model driven software engineering

domain specific languages

4DV651

domain specific languages

focuses on a particular aspect

small

improves productivity

internal and external dsl:s

Internal

External

internal dsl

fluent APIs

defined in a host language

method chaining

```
Author author = AUTHOR.as("author");
create.selectFrom(author)
.where(exists(selectOne() .from(BOOK)
.where(BOOK.STATUS.eq(BOOK_STATUS.SOLD_OUT))
.and(BOOK.AUTHOR_ID.eq(author.ID))));
```

jOOQ

external dsl

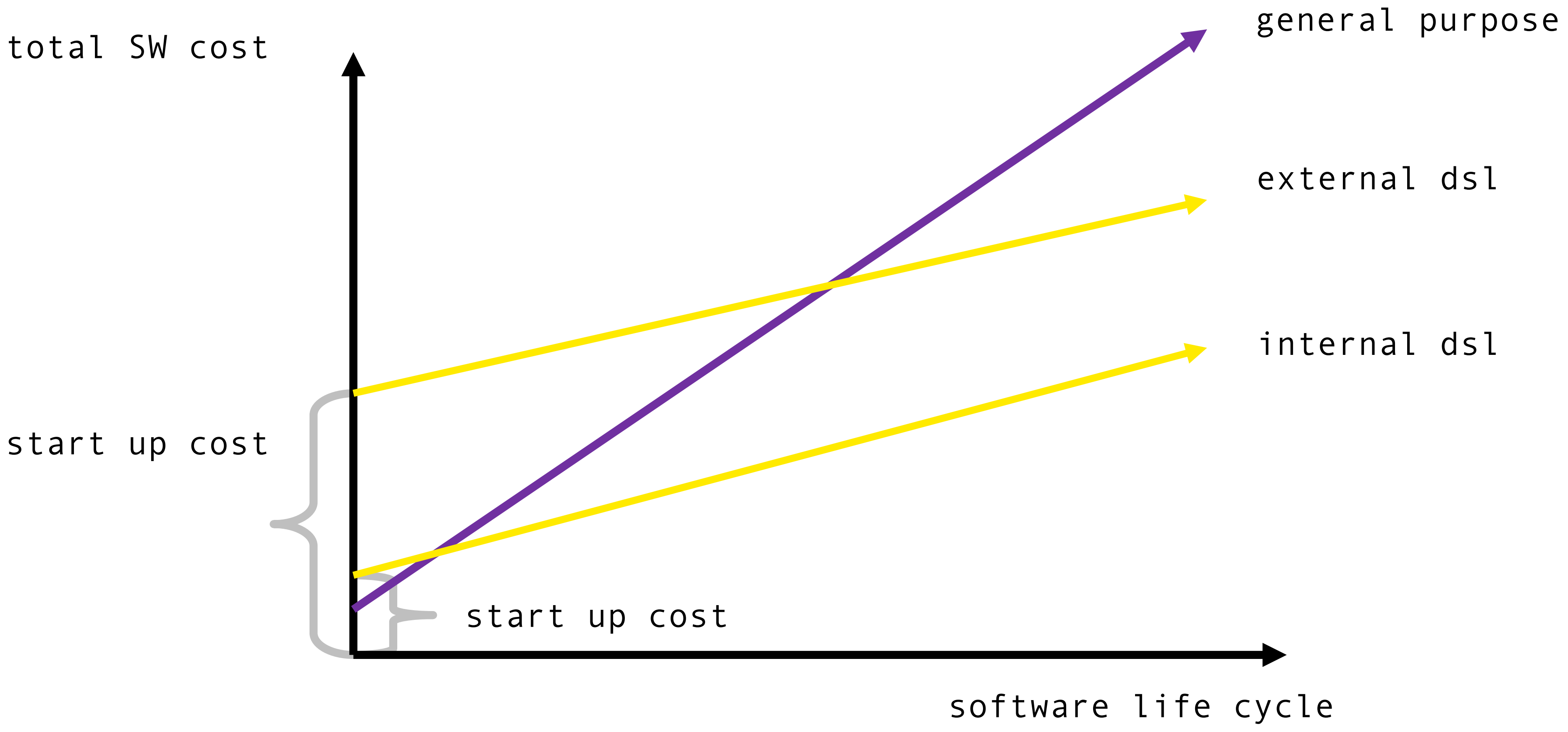
tailored for a particular application domain

captures precisely the semantics of the application domain -- no more, no less.

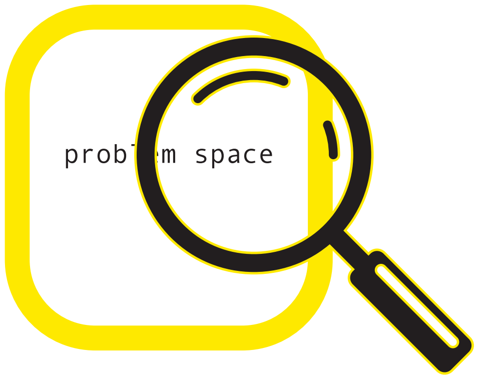
A programming language *A DSL allows one to develop software for a particular application domain quickly, and effectively, yielding programs that are easy to understand, reason about, and maintain.*

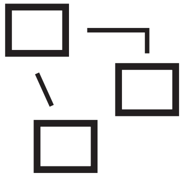
Hudak

the cost


total SW cost
general purpose
external dsl
internal dsl
start up cost
start up cost
software life cycle

domain driven design





Domain Driven Design (DDD) is an approach for building complex software applications that is centered on the development of **domain model**

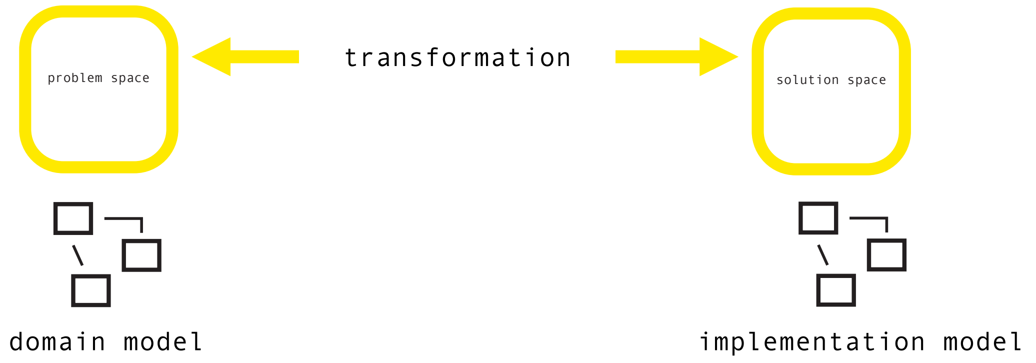
DDD in a nutshell

Domain Driven Design (DDD) is an approach for building complex software applications that is centered on the development of **domain model**

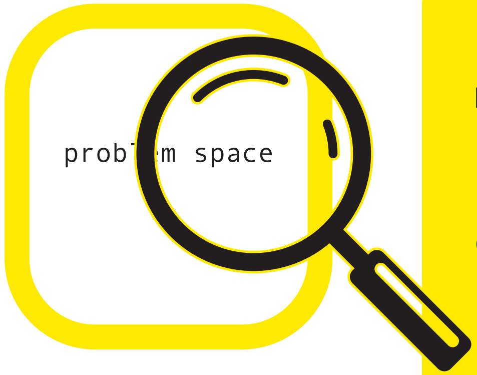
two useful concepts

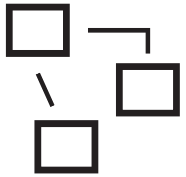
subdomains

bounded contexts



subdomains





domain model

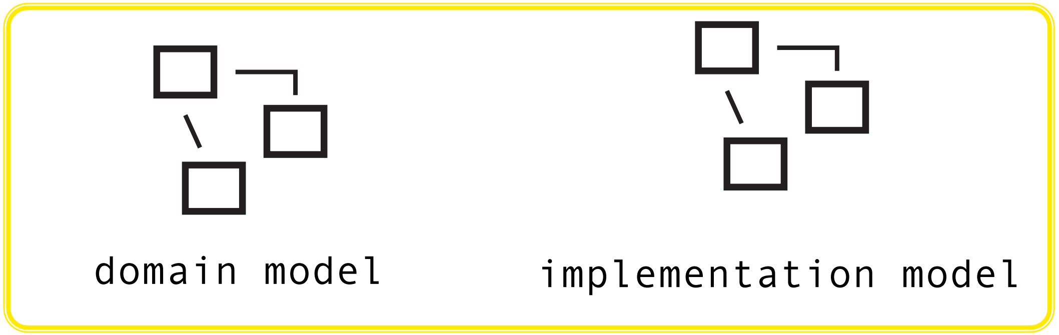
DDD defines a separate domain model for each subdomain

a subdomain is a part of the domain – the application's problem space

subdomains are identified by analyzing the **business** and identifying **areas of expertise**

e.g.: order taking, order management, restaurant order ticket management, delivery, and financials.

bounded context

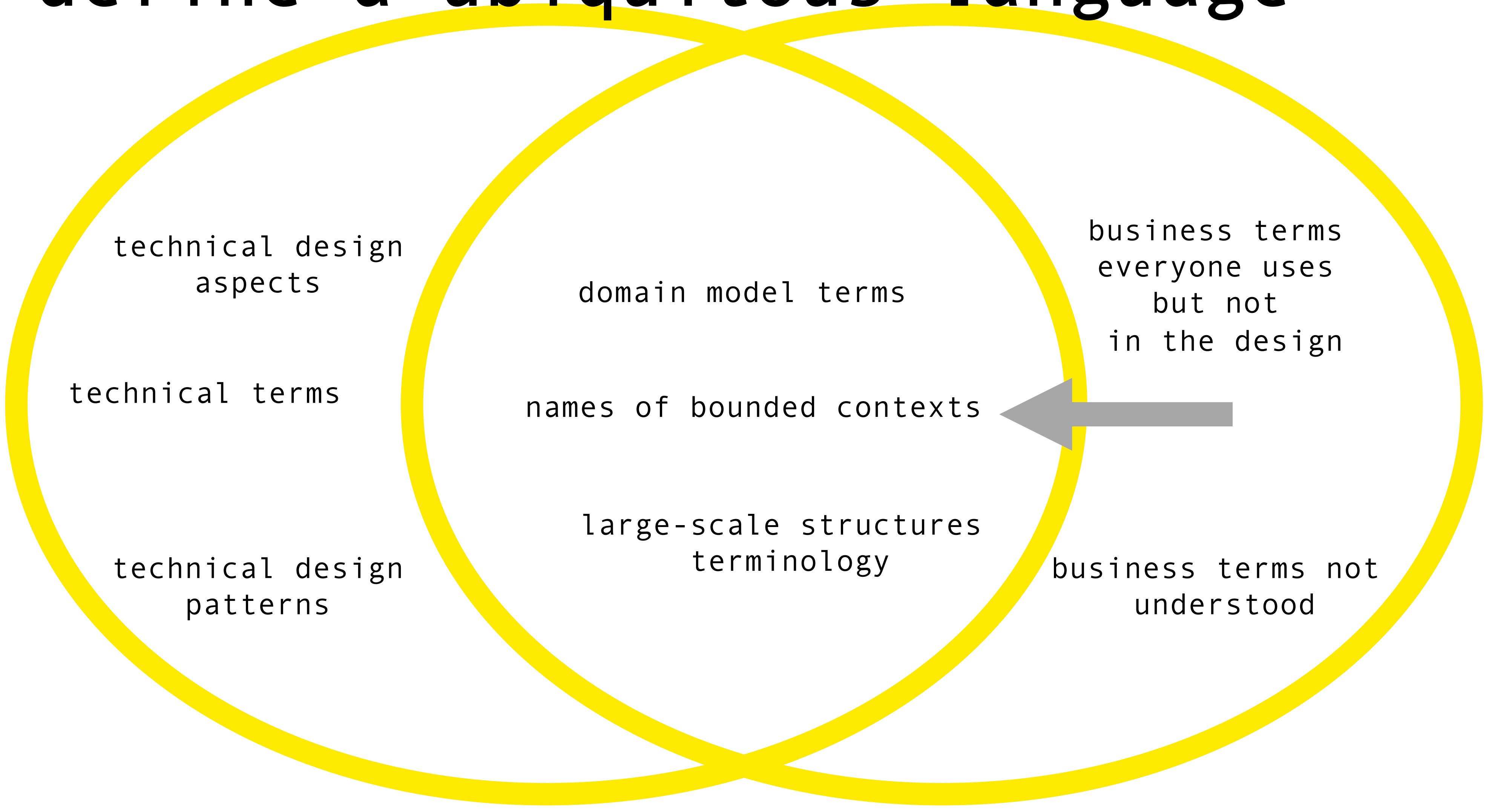


DDD calls the scope of a domain model a bounded context

a bounded context includes the solution space artifacts that implement the model

each bounded context may be considered as a service

define a ubiquitous language


technical design aspects
technical terms
technical design patterns
domain model terms
names of bounded contexts
large-scale structures terminology
business terms everyone uses but not in the design
business terms not understood

ubiquitous Language

to create a knowledge-rich design calls for a versatile, shared **team language**, and a lively experimentation with language that seldom happens on software projects

a UL carries **knowledge** in a dynamic form

- Discussion in the UL brings to life the meaning behind the diagrams and code

the vocabulary of UL includes

- **Names** of modules and prominent operations
- **Terms** to discuss rules that should be made explicit in the model

with a UL, the model is not just a design artifact, it becomes **integral** to everything the developers and domain experts do together