Discovering Domain Knowledge

In this chapter we will learn the domain-driven design tool for effective communication

and knowledge sharing: **the ubiquitous language:**

* Here we will use it to learn the intricacies of business domains.
* Later in the book we will use it to model and implement their business logic in software.

# Business Problems

Software systems are solutions to business problems. In the context of business domains, “problem” has a broader meaning than just a puzzle or a math problem. A business problem can be challenges associated with optimizing workflows and processes, minimizing manual labor, managing resources, supporting decisions, managing data, and so on.

Business problems appear both at the business domain and subdomain levels. **Subdomains are finer-grained problem domains whose goal is to provide solutions for specific business capabilities.**

A knowledge management subdomain optimizes the process of storing and retrieving information. A clearing subdomain optimizes the process of executing financial transactions. An accounting subdomain keeps track of the company’s funds.

# Knowledge Discovery

To design an effective software solution, we have to grasp at least **the basic knowledge of the business domain**. As we discussed earlier, this knowledge belongs to domain experts.

By no means should we, nor can we, become domain experts. That said, **it’s crucial for us to understand domain experts and to use the same business terminology they use.**

To be effective, **the software has to mimic the domain experts’ way of thinking about**

**the problem—their mental models.**

*The next two sentences are not quite clear for me at this point but we will probably understand them better later on:*

**Without an understanding of the business problem and the reasoning behind the requirements, our solutions will be limited to “translating” business requirements into source code.**

What if the requirements miss a crucial edge case? Or fail to describe a business concept, limiting our ability to implement a model that will support future requirements?

Effective knowledge sharing between domain experts and software engineers requires

effective communication.

# Communication

It’s safe to say that almost all software projects require the collaboration of stakeholders

in different roles: domain experts, product owners, engineers, UI and UX designers,

project managers, testers, analysts, and others. As in any collaborative effort, **the**

**outcome depends on how well all those parties can work together**:

* do all stakeholders agree on what problem is being solved?
* What about the solution they are building—do they hold any conflicting assumptions about its functional and nonfunctional requirements?

Agreement and alignment on all project-related matters are essential to a project’s success.

Page 23…