Table of Contents

Integrating with External Systems (15 pages)

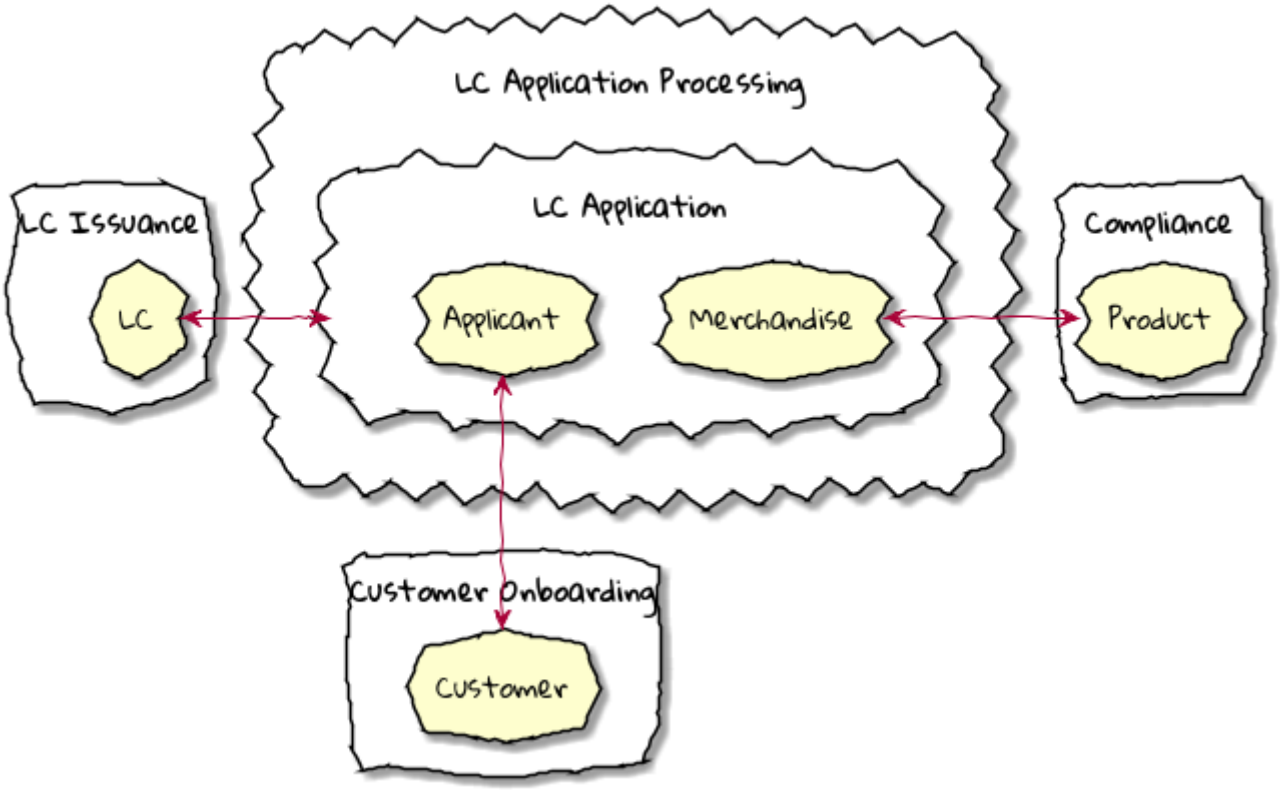
> Wholeness is not achieved by cutting off a portion of one's being, but by integration of the contraries.
>
> — Carl Jung

Thus far, we have used DDD to implement a robust core for our application. However, most bounded contexts usually have both upstream and downstream dependencies which usually change at a pace which is different from these core components. To maintain both agility and reliability and enable loose coupling, it is important to create what DDD calls the anti-corruption layer in order to shield the core from everything that surrounds it. In this chapter, we will look at integrating with a legacy Inventory Management system. We will round off by looking at common patterns when integrating with legacy applications.

Technical Requirements

Continuing our design journey

From our eventstorming session, we have arrived at four bounded contexts for our application as depicted here:



[lc application context map] | *lc-application-context-map.png*

Figure 1. A simple context map for the LC application

Integration mechanisms

Symmetric relationship patterns

Partnership

Shared kernel

Separate ways

Asymmetric relationship patterns

Conformist

Anti-corruption layer

Open host service

Implementation patterns

Data-based

API-based

HTTP-based APIs

Message-based APIs

Shared code artifacts

Enforcing contracts

Legacy Application Migration Patterns