Mikado + Kanban

Using Japanese to save your Legacy S/W

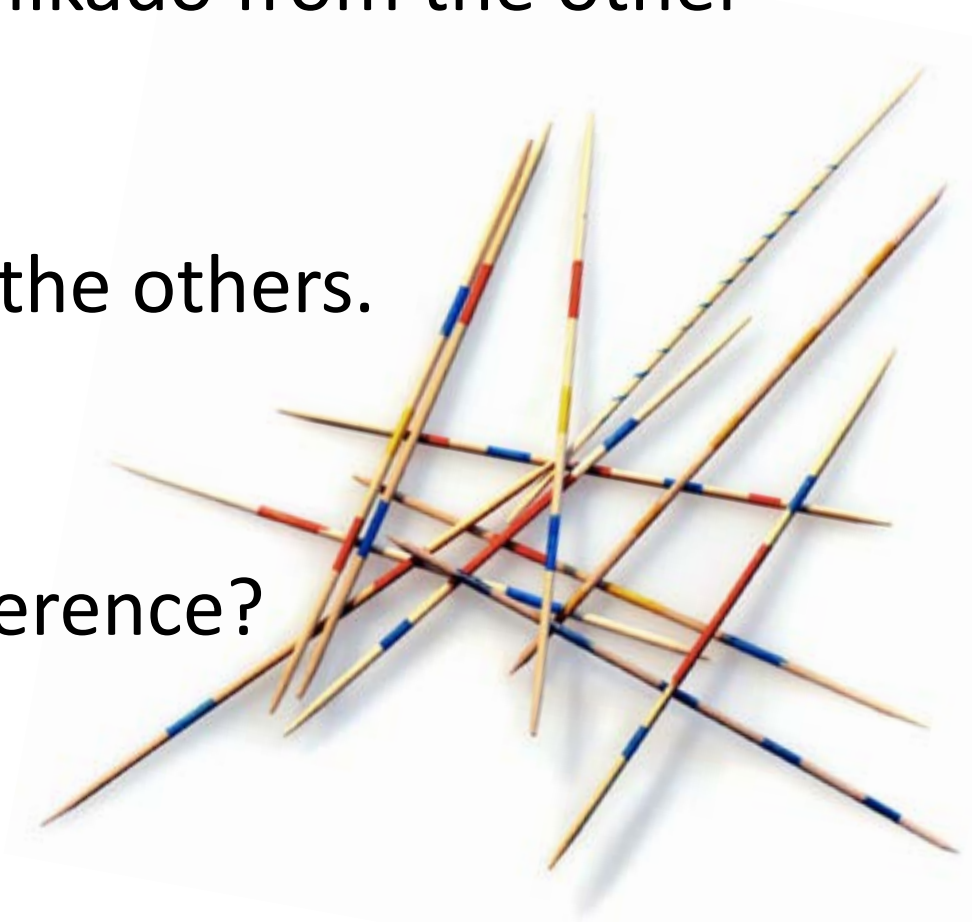
Mikado Method → Mikado Game

Making a change to your legacy code is not unlike picking out the Mikado from the other sticks in the pile.

Removing one impacts the others.

And not in a good way…

What is the biggest difference?

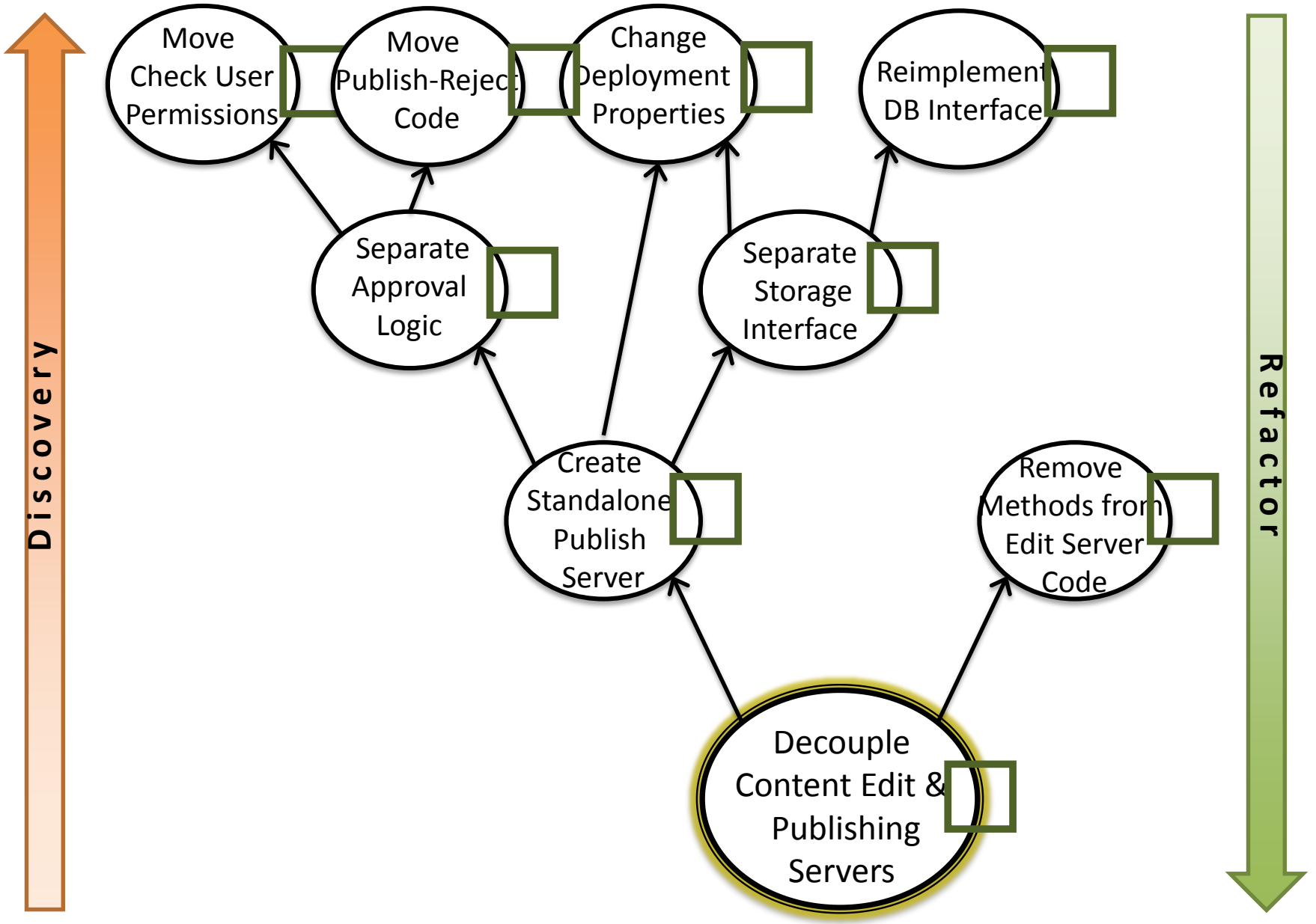


We can
Revert
our
code!

The Guts of the Mikado Method

- Make the ONE change you need/want to make
- See what broke – these are pre-requisites (the next set of changes to make before doing this one)
- Visualize (add to graph)
- **REVERT !!**

Repeat until no breakages occur.


Discovery
Refactor
Move Check User Permissions
Move Publish-Reject Code
Change Deployment Properties
Reimplement DB Interface
Separate Approval Logic
Separate Storage Interface
Create Standalone Publish Server
Remove Methods from Edit Server Code
Decouple Content Edit & Publishing Servers

For Defects…

- Need to use exploratory and regression testing to ID the next items to be graphed
- If not present, wrap unit tests around what you are touching
- Have testing occur in an environment that mimics as close to production as possible

Real Mikado Power

- Comes when you have unit, *integration*, and acceptance tests
- These help reveal necessary refactors by showing failures that the change creates
- Safer than exploration to find breakages

Some Restructuring Considerations

- Refactor Before Adding Functionality
- Separate Bulky Interfaces (limiting the consumers it serves)
- Extract Methods/Classes into Common Libraries

Let's look at the first one…

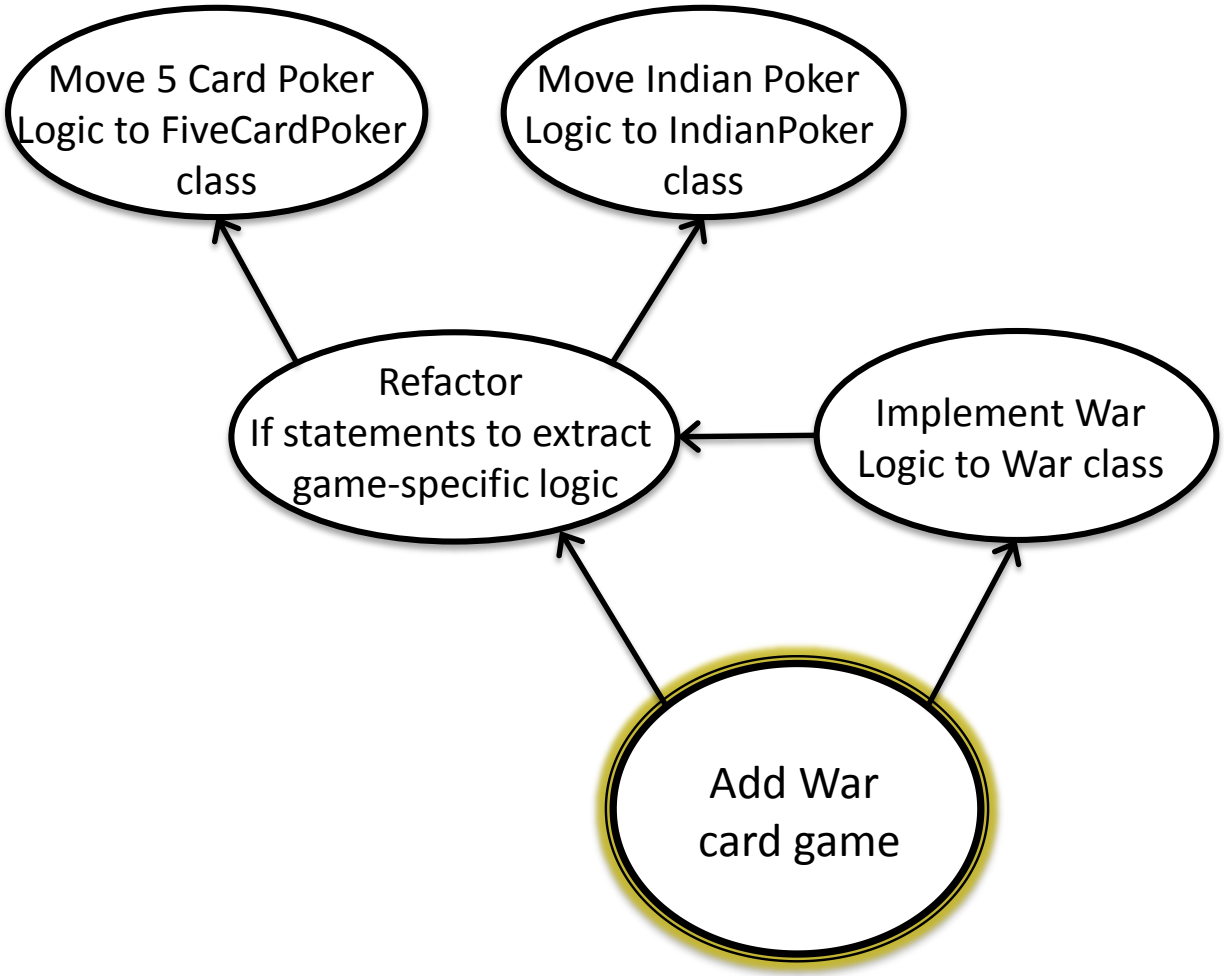
Refactor Before Adding Functionality

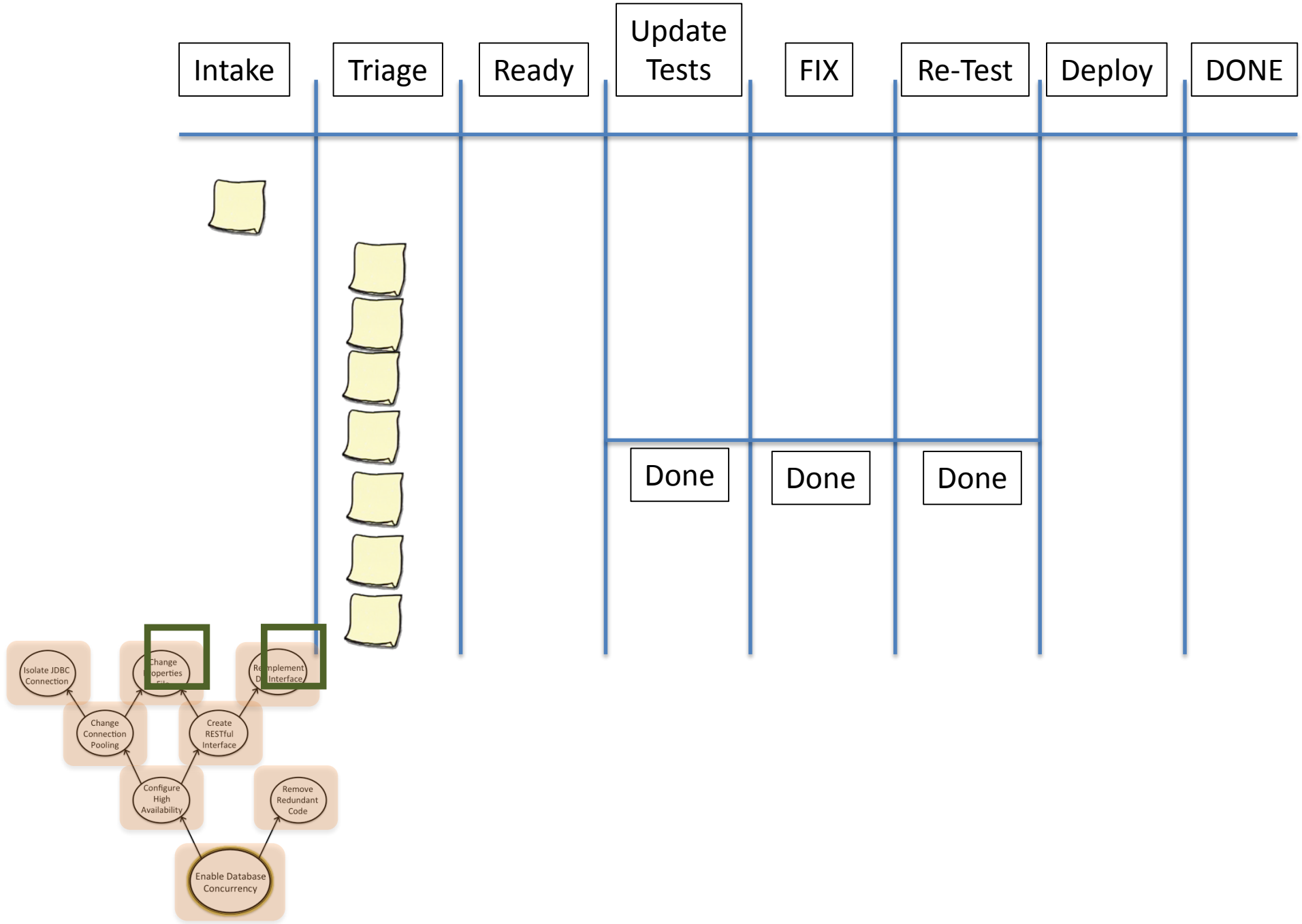
- Suppose -

```
. . .
public static final int FIVE_CARD_POKER = 0;
public static final int INDIAN_POKER = 1;
. . .
Public class CardGameEngine {
        . . .
        public void deal (Game game, User user, Dealer dealer) {
                if (game.getType() == FIVE_CARD_POKER {
                        user.setCards(dealer.deal(5));
                } else if (game.getType() == INDIAN_POKER {
                        user.setCards(dealer.deal(1));
                } else {
                        // else what?
                }
}
```

- Want to add War (each player is dealt 26 cards)
- Could add it to the If-Then-Else logic, but this will get messy, especially if I wanted to add more…

Mikado Graph for Refactor




Intake
Triage
Ready
Update Tests
FIX
Re-Test
Deploy
DONE
Done
Done
Done
Isolate JDBC Connection
Change Connection Pooling
Create RESTful Interface
Configure High Availability
Remove Redundant Code
Enable Database Concurrency

YOU

THANK

ありがとうございました