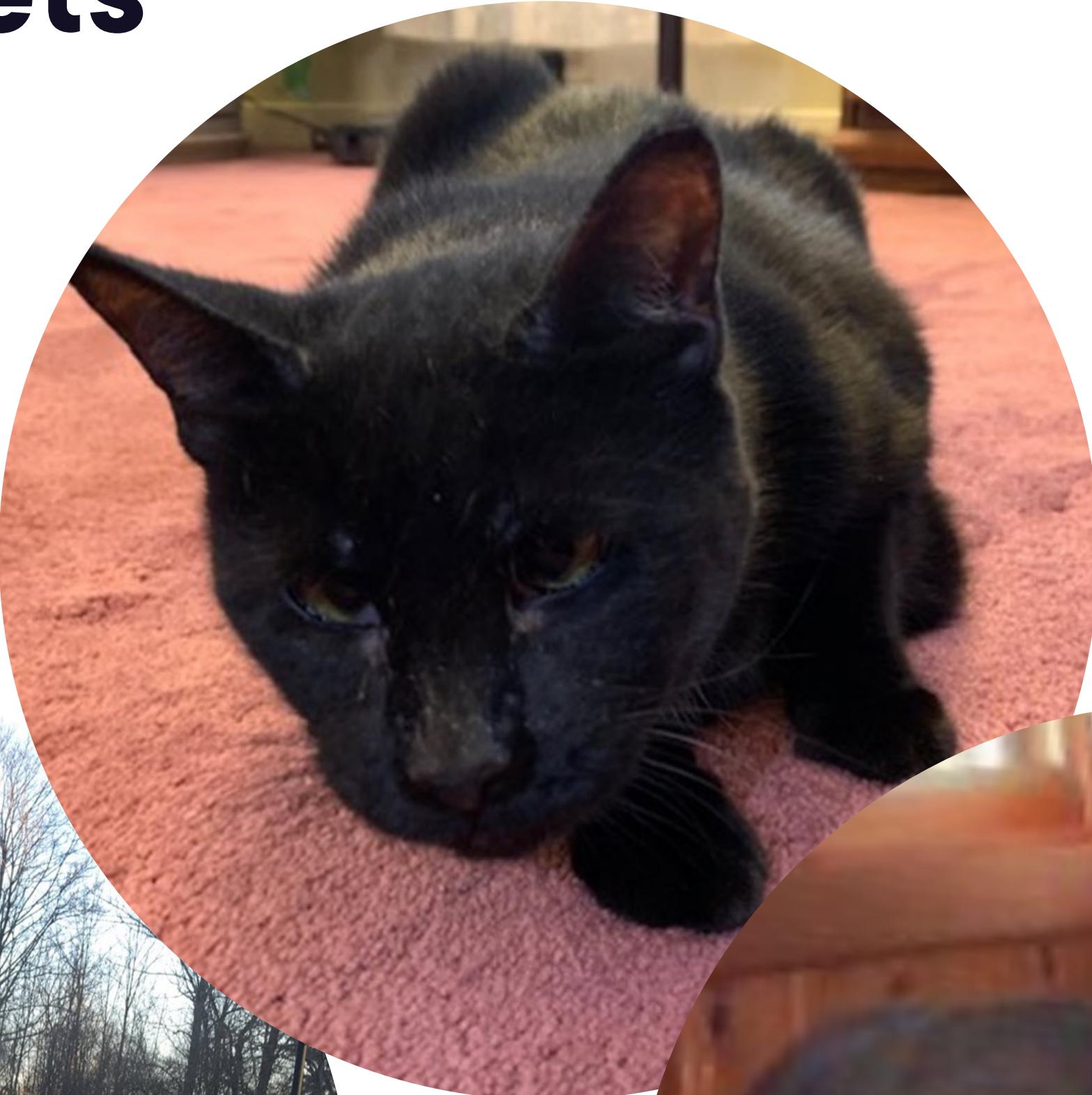
Subdomains and bounded contexts

That ubiquitous term: ubiquitous language



Introducing Our Domain

Some of Our Pets



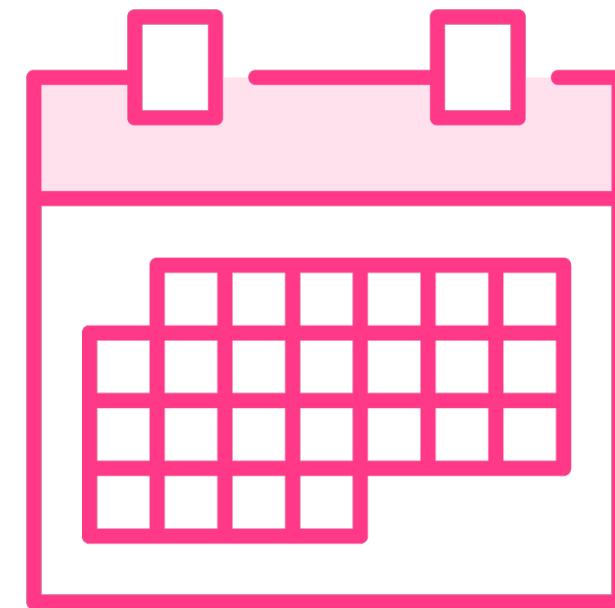
Our Domain: A Veterinary Practice





Dr. Smith

More Than Just Caring for Pets



Schedule



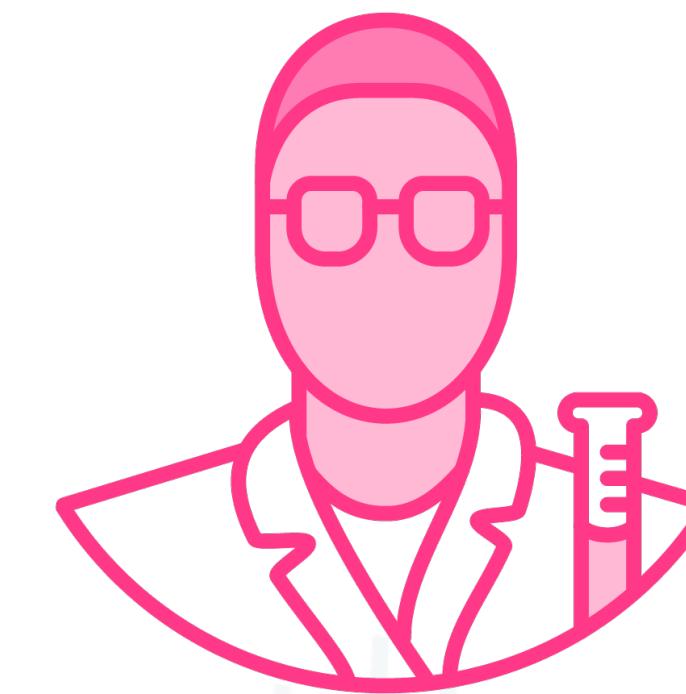
Invoices



Payments



Records

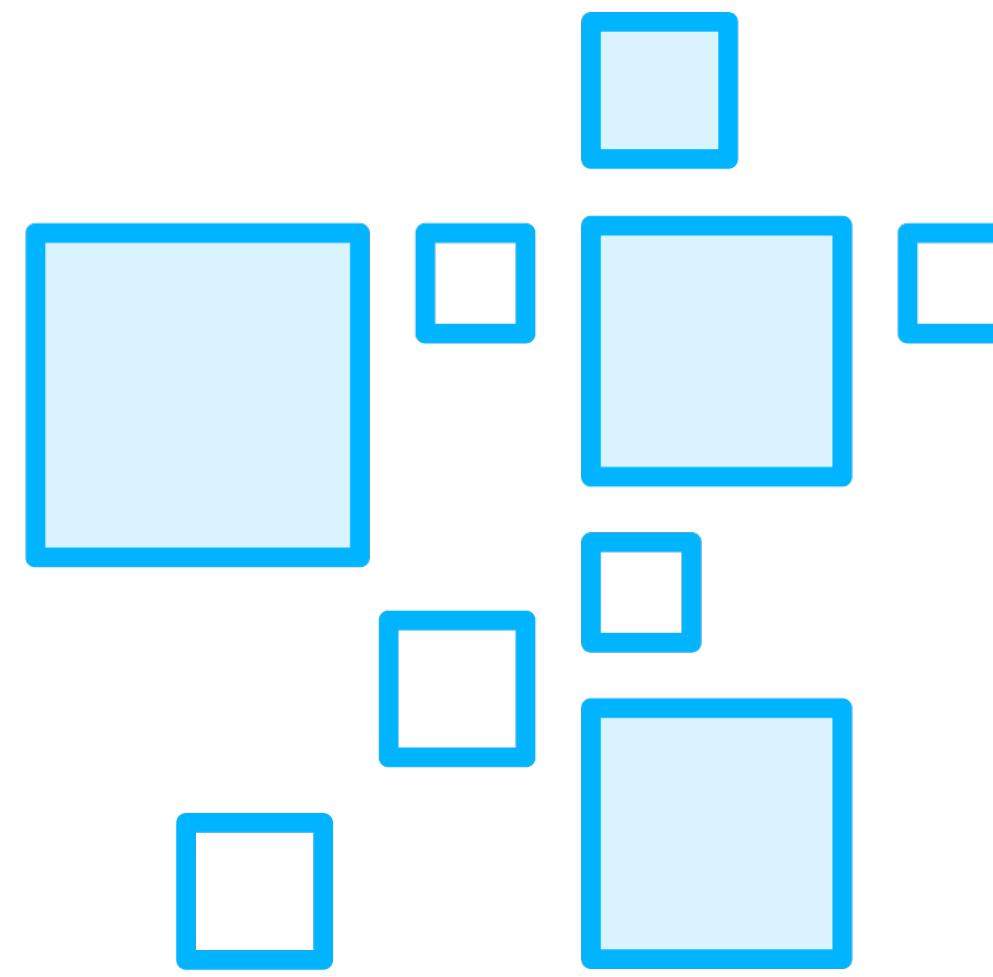


External resources

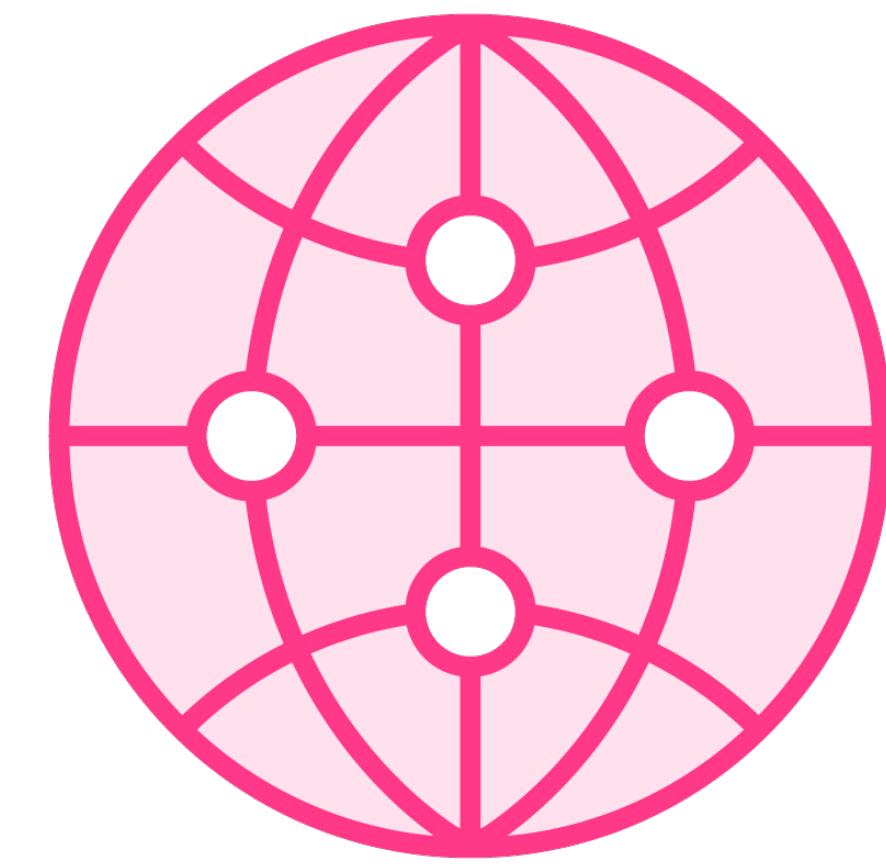


Planning Ahead to Learn About the Domain

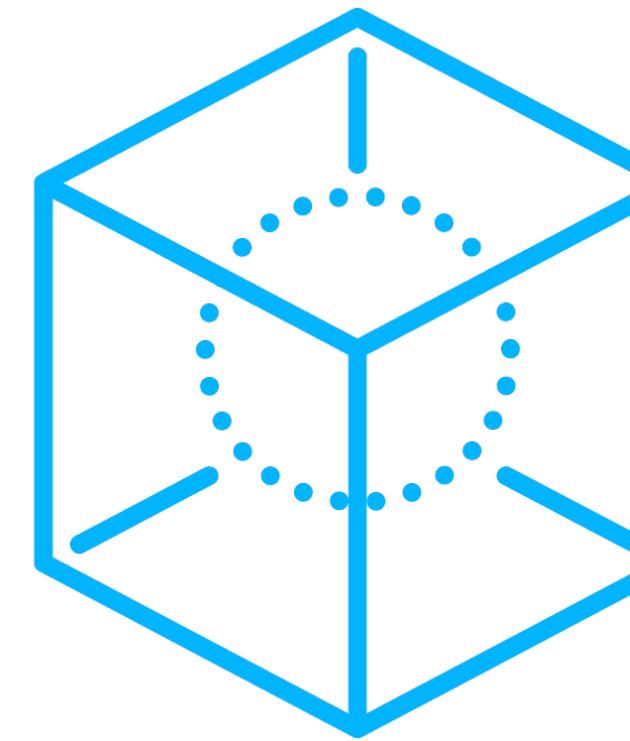
Our Goals for Learning About the Domain



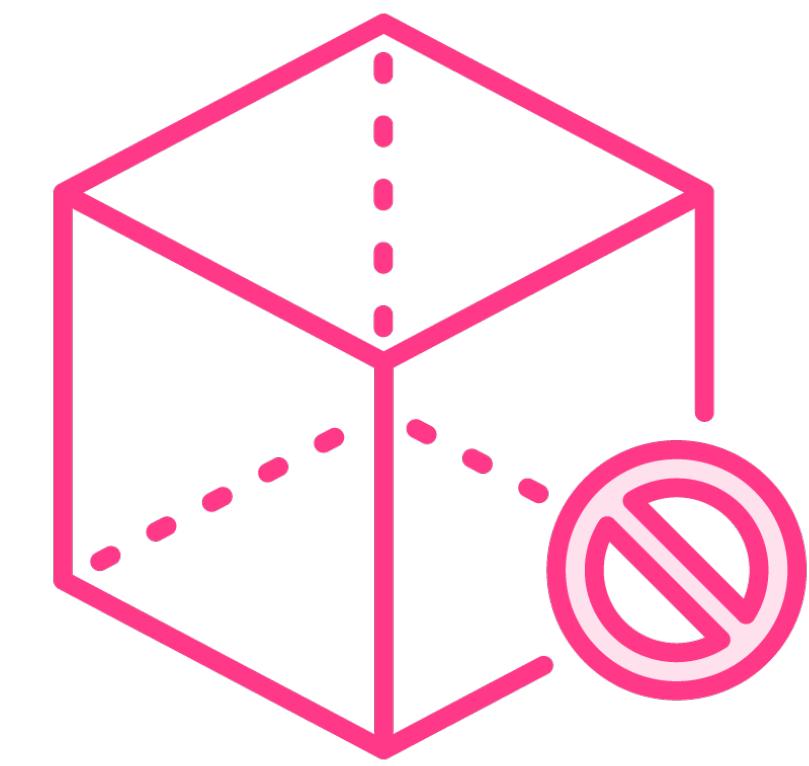
**Understand
client's business**



**Identify processes
beyond project
scope**



**Look for
subdomains we
should include**



**Look for
subdomains we
can ignore**

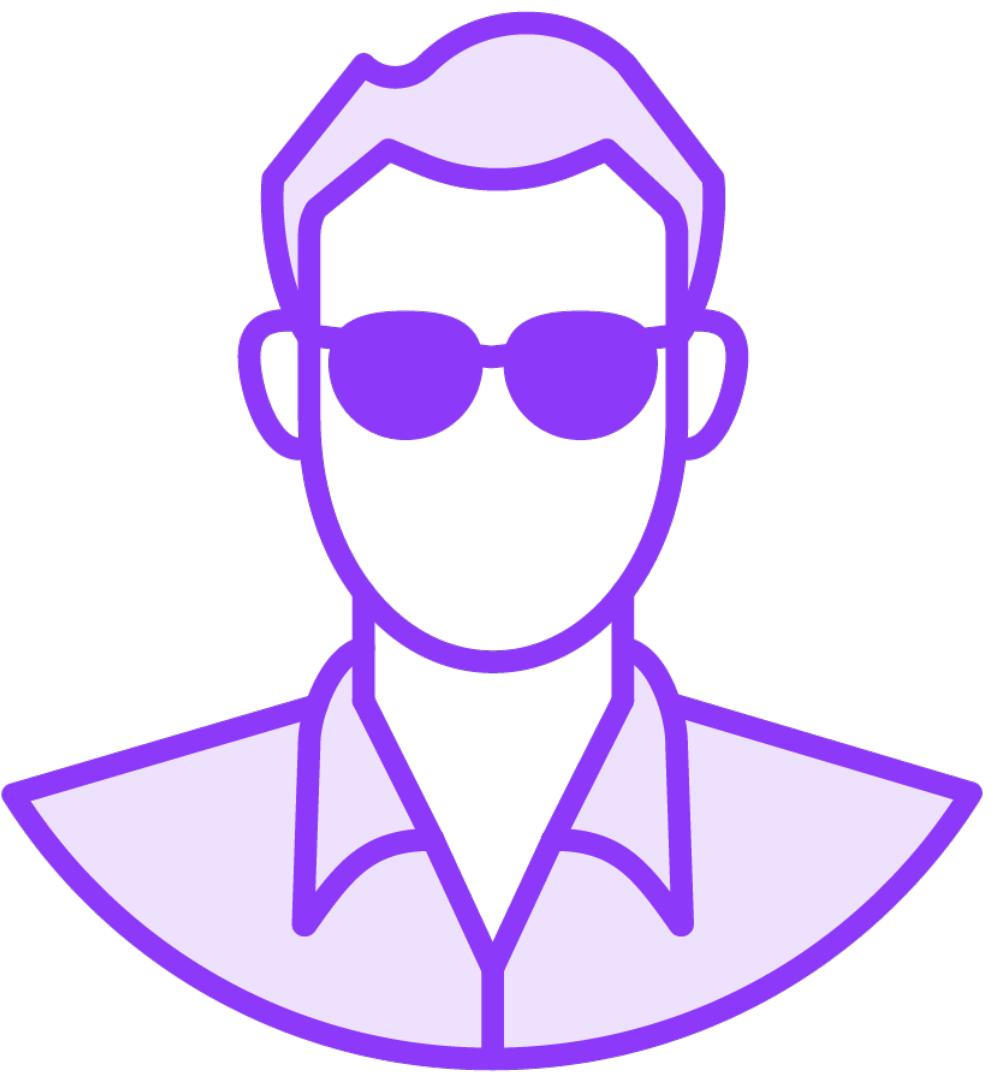


Conversation with a Domain Expert: Exploring the Domain and Its Subdomains

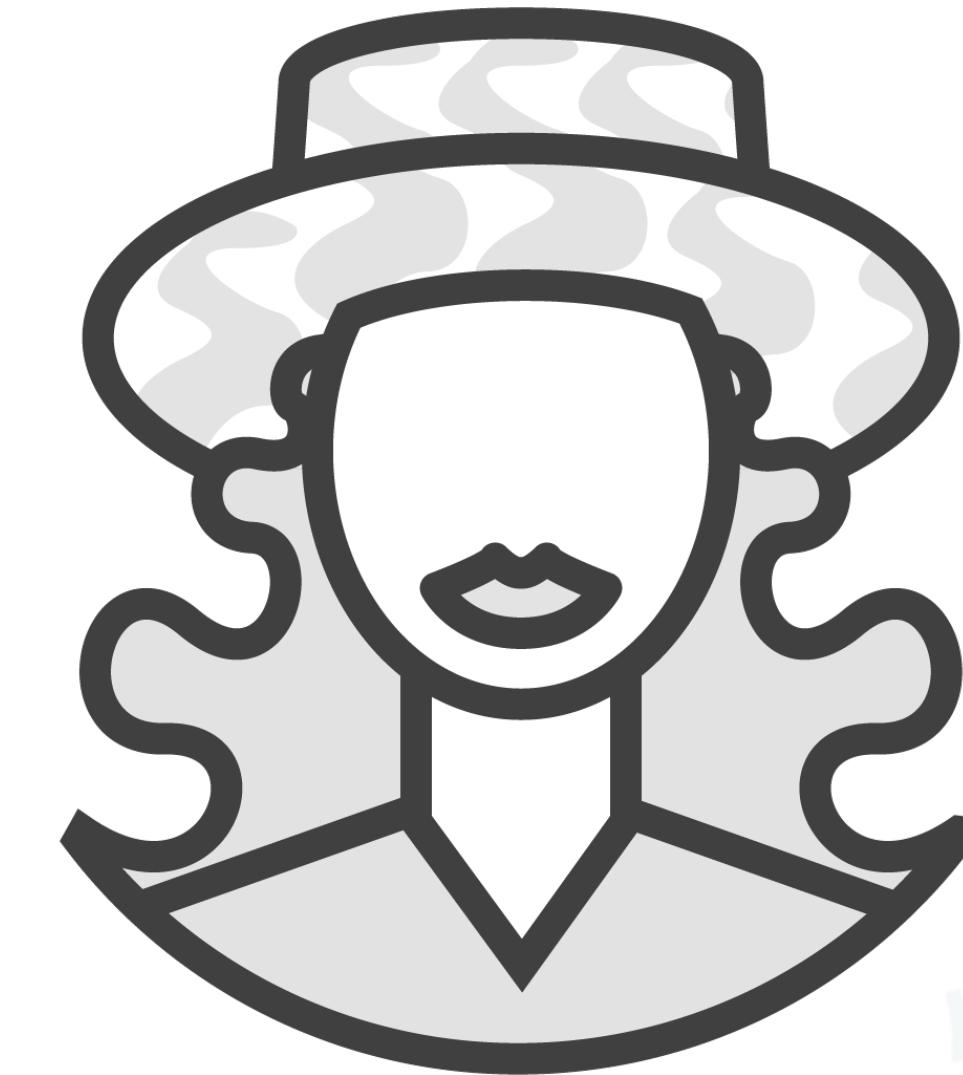
Conversation with a Domain Expert



Dr. Smith



Steve



Julie

Learning about the Complete Domain

Patient scheduling

Owner and pet
data management

Surgery scheduling

Office visit
data collection

Lab testing
(schedule, results, bill)

Prescriptions

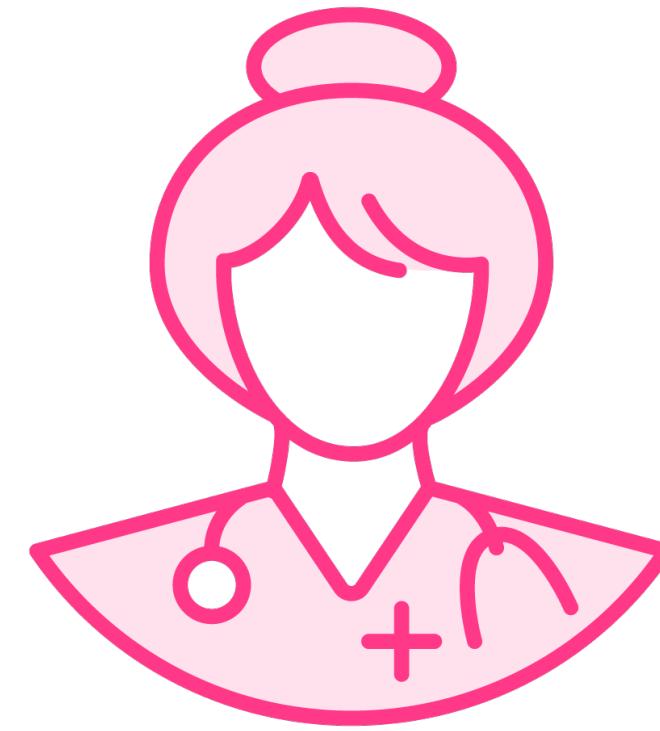
Billing (External?)

Sales and Inventory

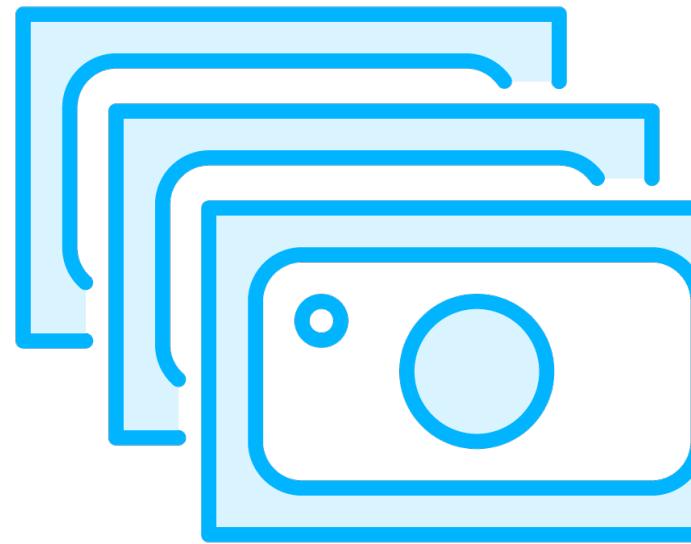
Staff scheduling

CMS (External?)

Some of the Identified Subdomains



Staff



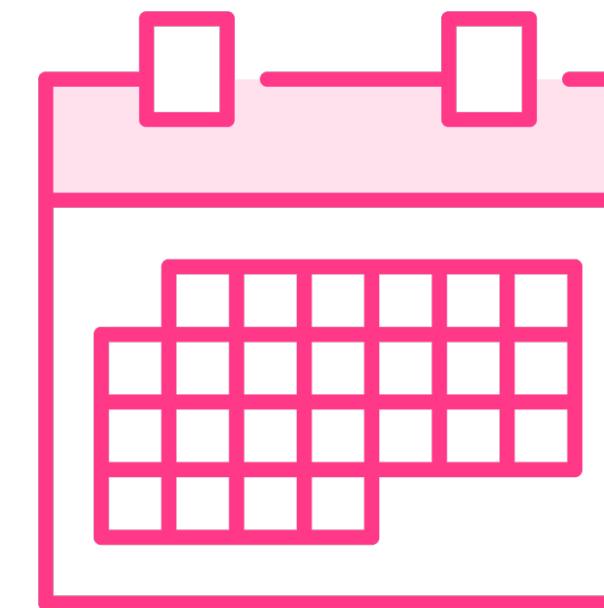
Accounting



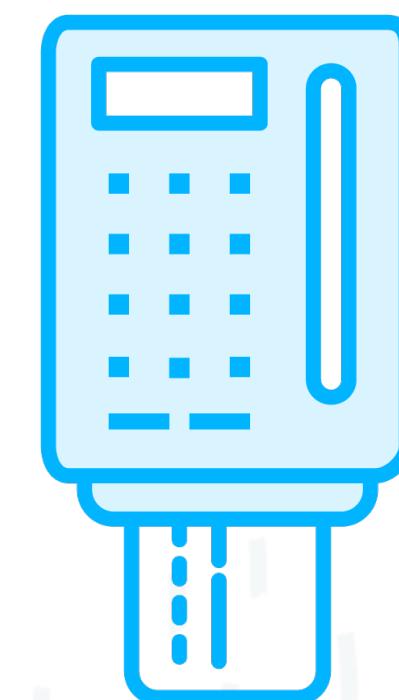
Client and patient records



Visit records



Appointment scheduling



Sales

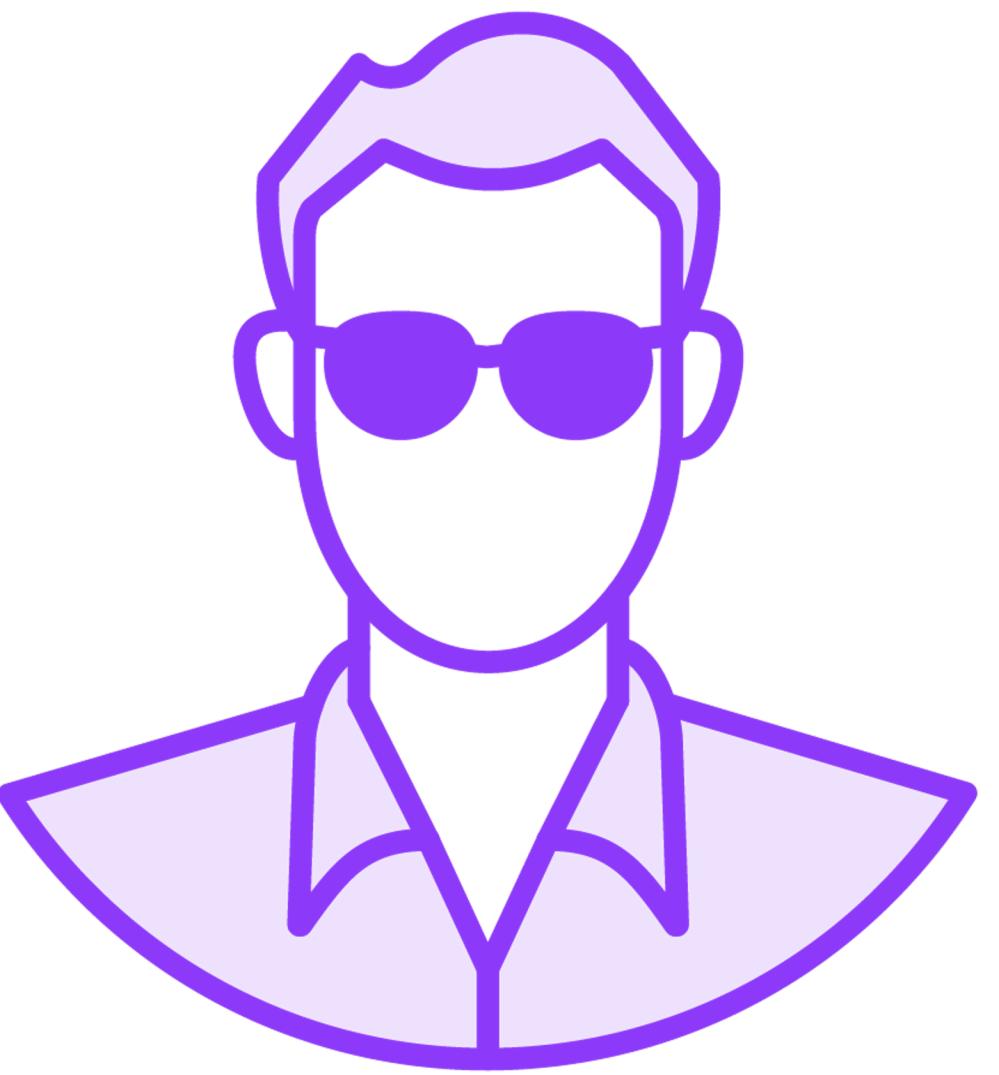


Conversation with a Domain Expert: Exploring the Scheduling Subdomain

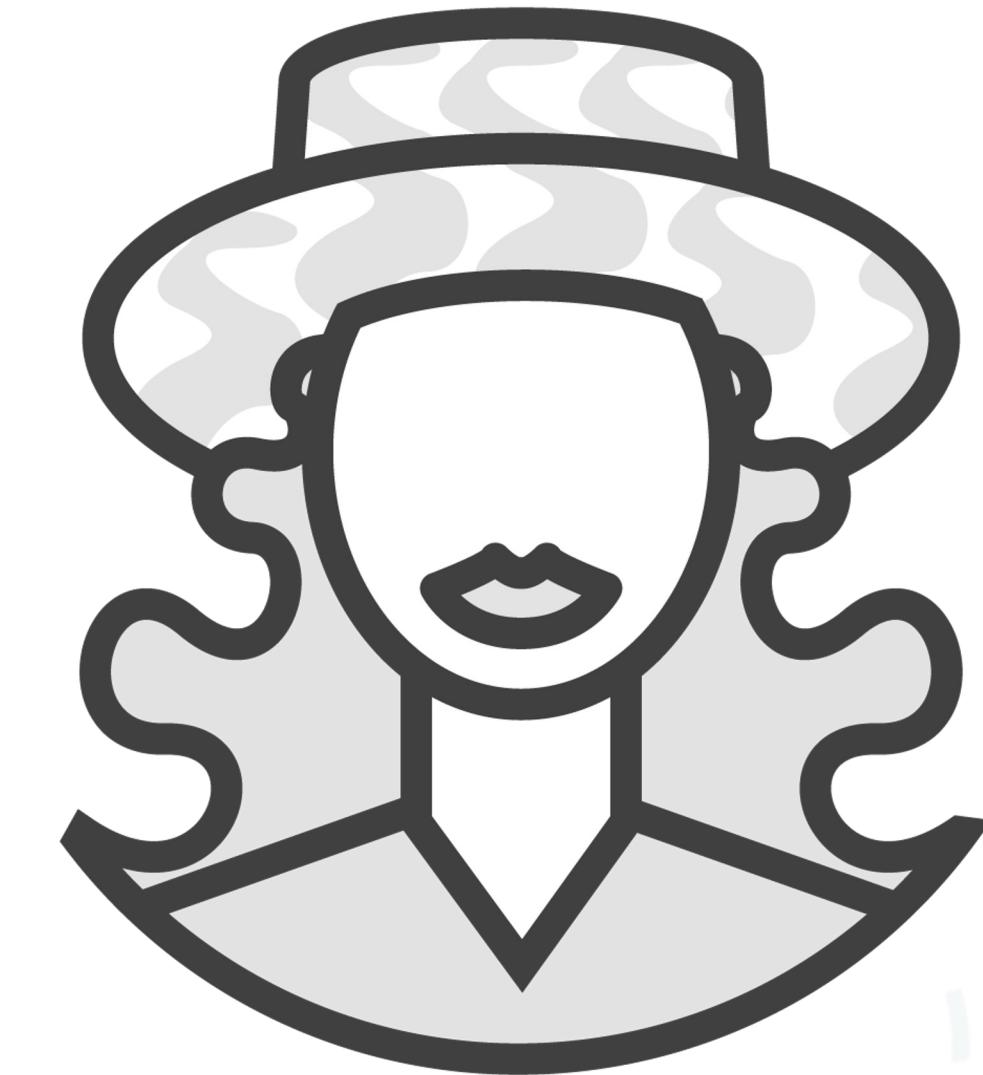
Conversation with a Domain Expert



Dr. Smith



Steve



Julie

Notes from Our Conversation



Clients (people) schedule appointments for patients (pets)

Appointments may be either office visits or surgeries

Office visits may be an exam requiring a doctor, or a tech visit

Office visits depend on exam room availability

Surgeries depend on O/R and recovery space availability, and can involve different kinds of procedures

Different appointment types and procedures require different staff

Lessons in Communicating with Domain Experts



Software class



Dog training class

**Learn and communicate in
the language of the domain,
not the language of
technology.**

More Questions for Dr. Smith



Any chance of concurrency conflicts?

Doesn't anticipate being big enough to have this problem any time soon



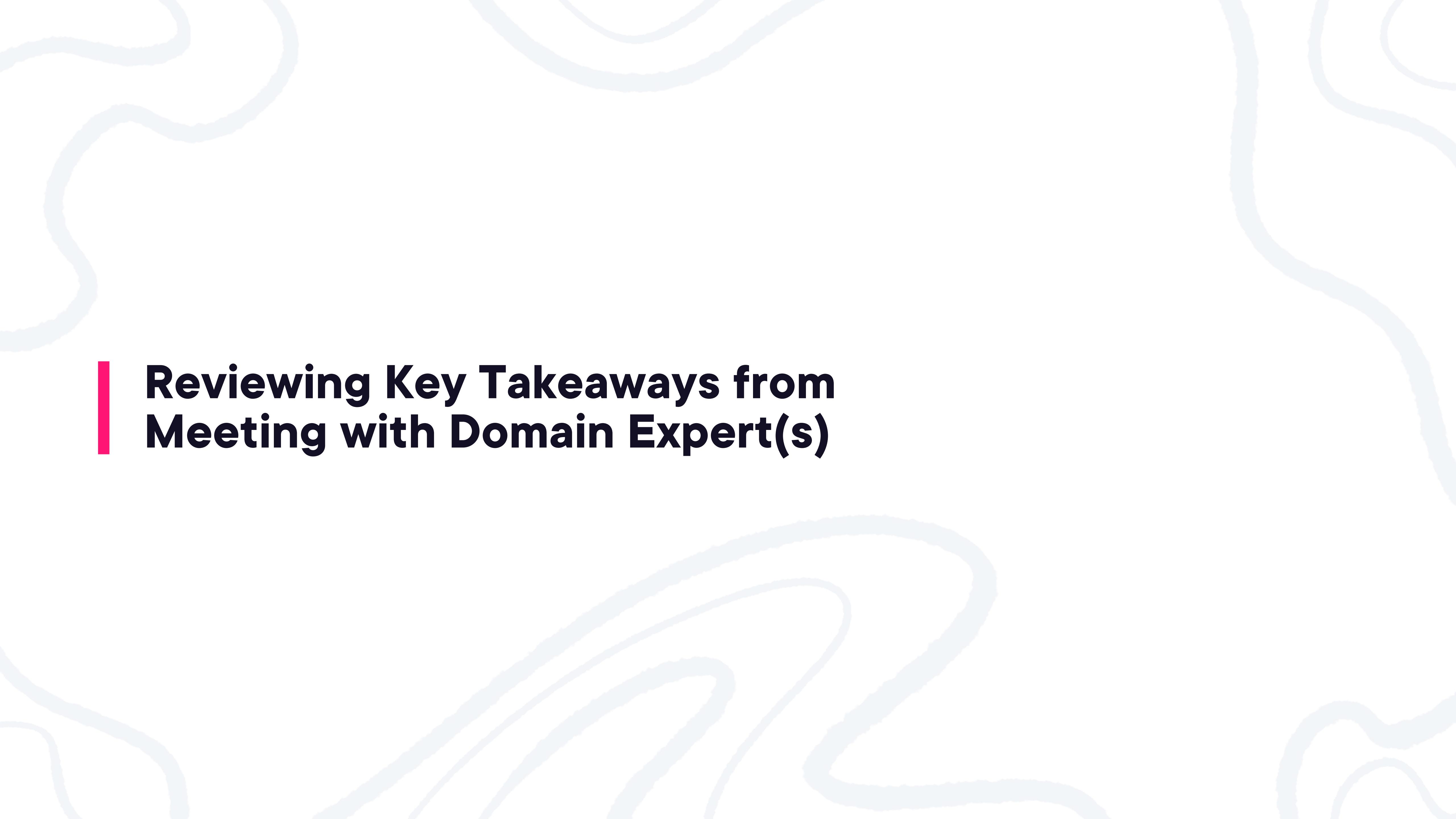
Do you need to schedule rooms and staff when you schedule an appointment? Dependencies?

Room + tech OR...
Room + doctor OR...
Room + doctor + tech



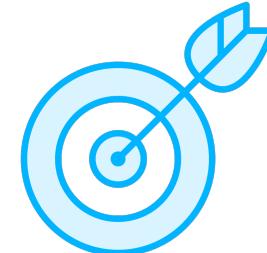
Does “resources” work as an umbrella term for them?

Yes!

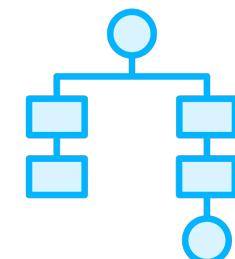


Reviewing Key Takeaways from Meeting with Domain Expert(s)

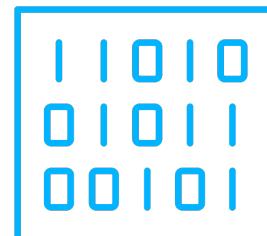
Key Takeaways from the Customer Conversation



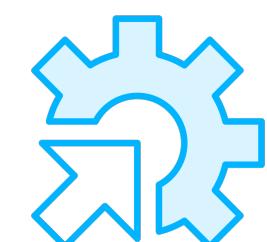
Important to get on the “same page” with the domain expert



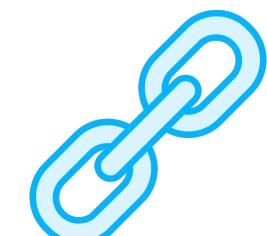
The customer gained better understanding of their own business process by describing it in terms we could understand and model



Avoid speaking in programmer terms



At this stage, the focus is on how the domain works, not how the software will work



Make the implicit knowledge of domain experts explicit

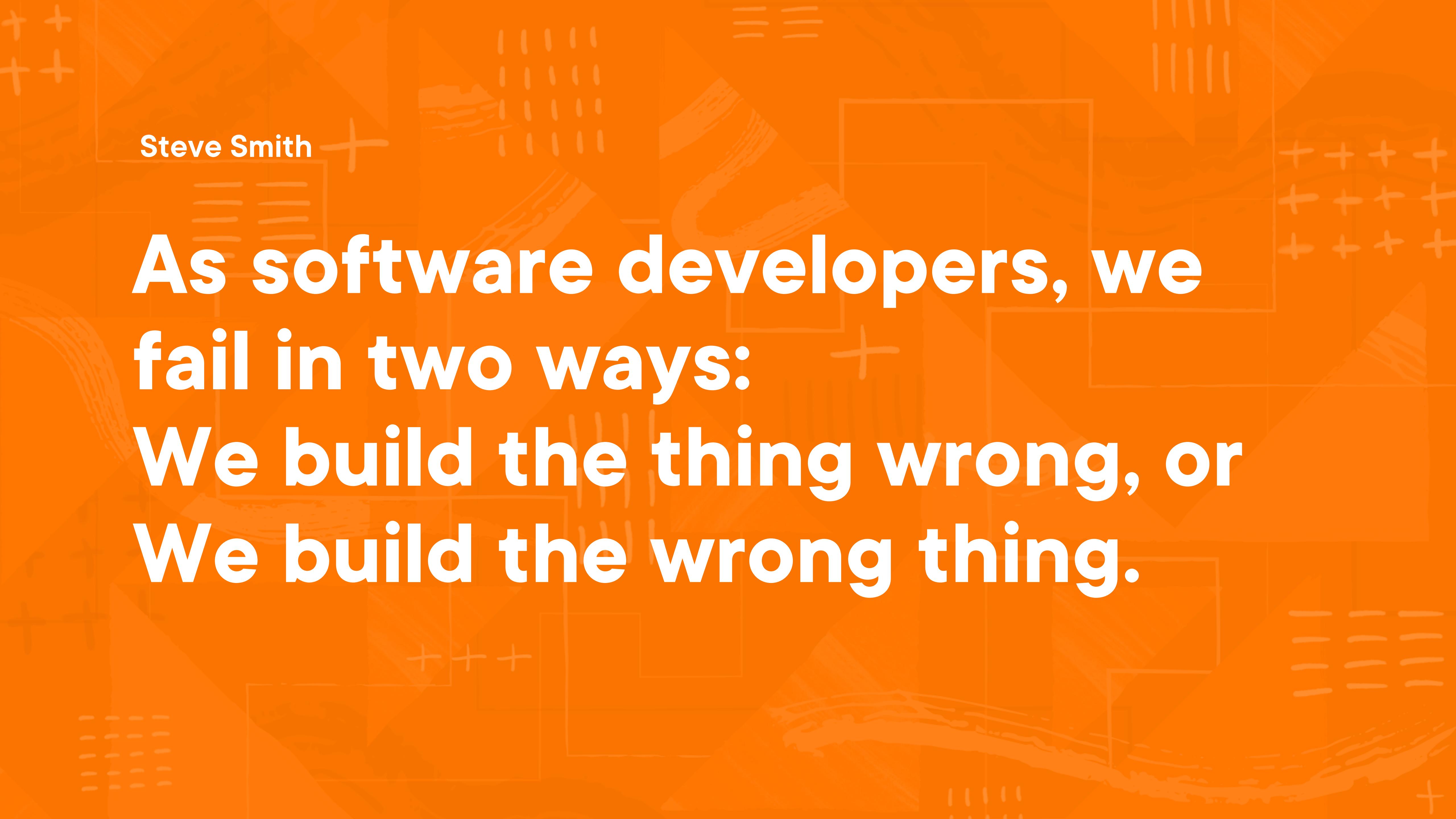
Domain Exploration Can Benefit the Domain Experts

“Wow, I never thought of that before!”

**“Considering our rules more explicitly
can help us in the future.”**

“This is going to be great!”



A photograph of a person's hands typing on a white keyboard. A light gray grid pattern is overlaid on the entire image, consisting of vertical and horizontal lines and small plus signs at intersections.

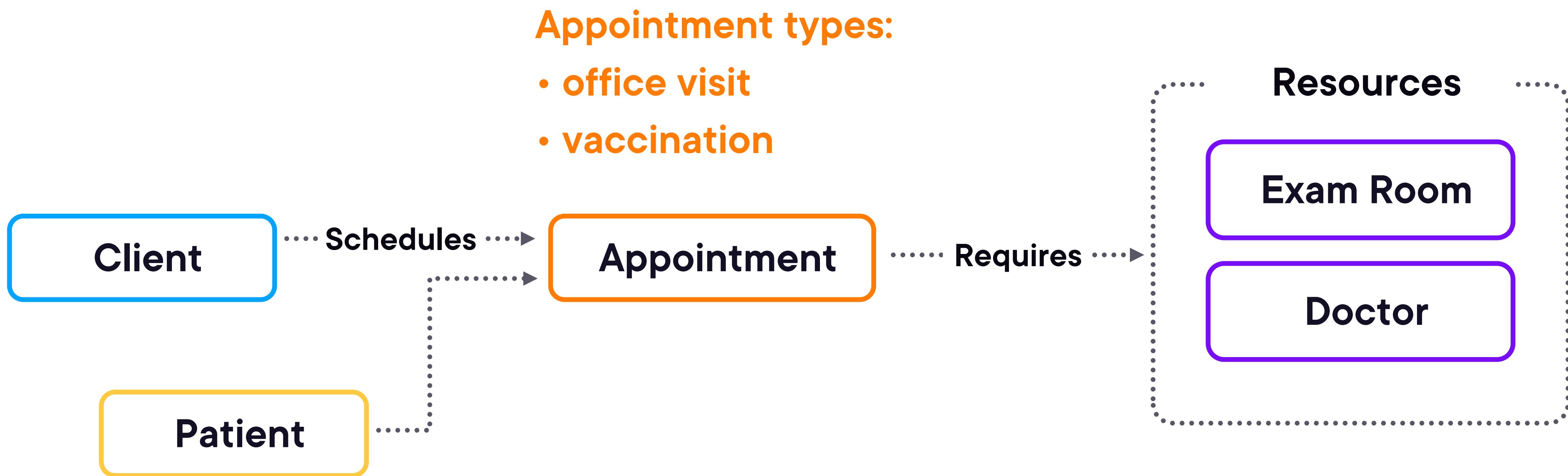
Steve Smith

**As software developers, we
fail in two ways:
We build the thing wrong, or
We build the wrong thing.**

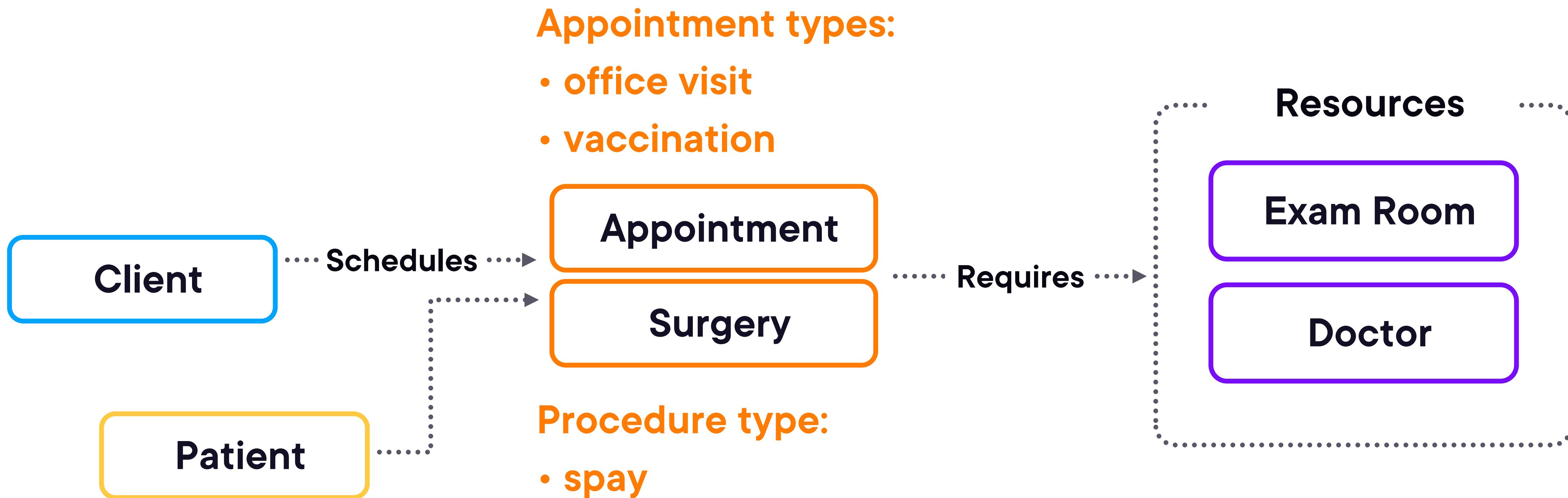


Taking a First Pass at Modeling Our Subdomain

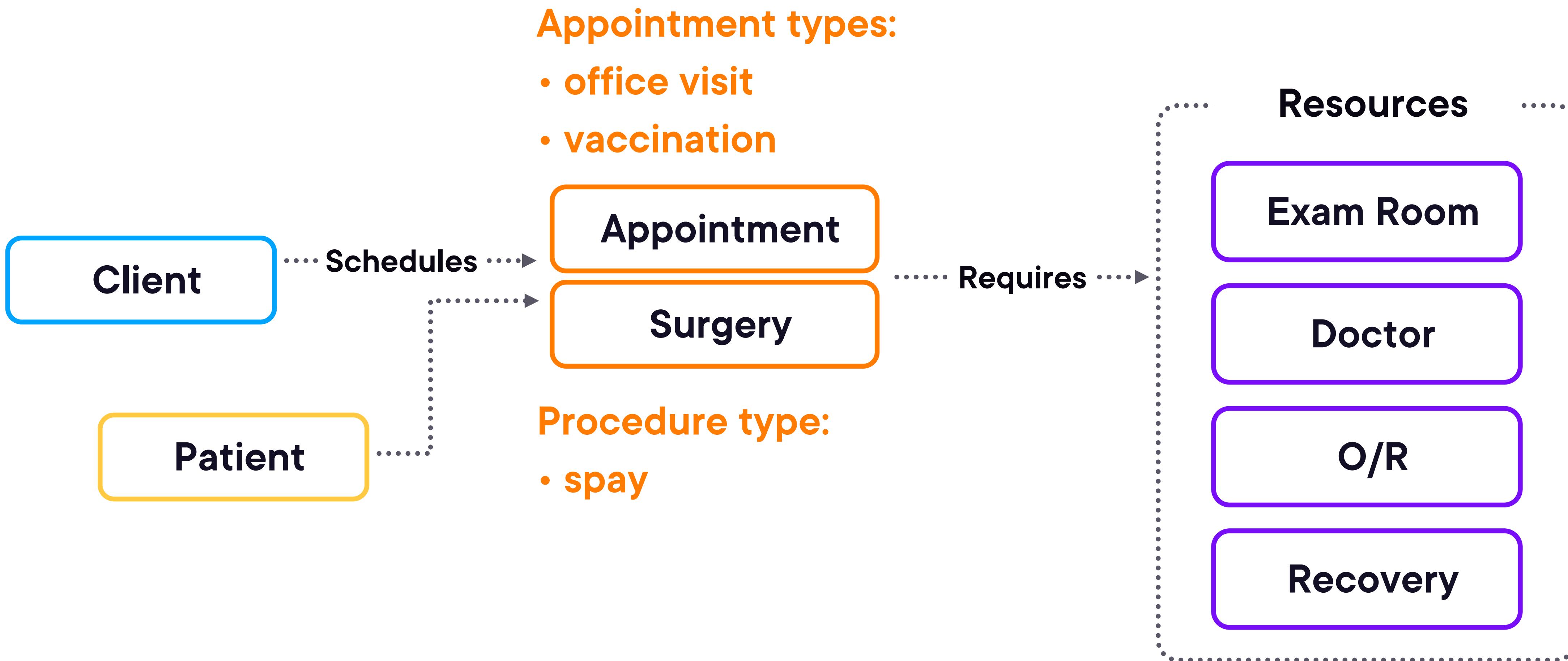
What We Know About the Appointment Manager Model



What We Know About the Appointment Manager Model



What We Know About the Appointment Manager Model





Using Bounded Contexts to Untangle Concepts that Appear to Be Shared

Defining Bounded Contexts



Define a strong boundary around the concepts of each model



Ensure model's concepts don't leak into other models where they don't make sense

But What About Those Common Types?

Appointment Scheduling

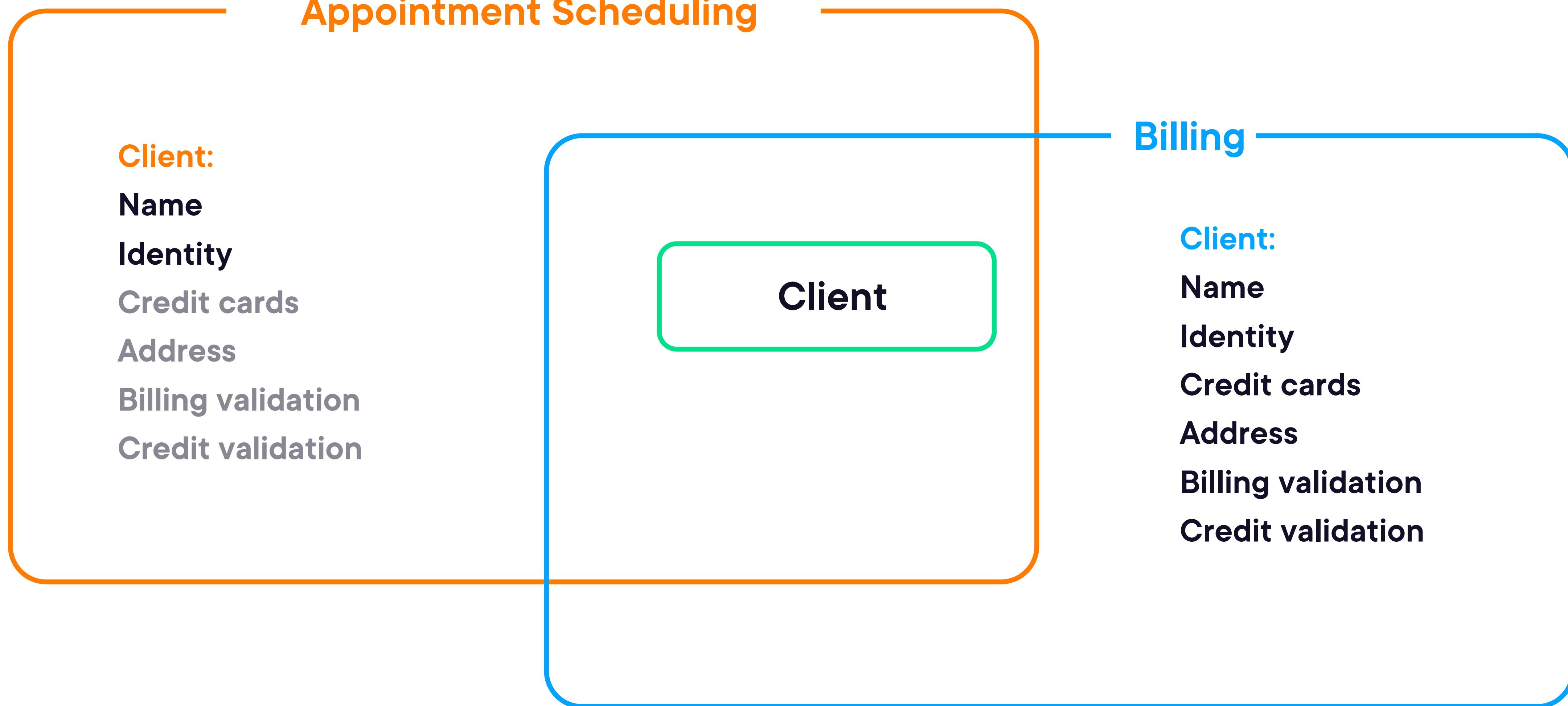
Client

Client:

Name

Identity

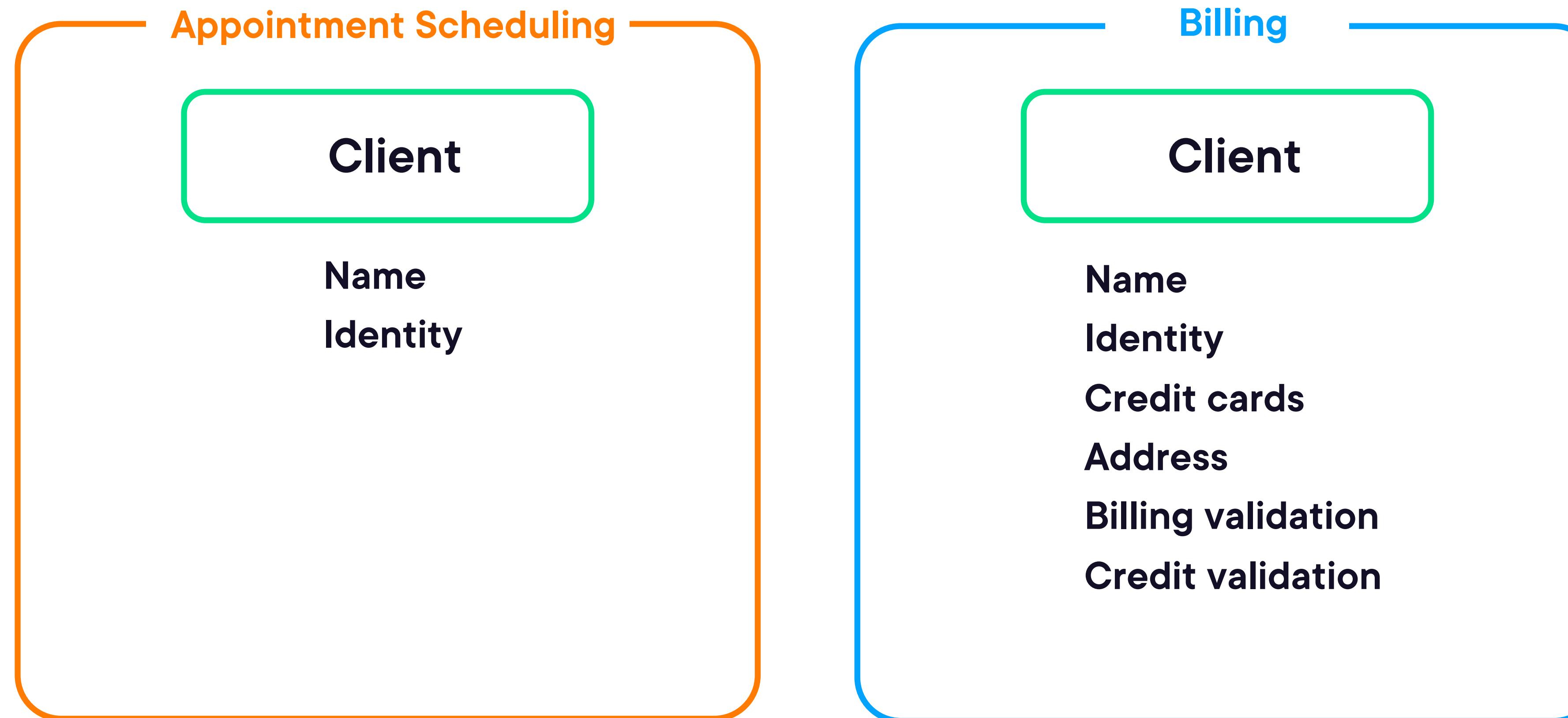
But What About Those Common Types?



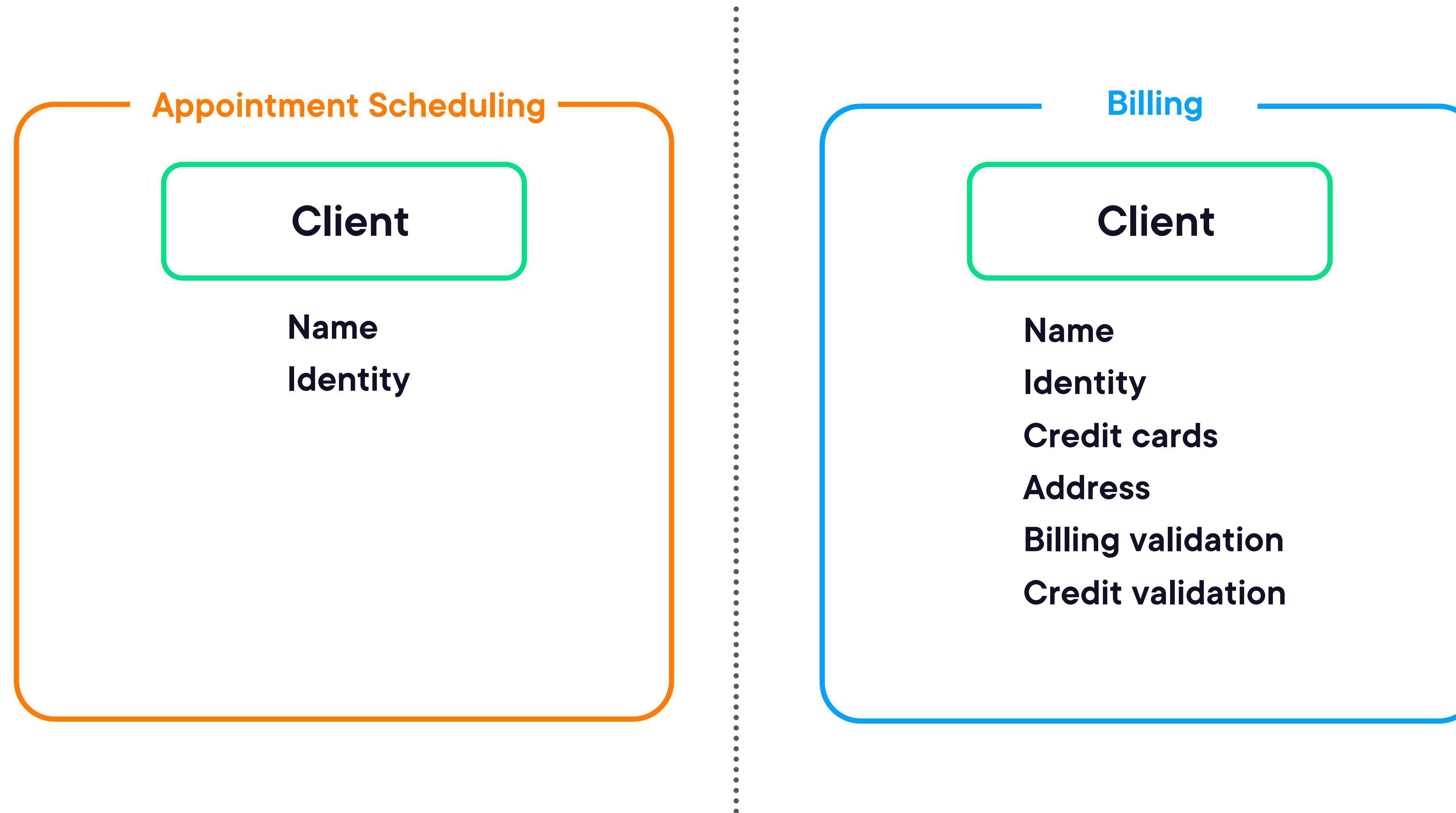
Eric Evans

Explicitly define the context within which a model applies... Keep the model strictly consistent within these bounds, but don't be distracted or confused by issues outside.

Bounded Context



Bounded Context



Distinct Data, Code, and Teams per Bounded Context?



Rarely see that level
of separation in real world

You can introduce separation through
namespaces, folders, and projects

While 100% is not achievable, strive for it



At least keep the concepts
in mind for guidance

Not hard rules, but guidance

Let DDD guide you



Conversation with Eric Evans on Subdomains and Bounded Contexts



Subdomain
is a problem space concept

Bounded Context
is a solution space concept



Problem Space vs. Solution Space



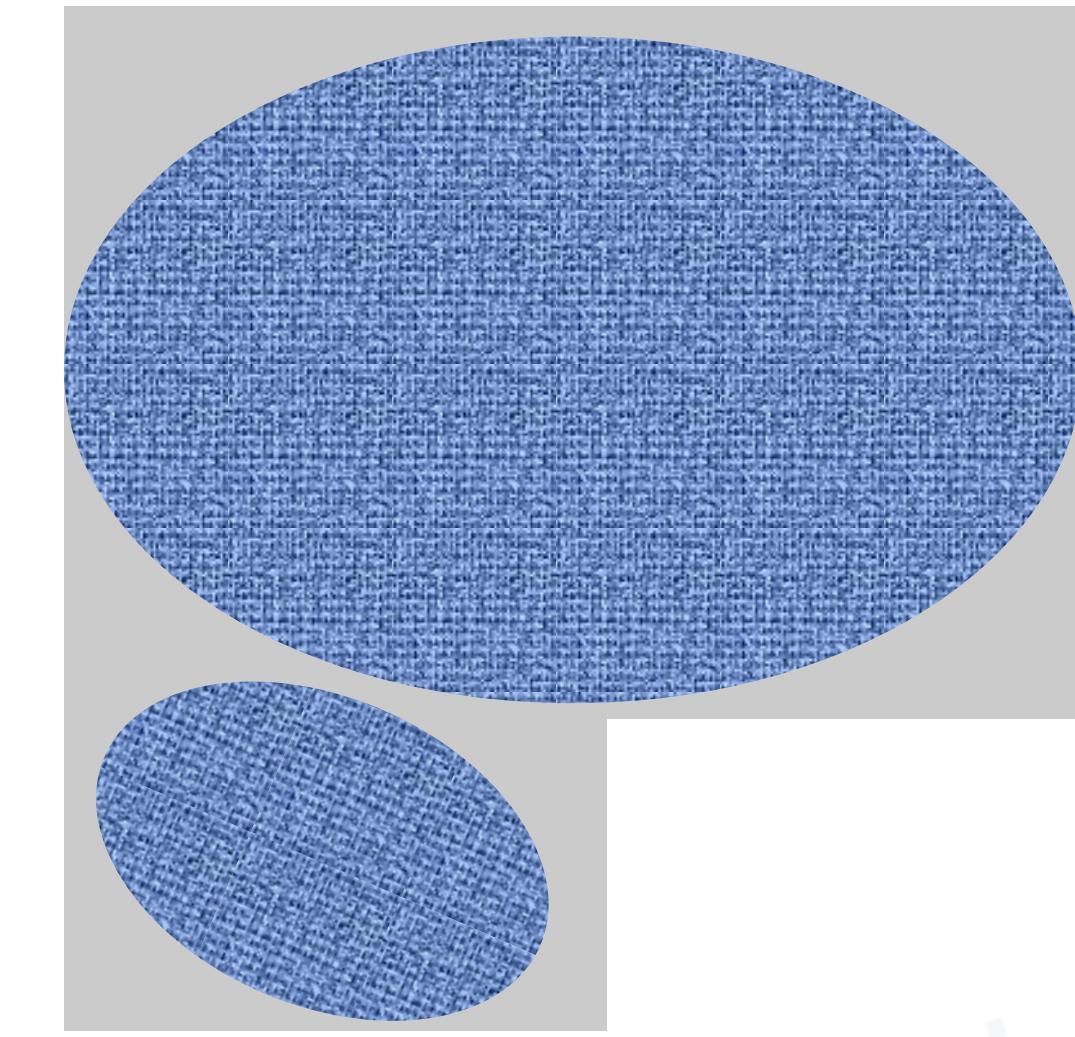
Problem Space

Bare floor of a room



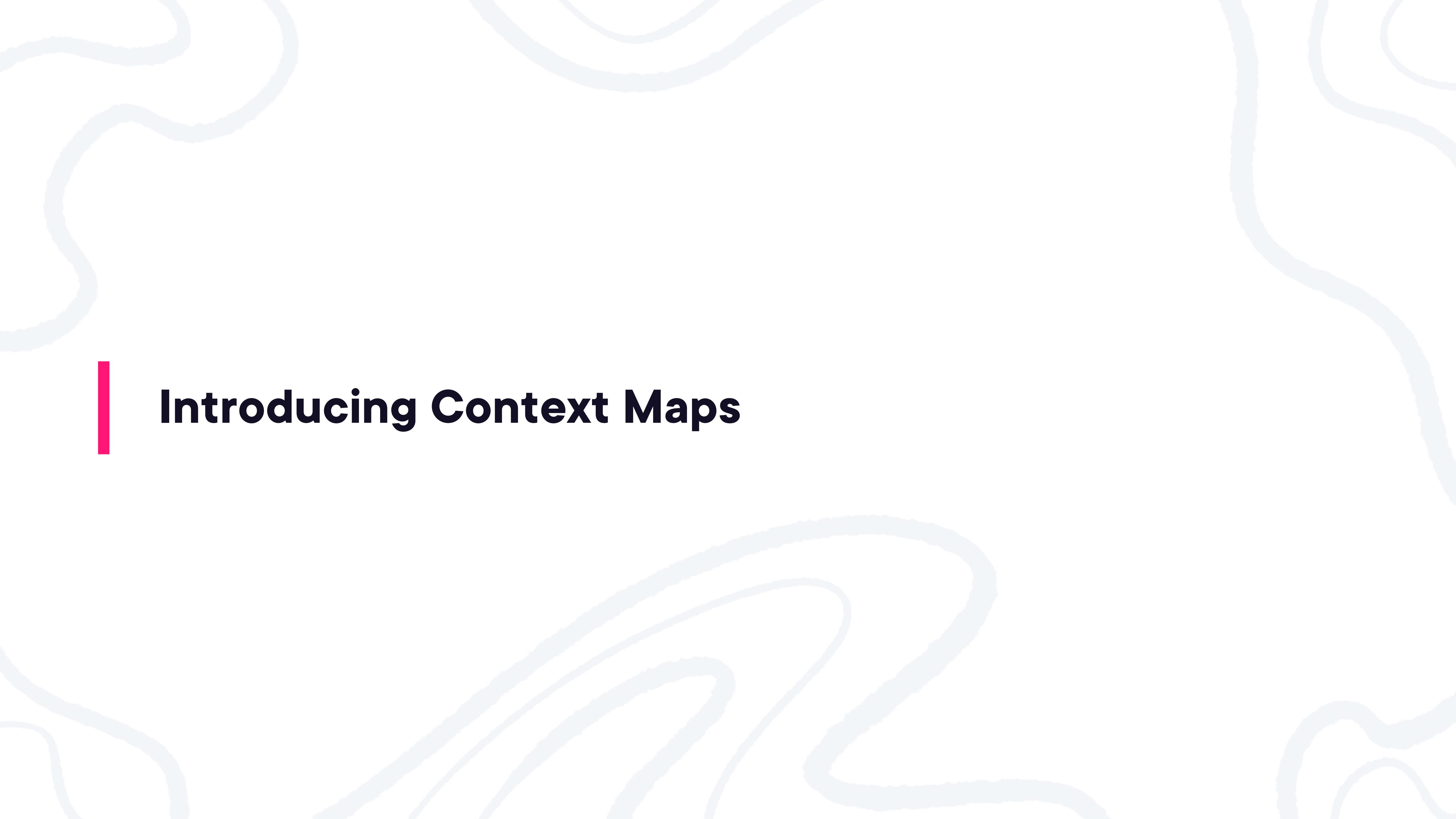
Solution Space A

Wall-to-wall carpet
Same shape and size
as the problem



Solution Space B

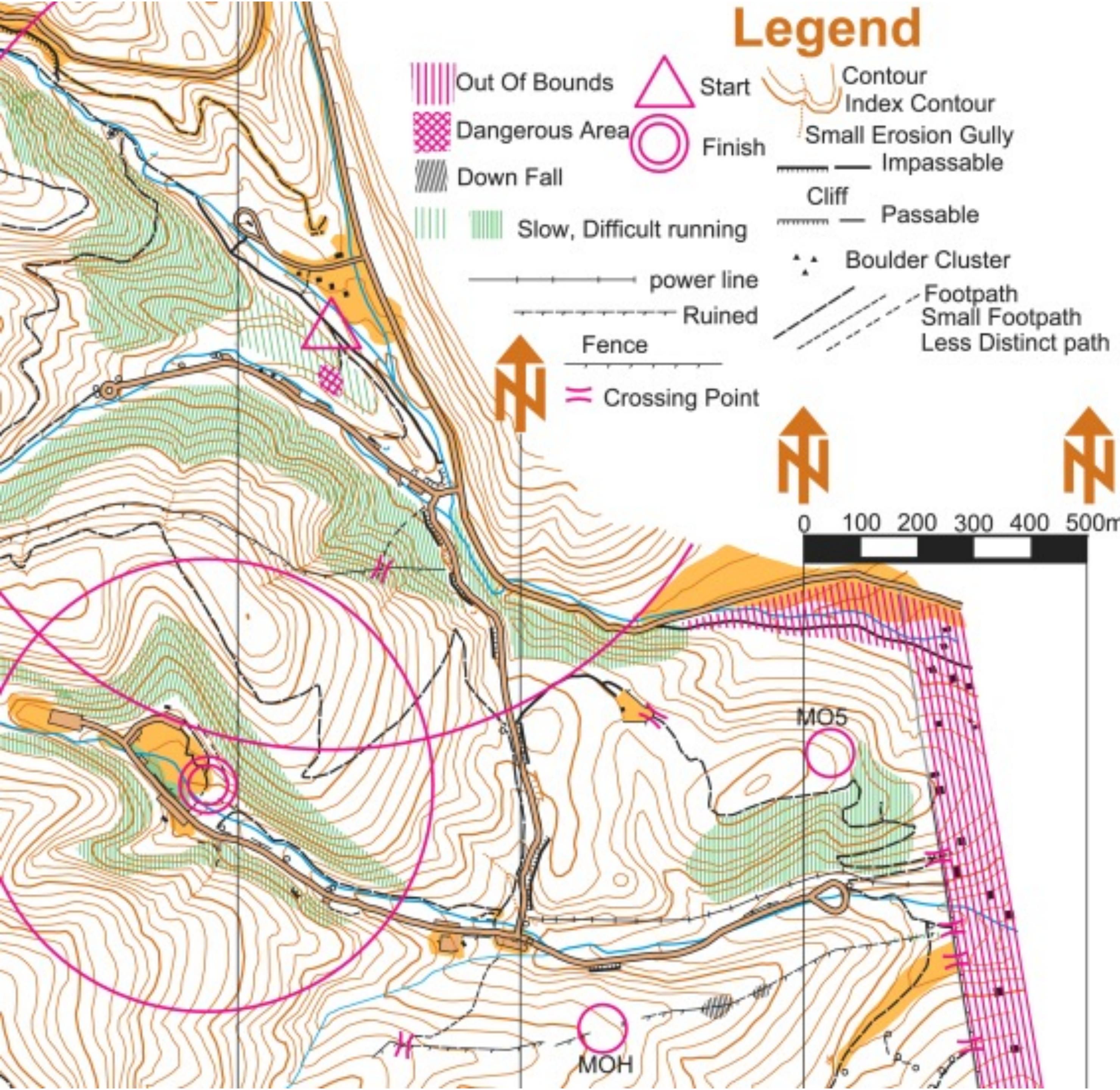
Area rugs
Clearly a different shape
and size as the problem



Introducing Context Maps

Context Map

Demonstrates how bounded contexts connect to one another while supporting communication between teams.



Source:
commons.wikimedia.org/wiki/File:Ardf_map.png

Context Maps

Appointment Scheduler

Client

Patient

Appointment

Notification

Exam Room

The Dev Team
Vet-manager



Billing

Client

Procedure

Invoice

Notification

Context Maps

Appointment Scheduler

Client

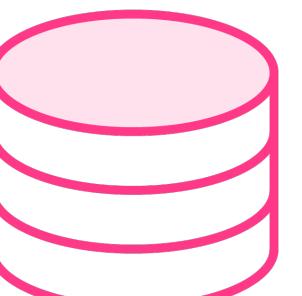
Patient

Appointment

Notification

Exam Room

Team Awesome
Vet-app-sched



Billing

Client

Procedure

Invoice

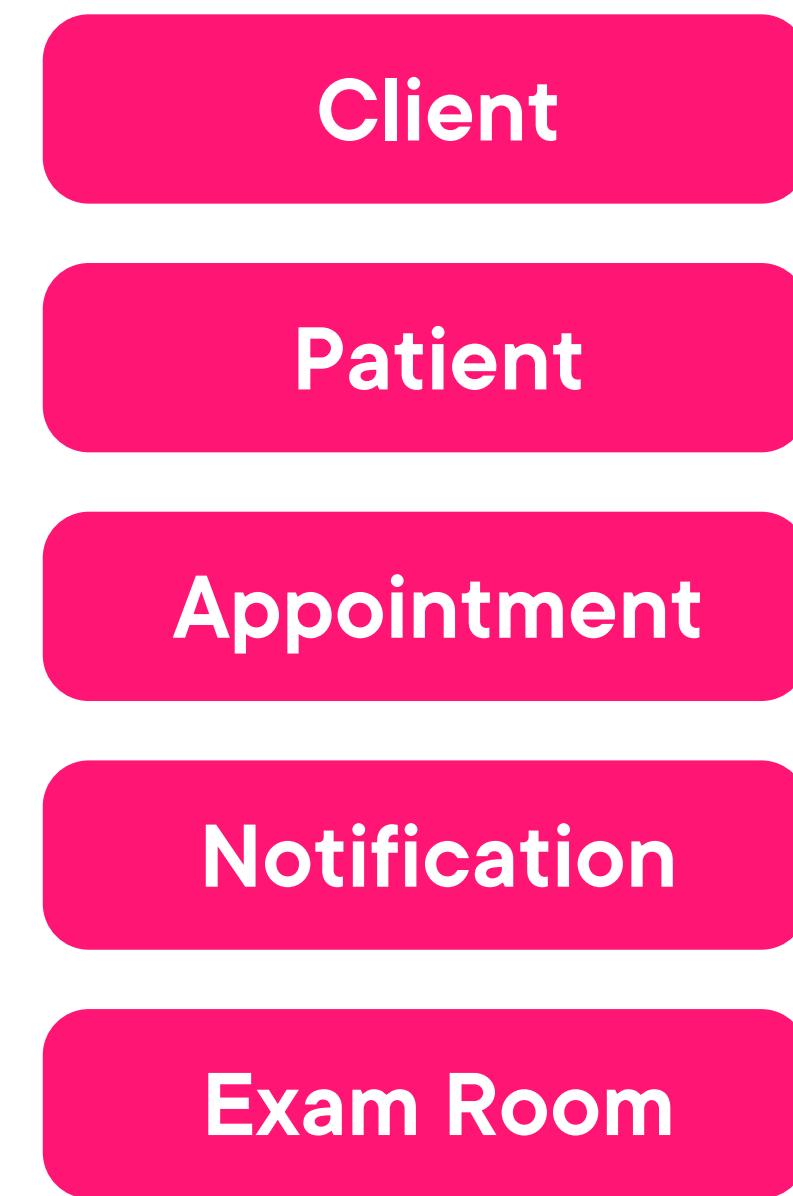
Notification

Team Ultimate
Vet-billing

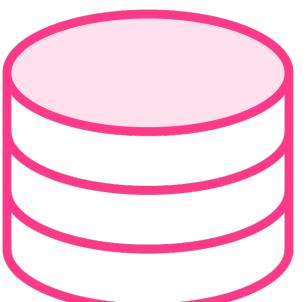


Context Maps

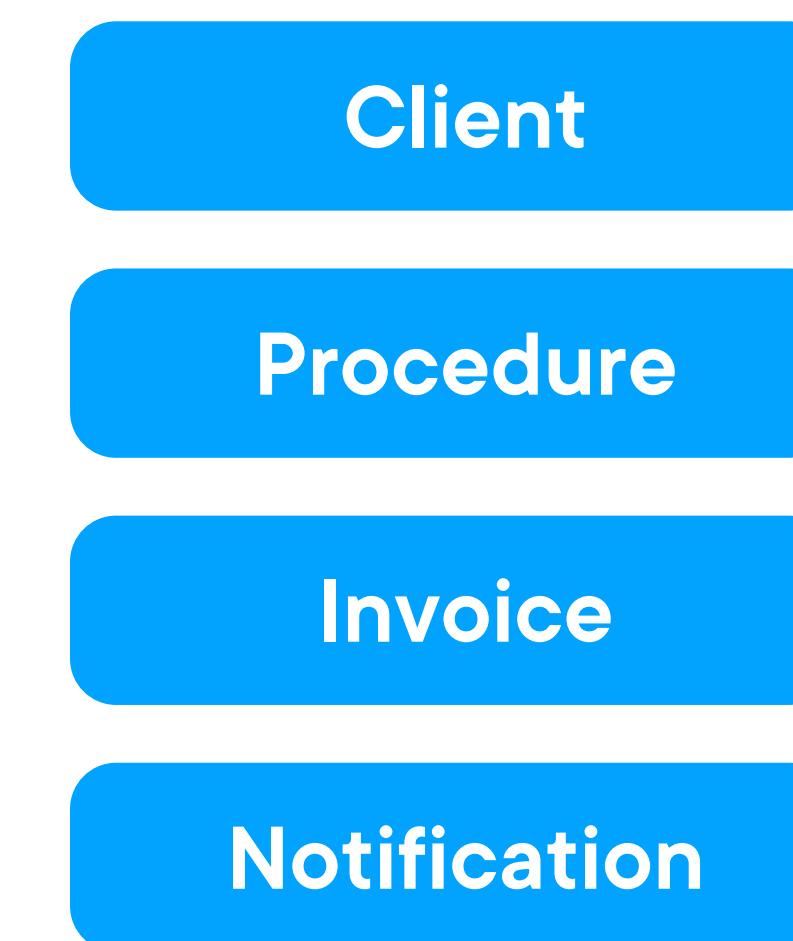
Appointment Scheduler



Team Awesome
Vet-app-sched



Billing



Team Ultimate
Vet-billing



Context Maps

Appointment Scheduler

Client

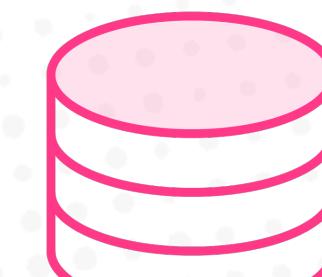
Patient

Appointment

Notification

Exam Room

Team Awesome
Vet-app-sched



Shared Kernel

Logging

PersonName

Billing

Client

Procedure

Invoice

Notification

Team Ultimate
Vet-billing





Addressing the Question of Separate Databases per Bounded Context

Context Maps

Appointment Scheduler

Client

Patient

Appointment

Notification

Exam Room

Team Awesome
Vet-app-sched



Shared Kernel

Logging

PersonName

Billing

Client

Procedure

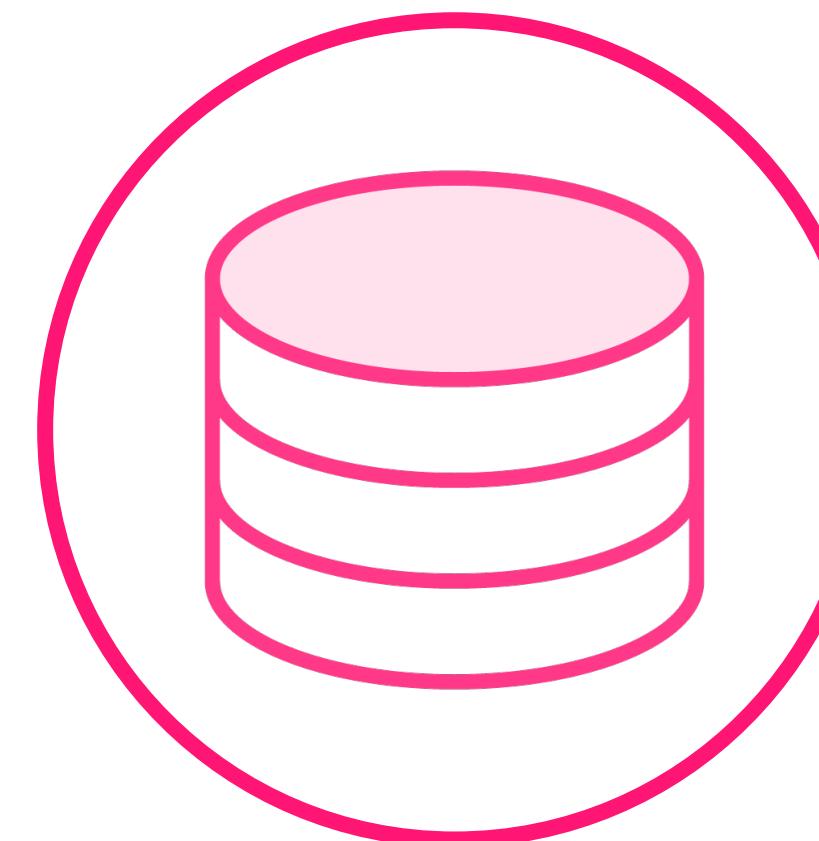
Invoice

Notification

Team Ultimate
Vet-billing



So Many Databases for Bounded Contexts



So Many Databases for Microservices



Eric Evans

“If you’re in a company where you share your database and it gets updated by hundreds of different processes, it’s very hard to create the kind of models that we’re talking about and then write software that does anything interesting with those models.”

Syncing Data Between the DBs of the Separate BCs

Appointment Scheduler

Client

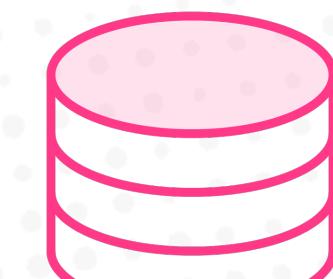
Patient

Appointment

Notification

Exam Room

Team Awesome
Vet-app-sched



Shared Kernel

Logging

PersonName

Billing

Client

Procedure

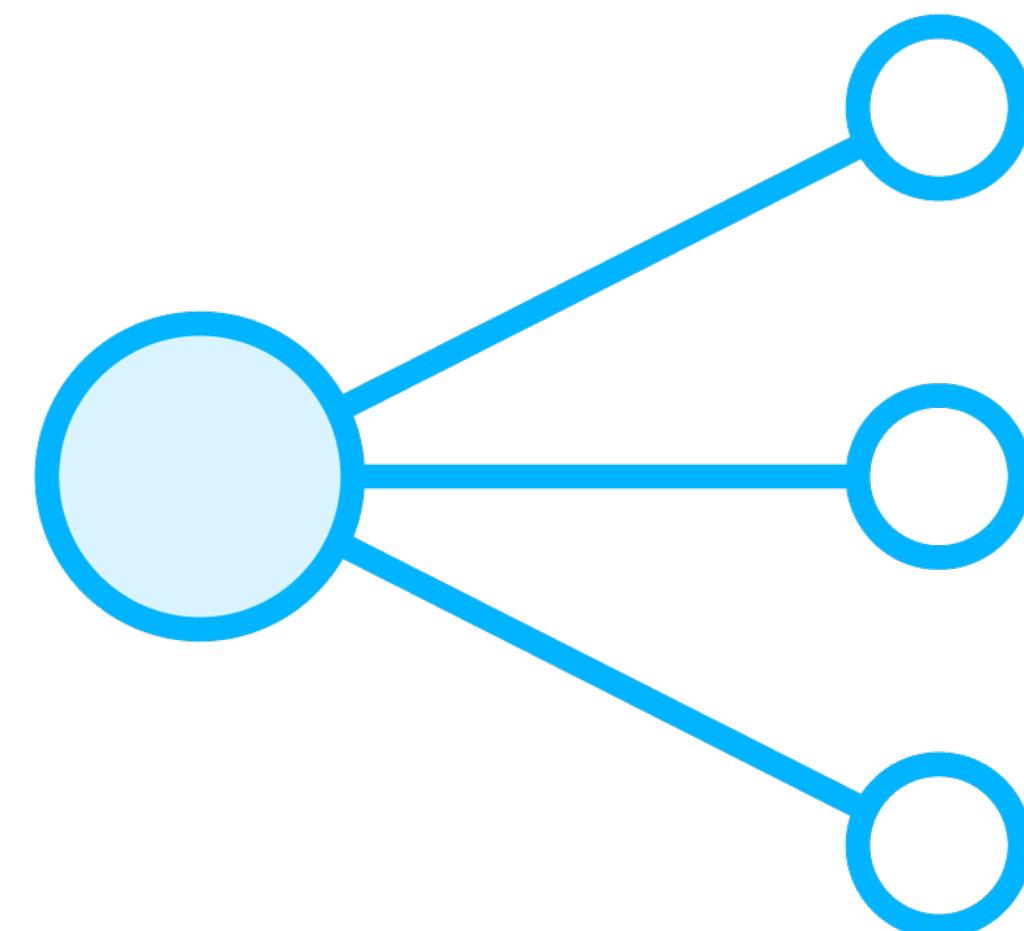
Invoice

Notification

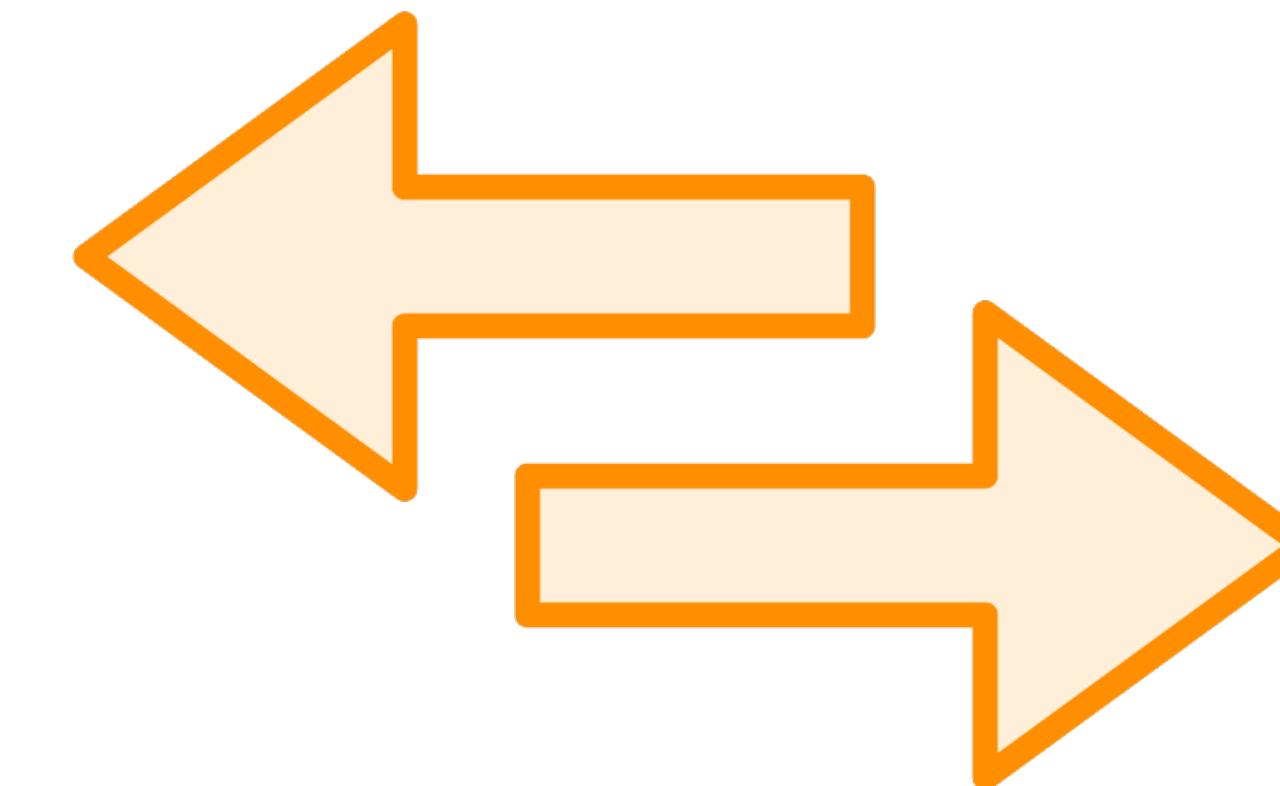
Team Ultimate
Vet-billing



Some Common Patterns for Data Syncing



Publisher Subscriber



2-way Synchronization

Implementations

Message Queues

Database processes

Batch jobs

Synchronization APIs

and more ...

Specifying Bounded Contexts in Our Application



Eric Evans

It's not a simple task, even with experience.

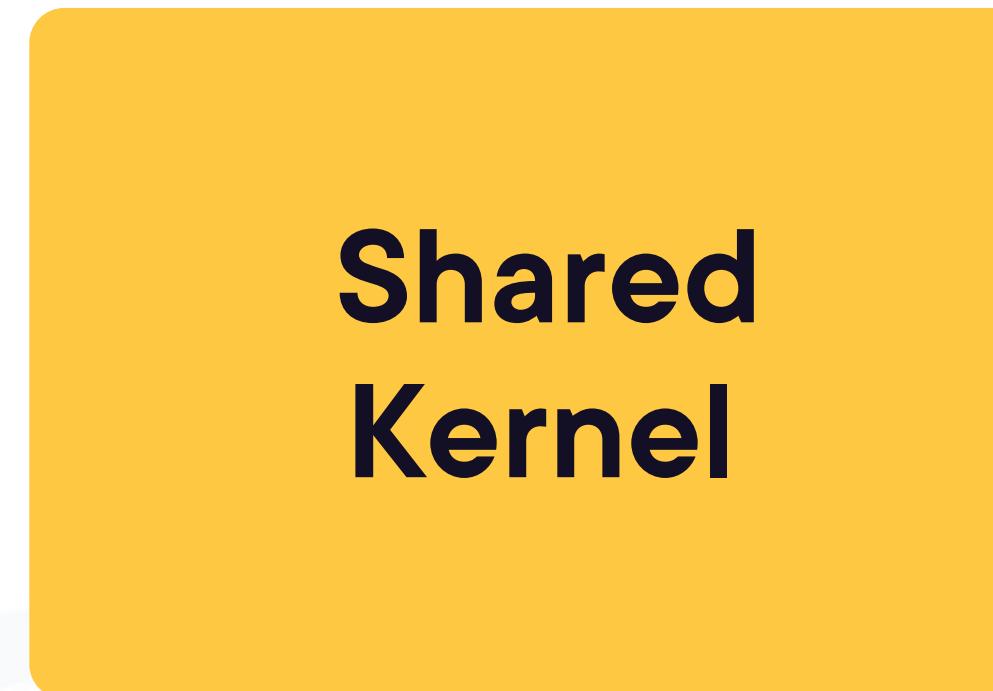
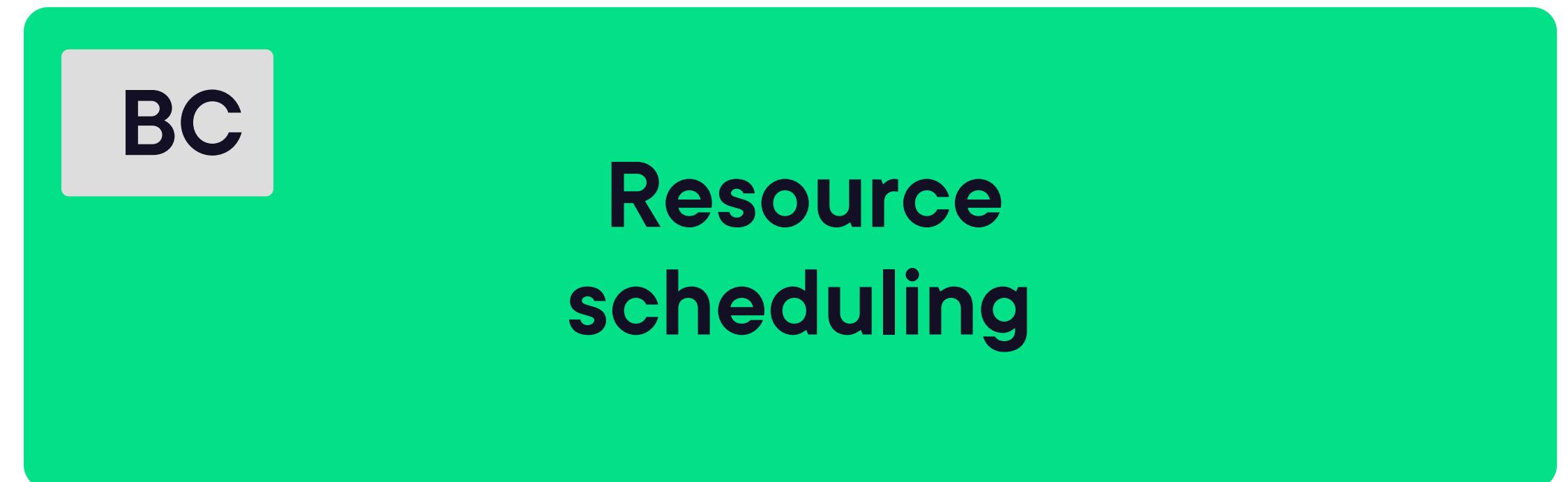
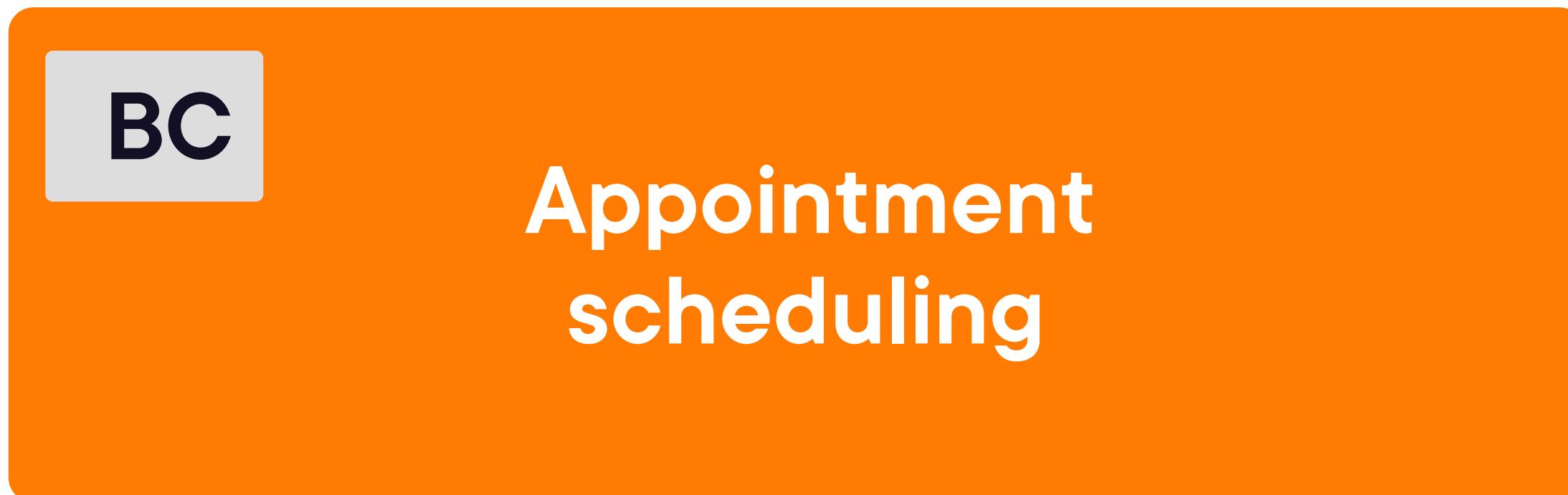
Lack of clear boundary impedes the application of DDD ideas.

Bounded context is an essential ingredient.

Defining boundaries is the biggest stumbling block.



Bounded Contexts in the Application



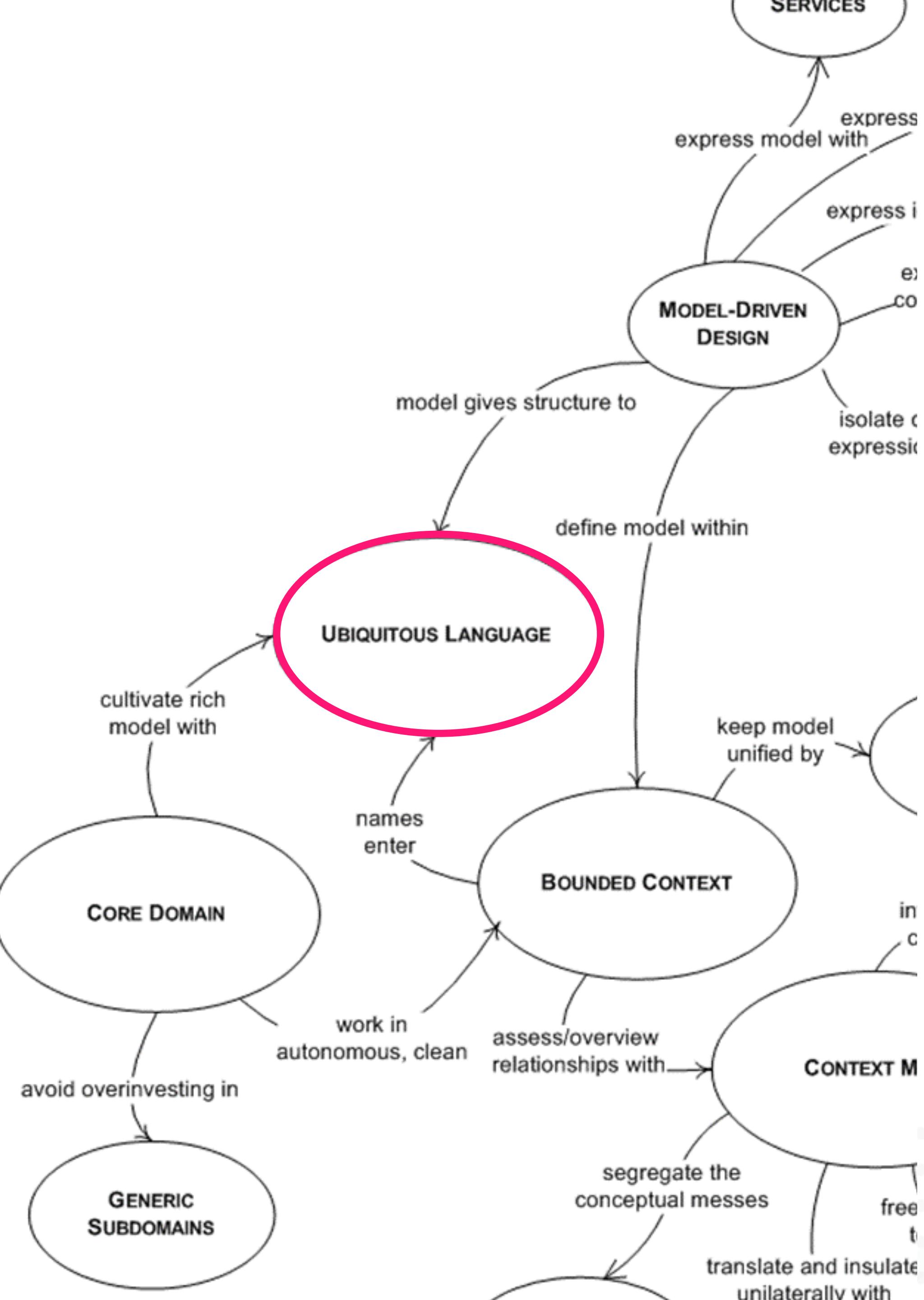


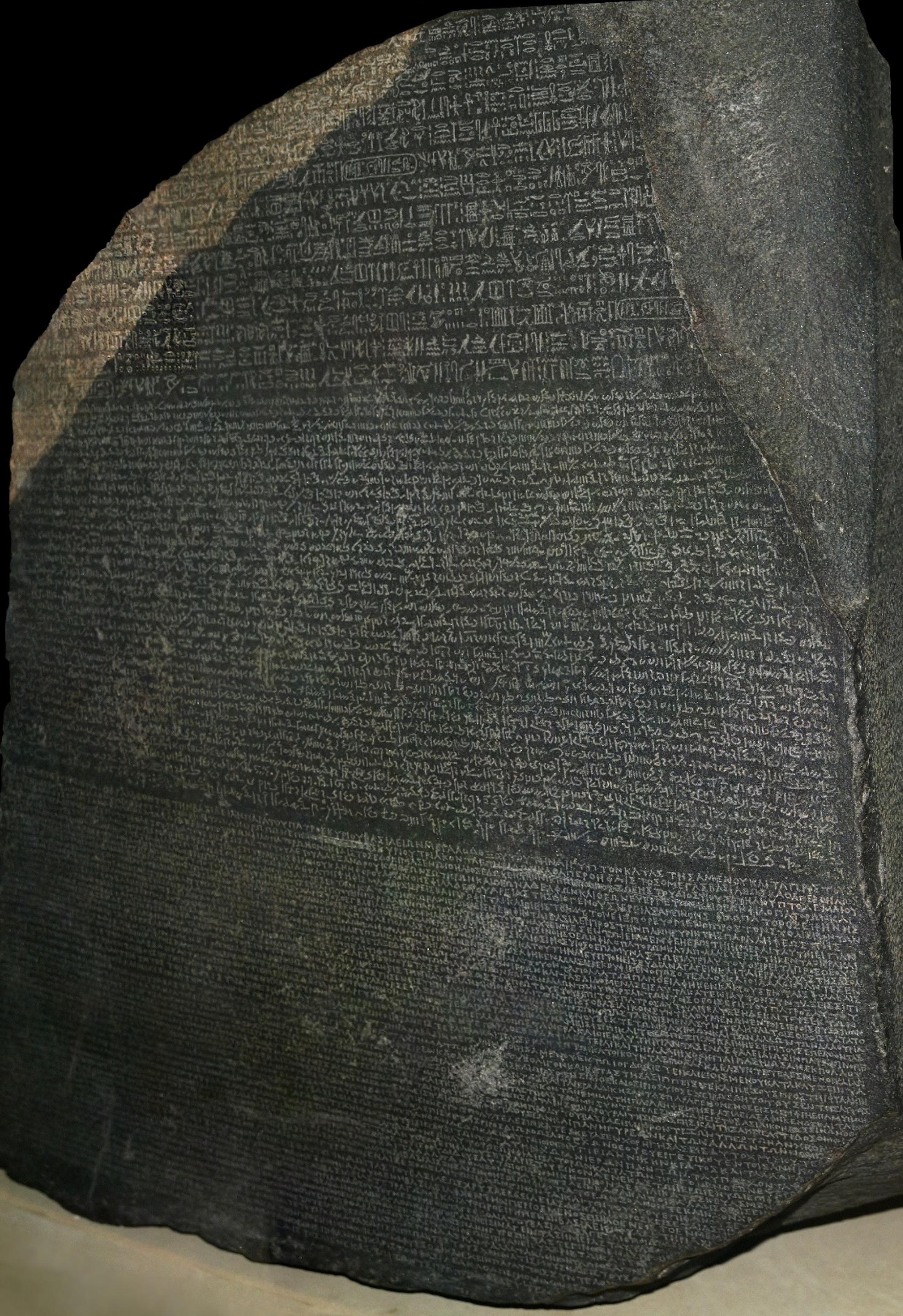
Understanding the Ubiquitous Language of a Bounded Context

**The language we use is key to
the shared understanding we
want to have with our domain
experts in order to be
successful.**

Ubiquitous Language

In the DDD Mind Map





We don't want to have to use a
Rosetta Stone!

Citation: By © Hans Hillewaert, CC BY-SA 4.0,
<https://commons.wikimedia.org/w/index.php?curid=3153928>

Context Maps

Appointment Scheduler

Client

Patient

Appointment

Notification

Exam Room

Team Awesome
Vet-app-sched



Shared Kernel

PersonName

Billing

Client

Procedure

Invoice

Notification

Team Ultimate
Vet-billing



Lost in Translation

English

As a veterinary technician,
I want to create an appointment for a patient.

Igbo

Dị ka ọkachamara ọgwugwọ ọriạ anụmanụ, achọro mo ịmepụta oge ọhụhu maka onye ọriạ.

As veterinarians, I want to create an appointment for the patient.

Dị ka ndị ọkachamara anụmanụ, Achọro m ịmepụta oge ọhụhu maka onye ọriạ.

According to animal experts, I want to create an appointment for the patient.

Pro Tip!

Try to explain back to the customer what you think they explained to you

**Avoid:
“What I meant was...”**

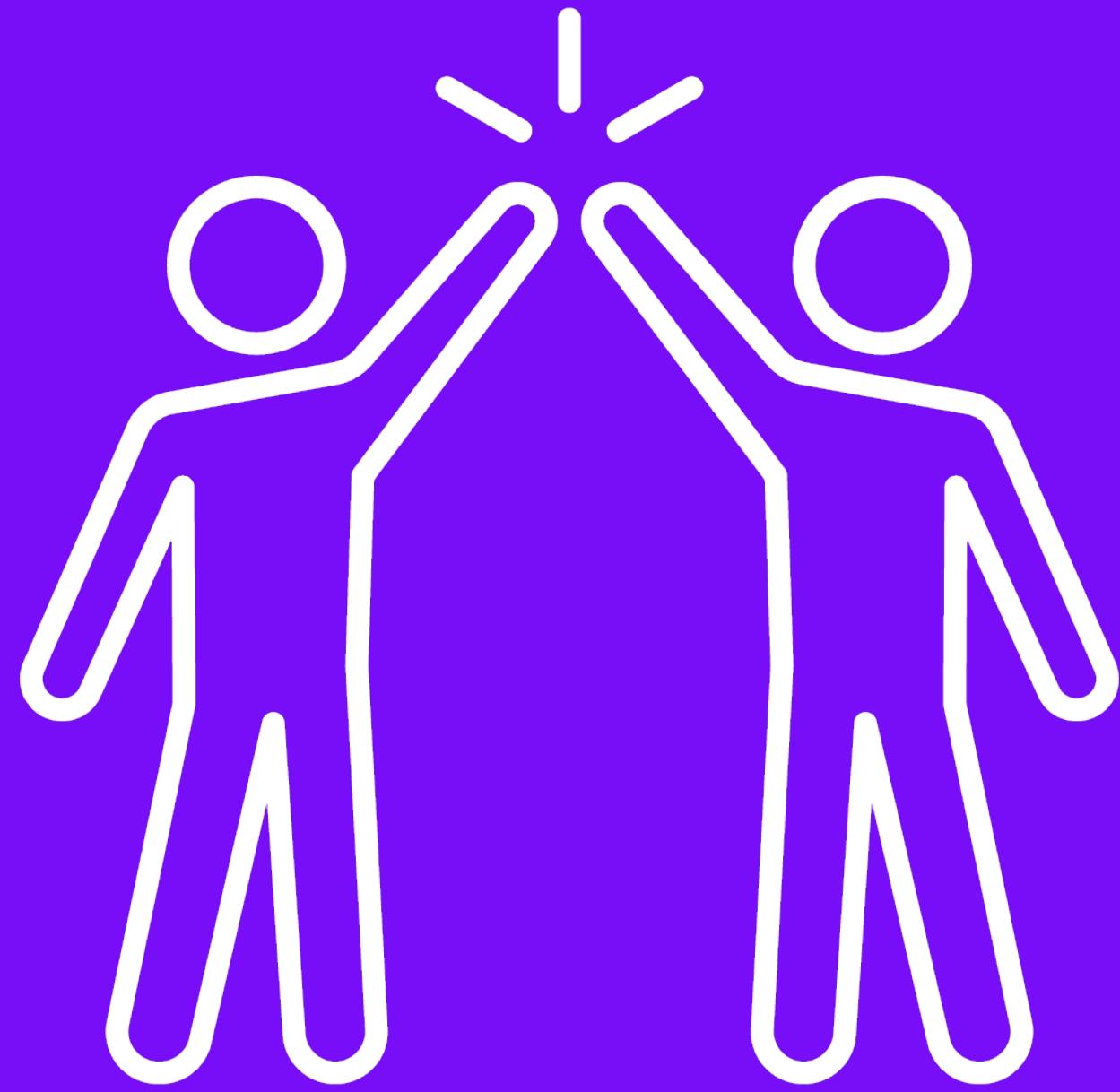
Eric Evans

A project faces serious
problems when its language
is fractured.

**A ubiquitous language applies
to a single bounded context and
is used throughout
conversations and code for that
context.**



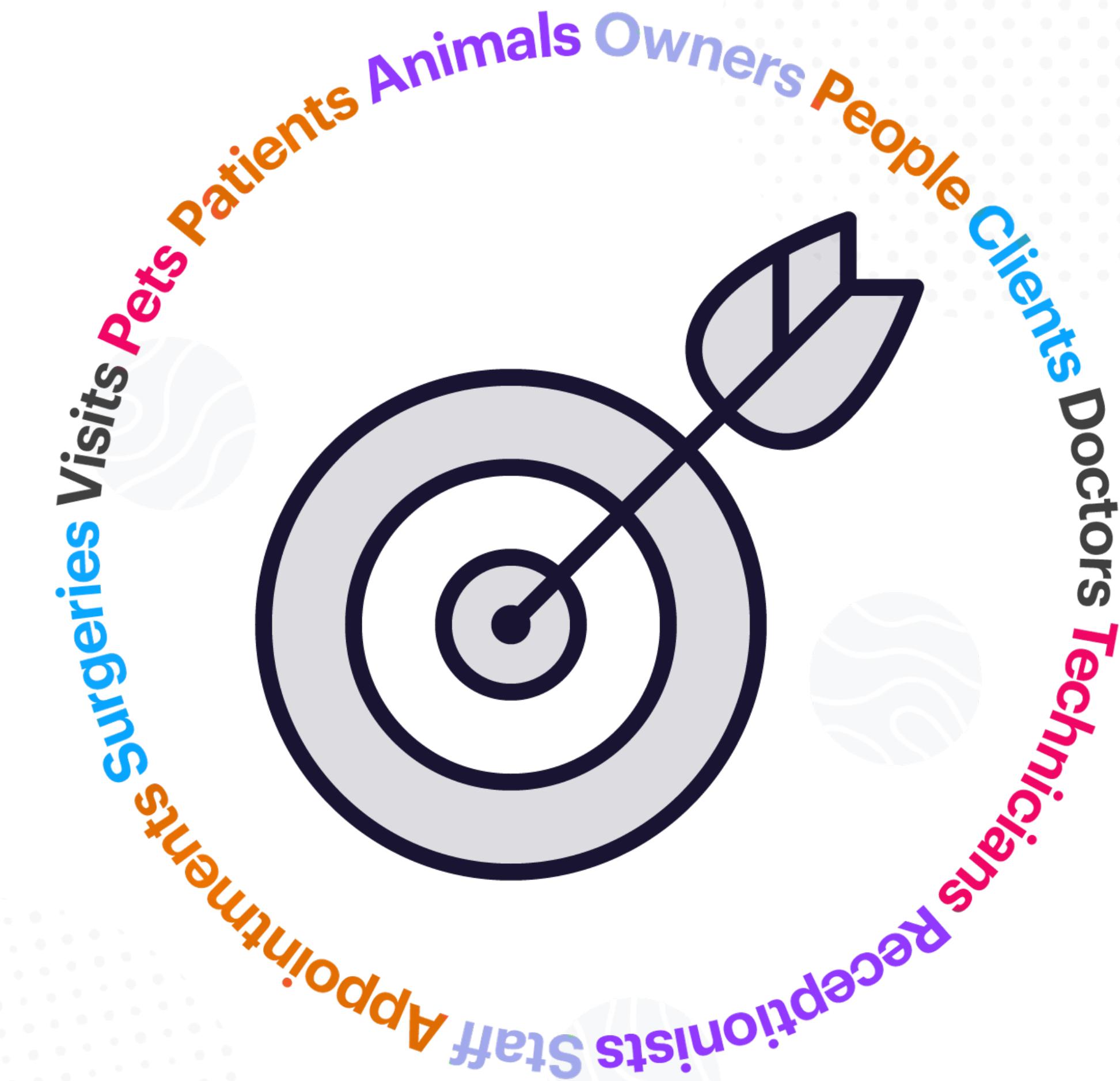
Conversation with a Domain Expert: Working on Our Ubiquitous Language



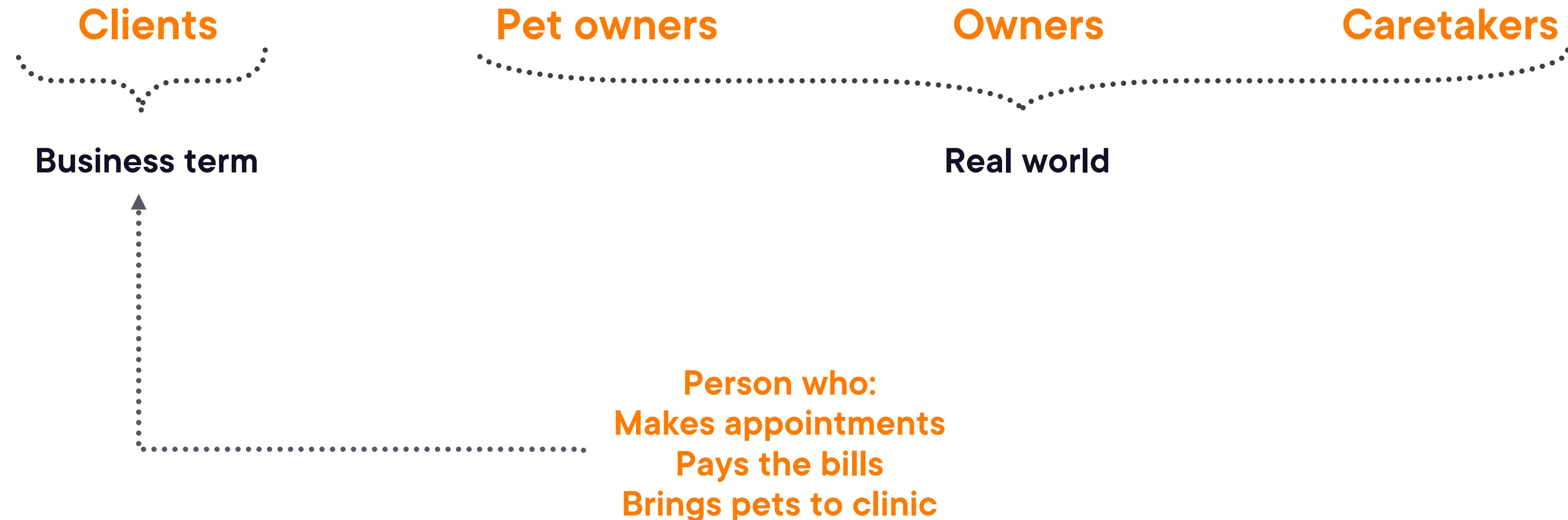
Equal Partners

Invest in your relationship with domain expert

Working on a Common, Ubiquitous Language for the Appointment Scheduling Bounded Context



Working on a Common, Ubiquitous Language for the Appointment Scheduling Bounded Context



Working on a Common, Ubiquitous Language for the Appointment Scheduling Bounded Context

Clients

Person who:
Makes appointments
Pays the bills
Brings pets to clinic

Pets

Clients

Patients

Animals

Working on a Common, Ubiquitous Language for the Appointment Scheduling Bounded Context

Clients

Person who:
Makes appointments
Pays the bills
Brings pets to clinic

Patients

Animal for whom:
Medical records are kept

Appointment

Surgery

Office visit

Wellness exam

Problem solving

Tech task

Working on a Common, Ubiquitous Language for the Appointment Scheduling Bounded Context

Clients

Person who:
Makes appointments
Pays the bills
Brings pets to clinic

Patients

Animal for whom:
Medical records are kept

Appointment

Scheduled time for:
Surgeries
Any type of regular office visit

Eric Evans

**Recognize that a change in
the ubiquitous language is a
change in the model.**

Reviewing Important Concepts from This Module

Problem Domain

The specific problem the software you're working on is trying to solve

Reviewing Important Concepts from This Module

Core Domain

The key differentiator for the customer's business -- something they must do well and cannot outsource

Subdomains

Separate applications or features your software must support or interact with

Bounded Context

A specific responsibility, with explicit boundaries that separate it from other parts of the system

Reviewing Important Concepts from This Module

Context Mapping

The process of identifying bounded contexts and their relationships to one another

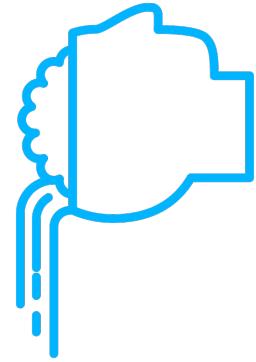
Shared Kernel

Part of the model that is shared by two or more teams, who agree not to change it without collaboration

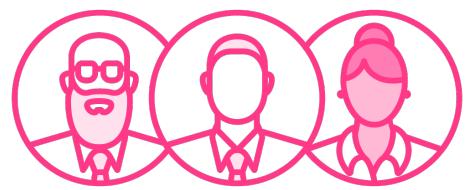
Ubiquitous Language

The language using terms from a domain model that programmers and domain experts use to discuss that particular sub-system

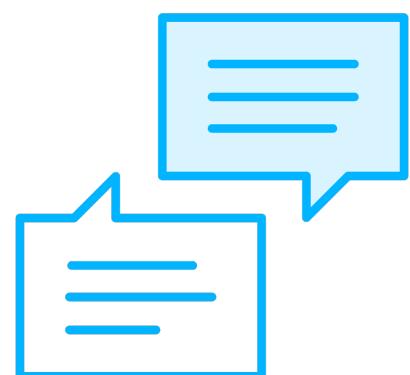
Julie and Steve Share Some More Thoughts About the Ubiquitous Language



Terms can seem overwhelming at first



But they are important to ensure a common understanding of the process



The terms are almost “the ubiquitous language” of DDD



Easier to convey meaning

The ubiquitous language of a bounded context is ubiquitous throughout everything you do in that context – discuss, model, code, etc.



In code, namespaces are helpful to quickly identify which bounded context you're working in



Review and Resources

Key Takeaways



- Introduced our domain: vet practice**
- Many discussions with domain expert**
- Identified core domain and subdomains**
- Identified bounded contexts**
- Addressed “shared” concepts**
- Began deriving a ubiquitous language**



**Thanks to Eric Evans for
his great advice and insights.**

Let's “drill into” the domain model.