Application Architecture

1. Spectrum of Change
2. Fundamental transformations
3. Dimensions to apply them along

Spectrum of Change

1. Data
2. Config
3. Code
4. Library
5. Language
6. Platform

Fundamental Transformations

1. Name
2. Abstract (generalize)
3. Instantiate (specialize)
4. Connect
5. Separate
6. Substitute
7. Aggregate

Name

Names Have Power

Name things to emphasize similarities or call out differences.

Example: P2P Lending

1. Commitment to fund part of a loan.
2. Piece of text that needs translation.

Example: P2P Lending

Common behavior: Limited time claim on a thing with a quantity.

Common data? URL, expiration, quantity available to reserve.

"Reservation"

Abstract

Emphasize one common aspect, hide others.

It's Not About Superclasses

Abstraction Example: Parity

- Elements: {0, 1}
- Operation: +

Abstraction Example: Seq

- Operations: first, rest

Instantiate

Create instances, customize via data

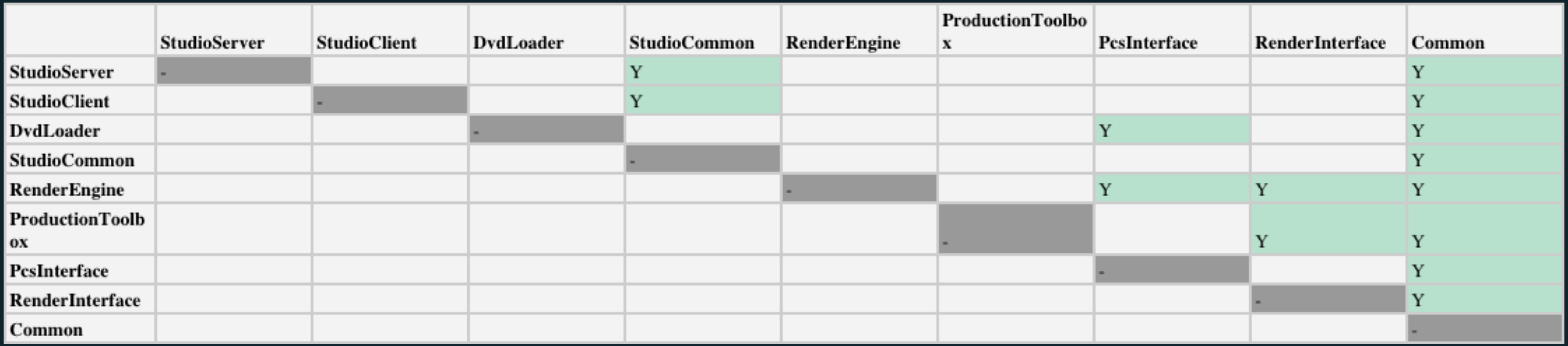
Instantiate

Counterexample: Many classes, one instance of each class.

Specialize via interactions

1. Instantiate "Reservations" as "Loan Funding Commitment"
2. No new behavior required, but useful for homogeneity of data.
3. Helps with analytics

Connect



	StudioServer	StudioClient	DvdLoader	StudioCommon	RenderEngine	ProductionToolbox	PcsInterface	RenderInterface	Common
StudioServer	-			Y					Y
StudioClient		-		Y					Y
DvdLoader			-				Y		Y
StudioCommon				-					Y
RenderEngine					-		Y	Y	Y
ProductionToolbox						-		Y	Y
PcsInterface							-		Y
RenderInterface								-	Y
Common									-

Adjacency Matrix

Separate

— Split functions to liberate from their original context
— Create different teams where you want a boundary

"Reverse Conway Maneuver"

Separating Lifecycle vs. Instance Data

```java
public interface Item {
    String getName(int version);
    String getDescription(int version);
    void publishItem(int version);
    int getLatestVersion();
    int[] getAllVersions();
    ...
}
```

Every consumer had to deal with the versioning & lifecycle.

Segregated Interface

```java
public interface ItemVersions {
    ItemDetails getLatestVersion();
    ItemDetails[] getAllVersions();
    void publishItem(ItemDetails details);
}

public interface ItemDetails {
    String getName();
    String getDescription();
    ...
}
```

Separating Lifecycle vs. Instance Data

- JSON API
  - Including status flags, effective date, etc.
- Most apps aren't involved in versioning and editing

Substitute

Replace a module, or insert a module between two others

Aggregate

Combine separate modules into one

Dimensions to Work With

1. Library
2. Executable
3. Process
4. Host
5. Service
6. Geography