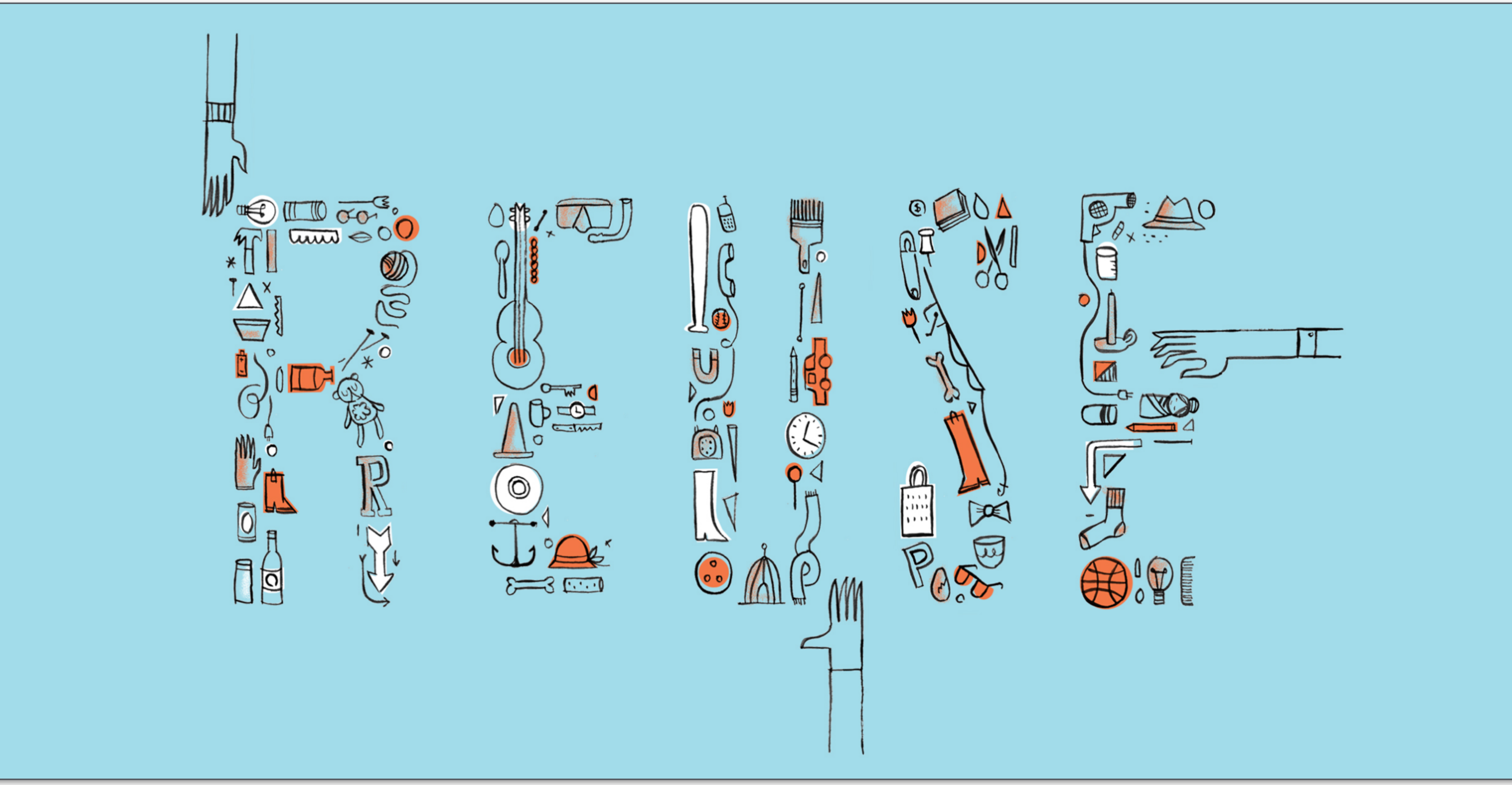


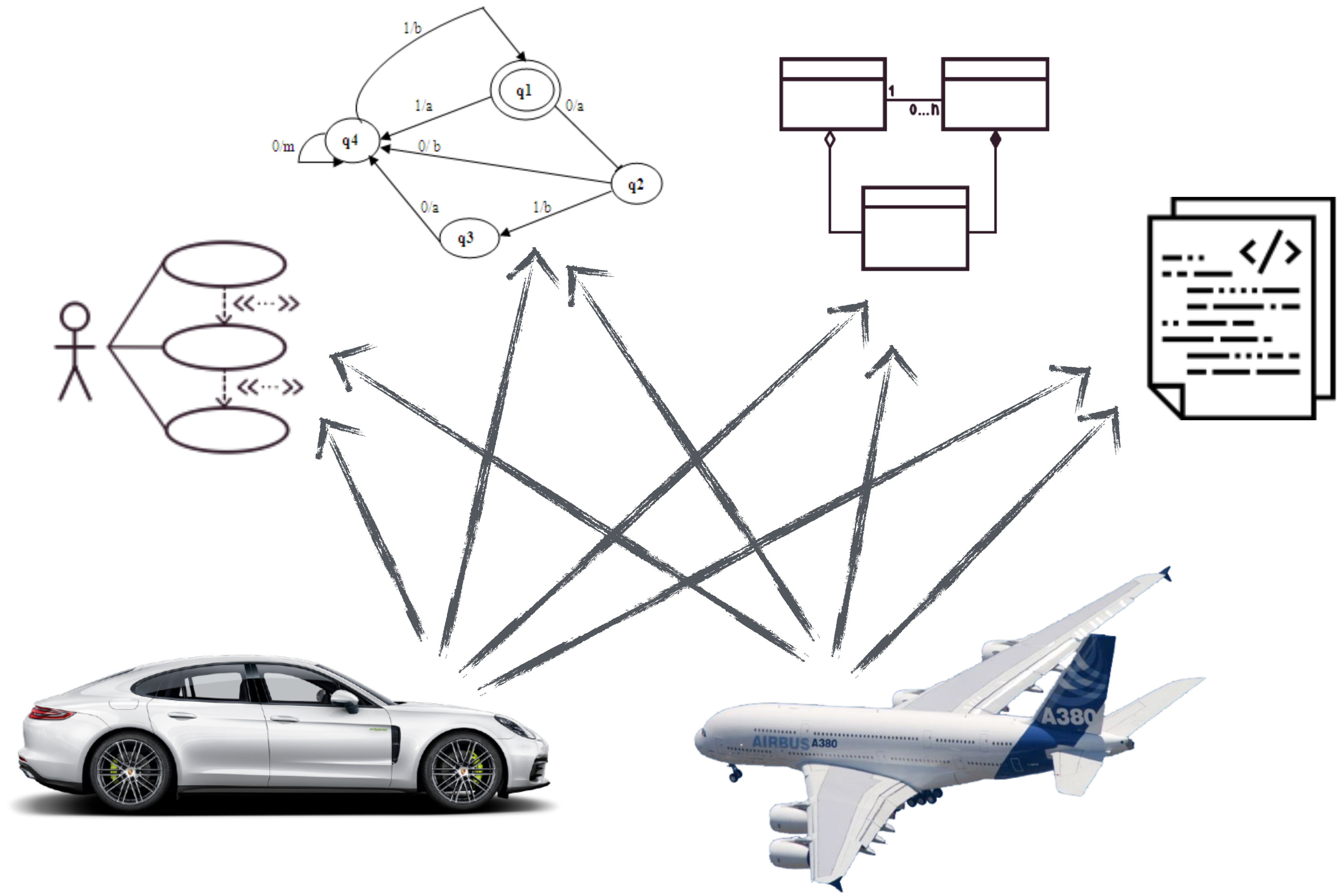
Model-Based Reuse of Framework APIs

Bridging the Gap Between Models and Code

Matthias Schöttle

McGill UNIVERSITY

Model-Driven Engineering



Reuse

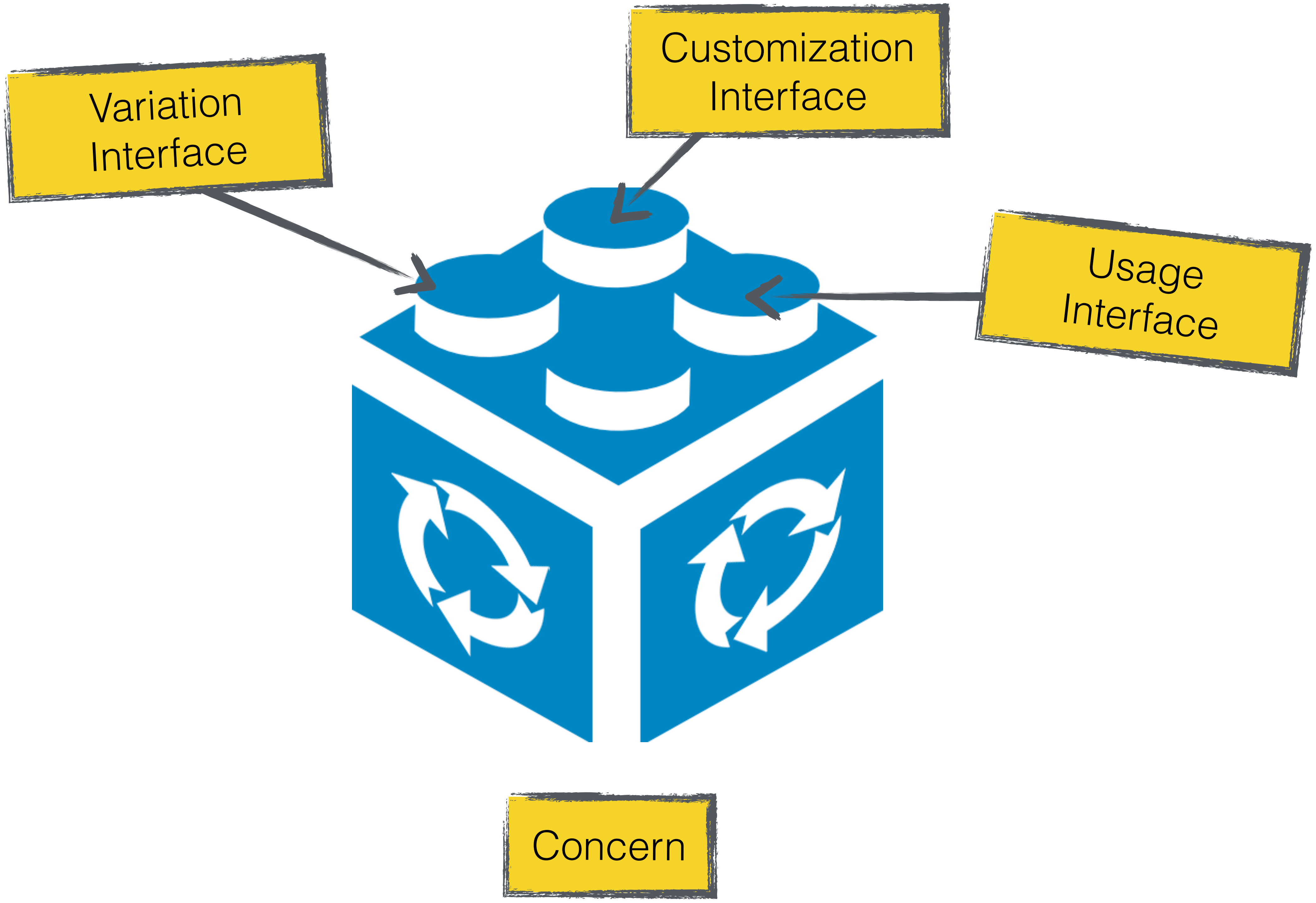
Concern-Oriented Reuse

Concern — Unit Of Reuse

Concern

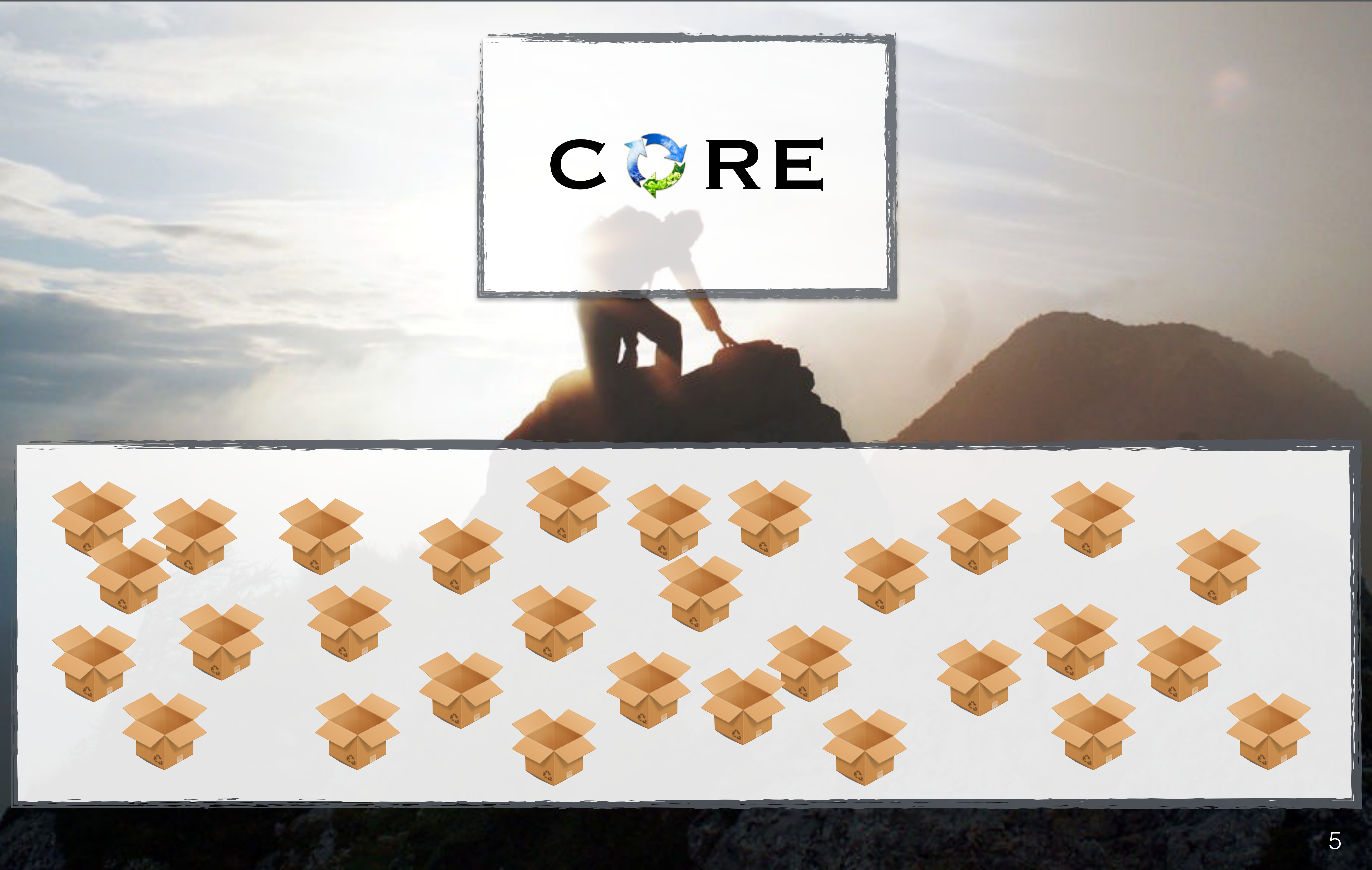
Concern-Oriented Reuse

Concern — Unit Of Reuse

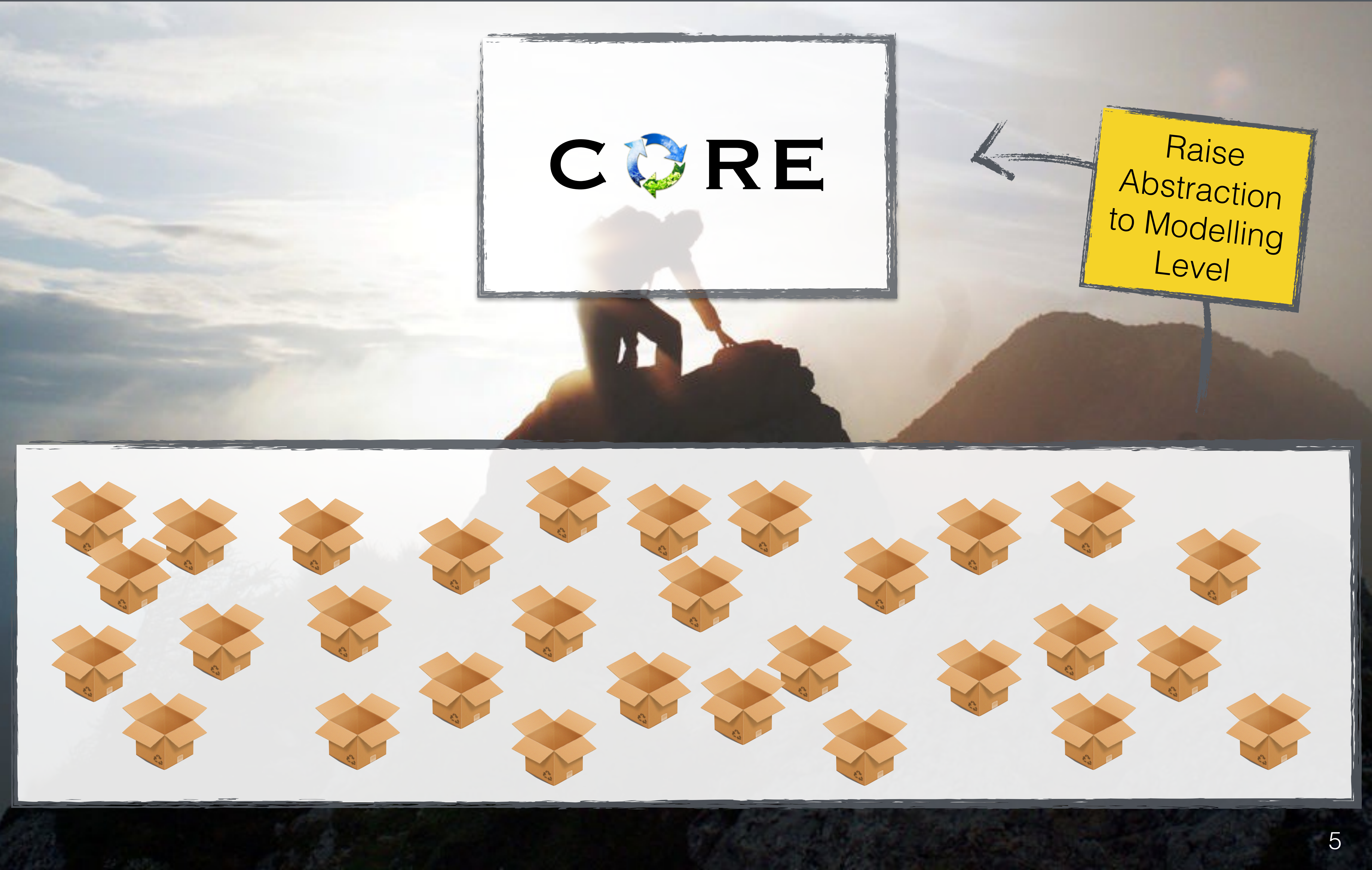


Bridging the Gap

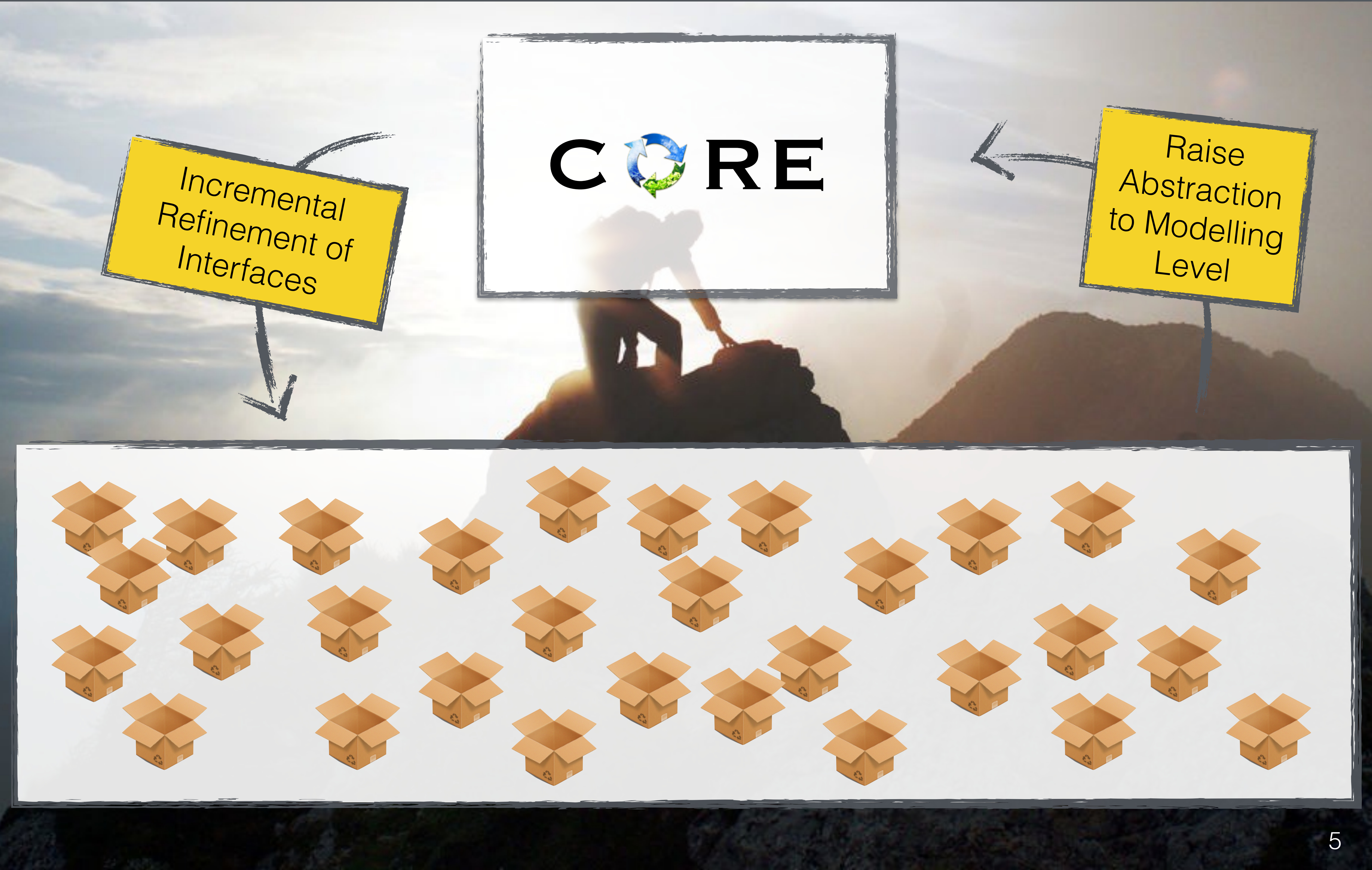
Bridging the Gap



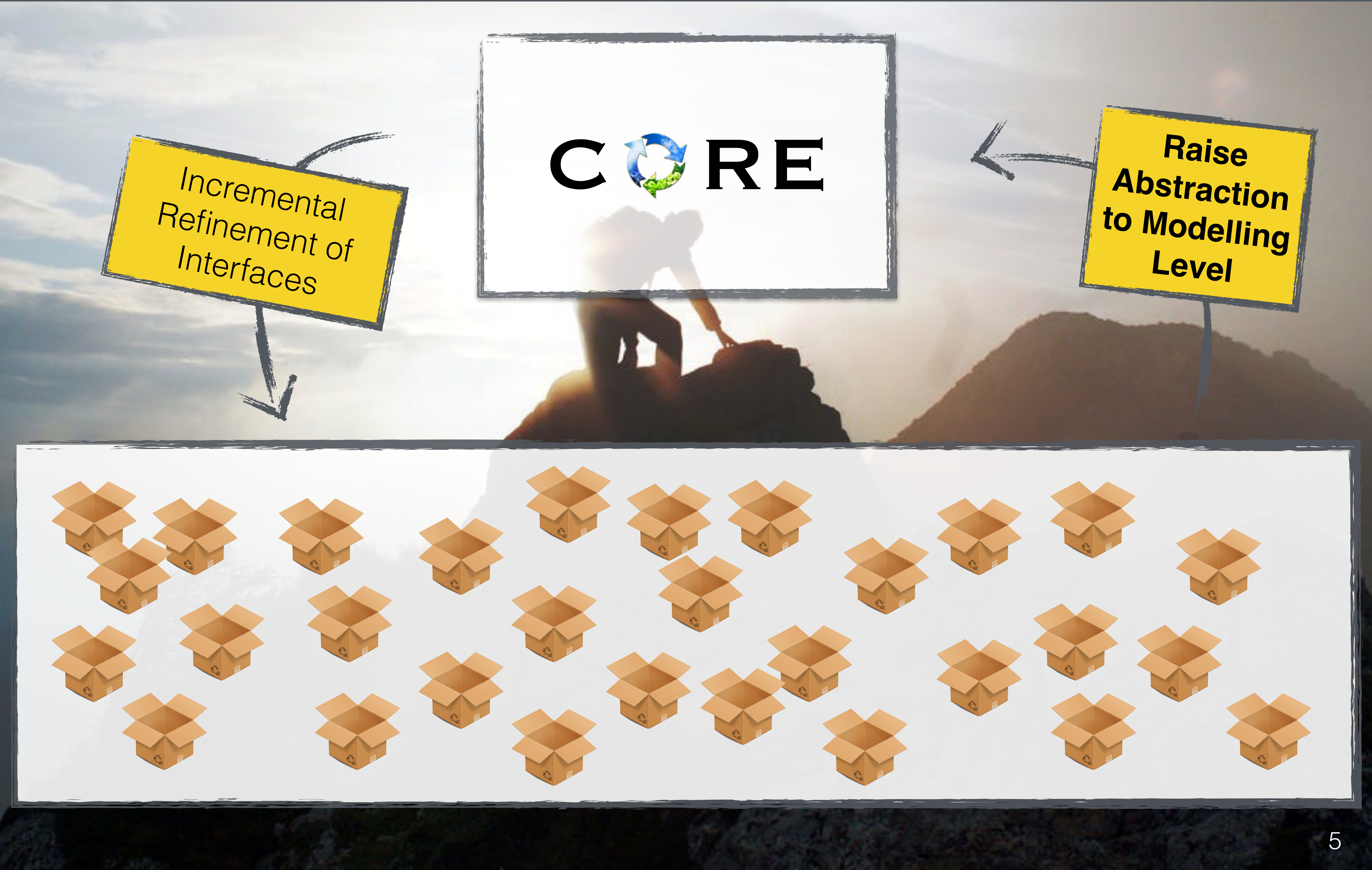
Bridging the Gap



Bridging the Gap


CORE
Incremental Refinement of Interfaces
Raise Abstraction to Modelling Level

Bridging the Gap


CORE
Incremental Refinement of Interfaces
Raise Abstraction to Modelling Level

The Variation Interface

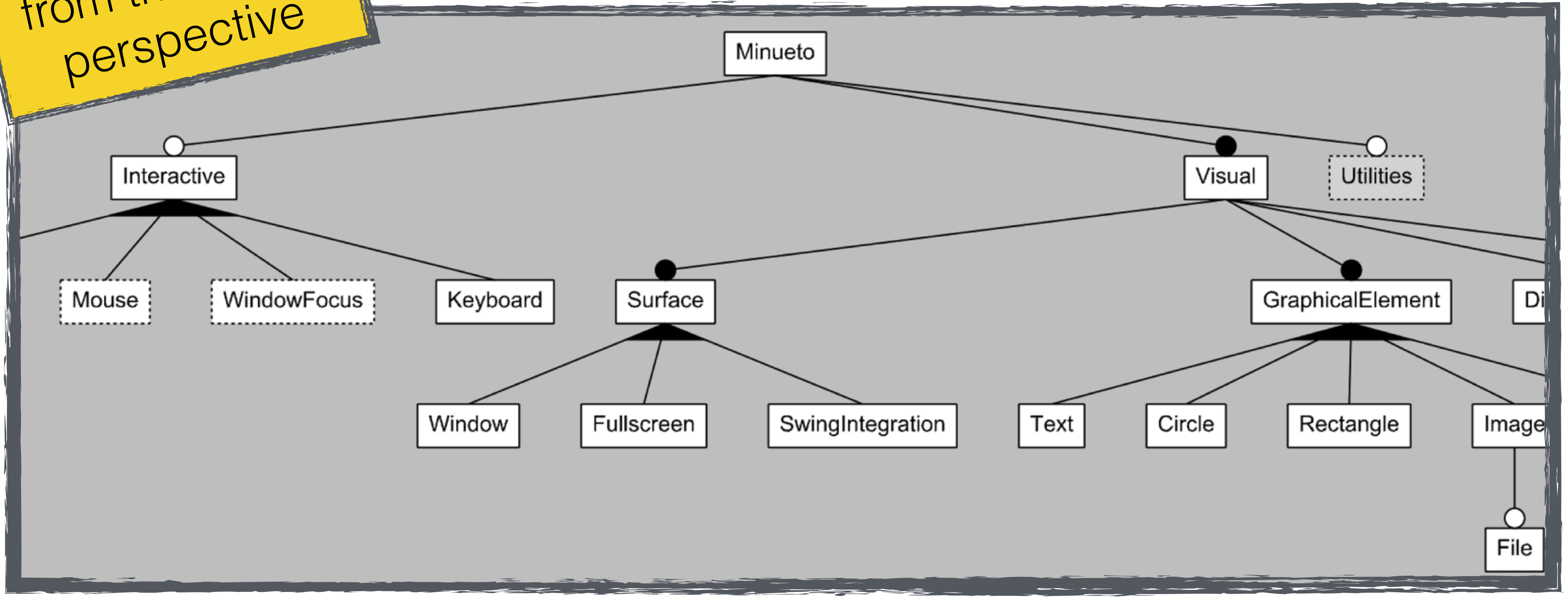
Documents and Organizes Features

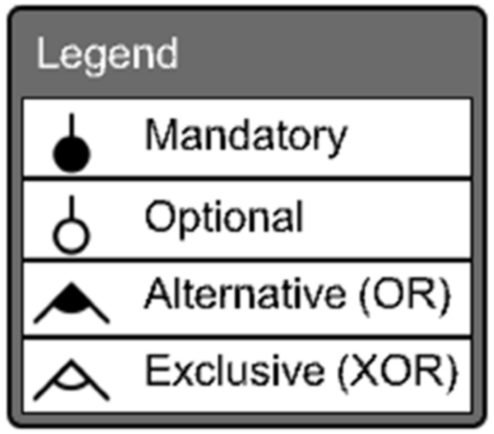
Feature Model of API

The Variation Interface

Documents and Organizes Features

from the user's perspective

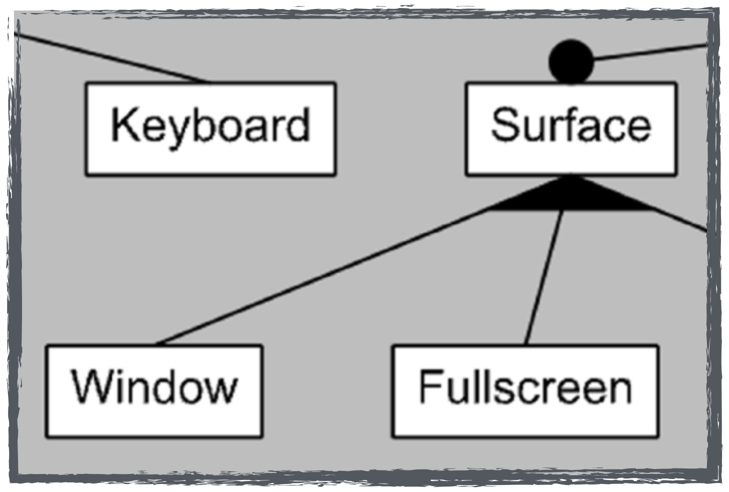




Feature Model of API

The Usage Interface

Tailors the API to the User's Need

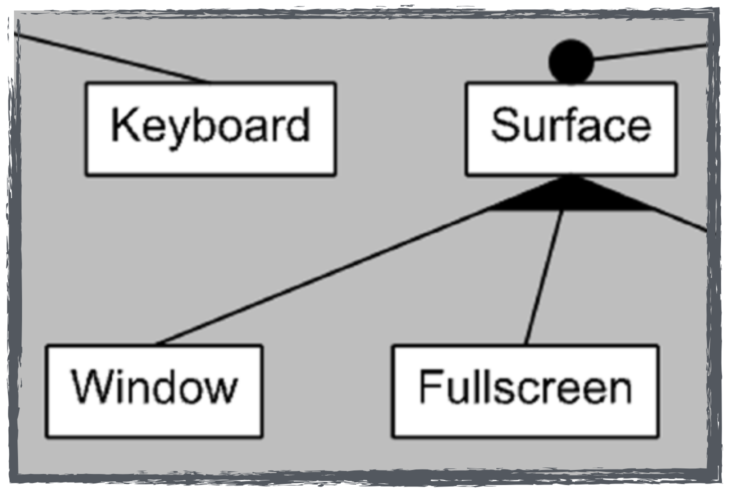


API split across features

Can be split at operation level

The Usage Interface

Tailors the API to the User's Need



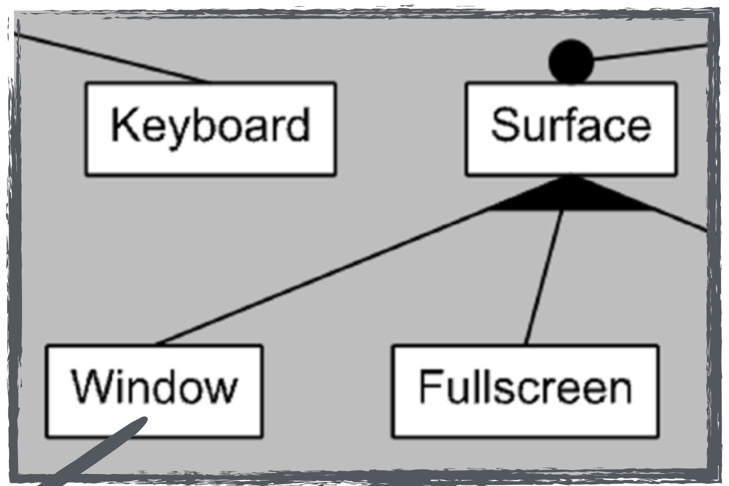
public classes and methods

API split across features

Can be split at operation level

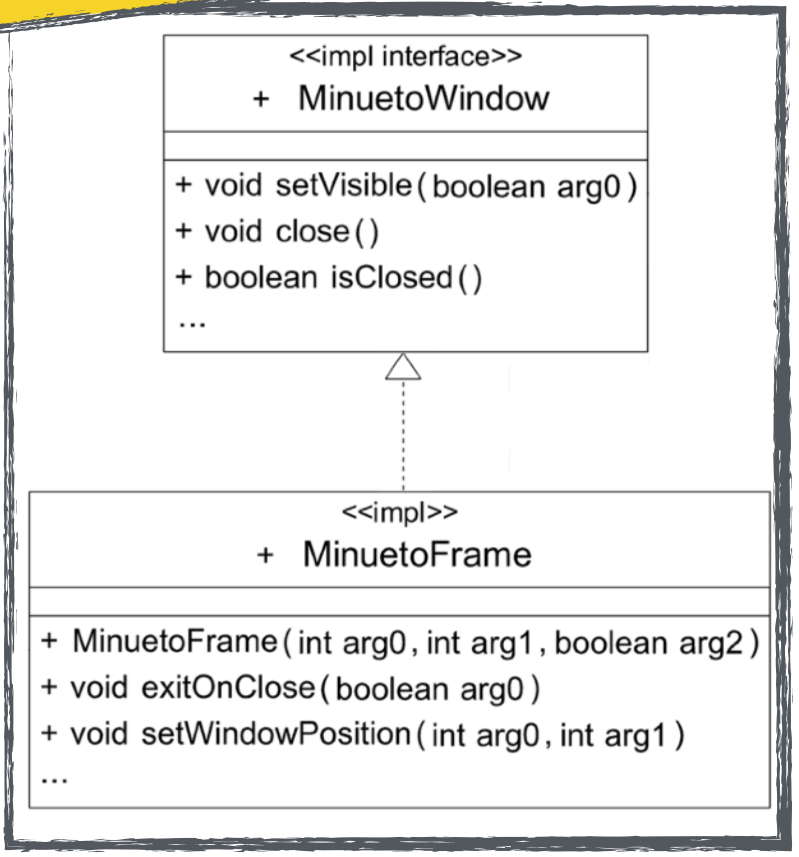
The Usage Interface

Tailors the API to the User's Need



public classes and methods

design model



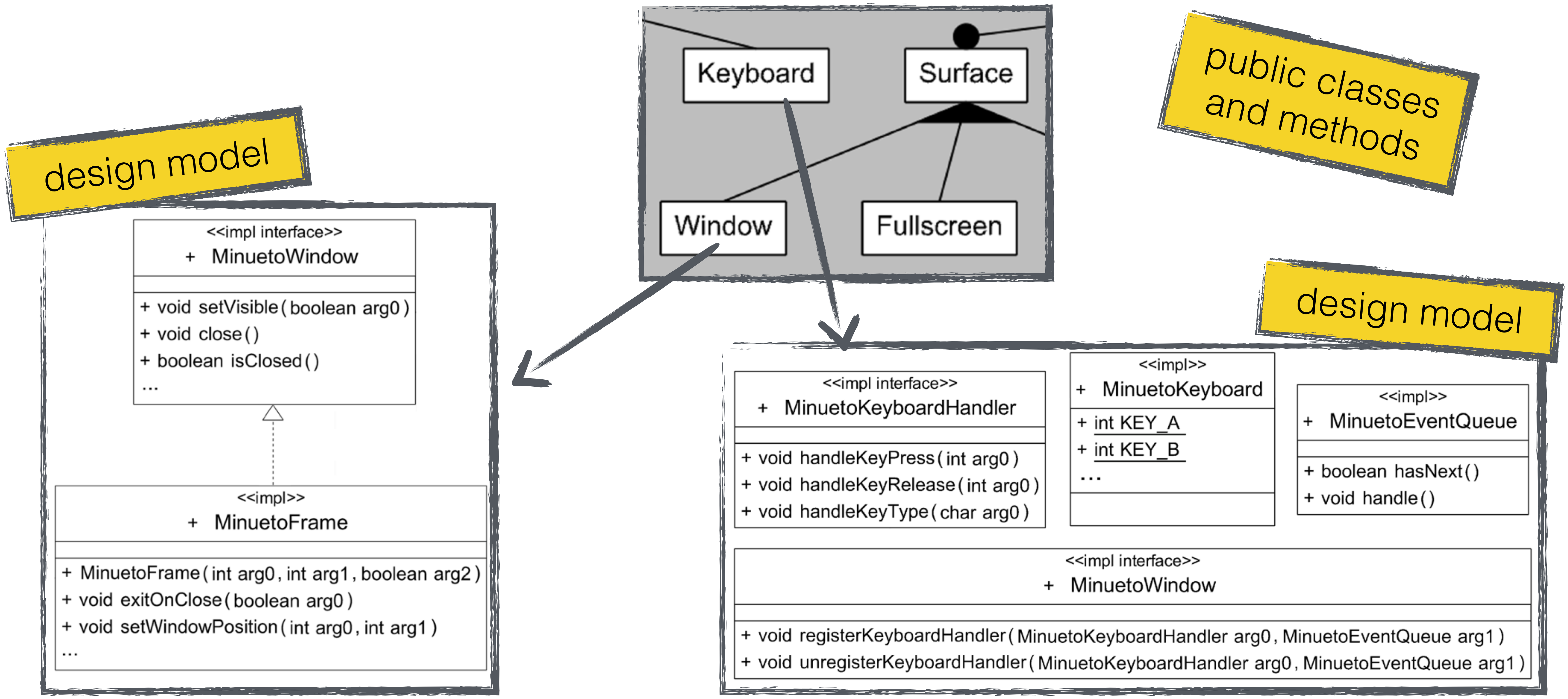
```
<<impl interface>>
+ MinuetoWindow
+ void setVisible(boolean arg0)
+ void close()
+ boolean isClosed()
...

<<impl>>
+ MinuetoFrame
+ MinuetoFrame(int arg0, int arg1, boolean arg2)
+ void exitOnClose(boolean arg0)
+ void setWindowPosition(int arg0, int arg1)
...
```

API split across features

Can be split at operation level

The Usage Interface

Tailors the API to the User's Need

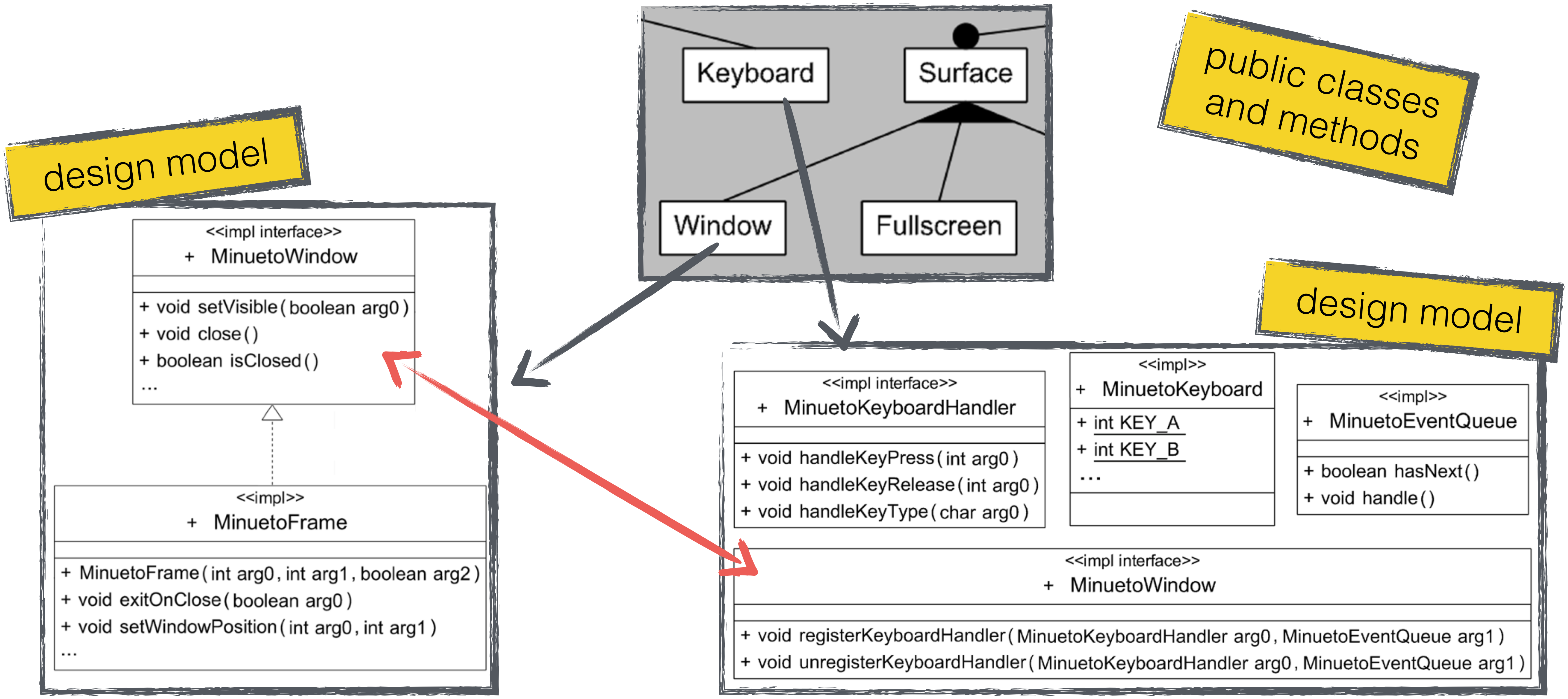

Keyboard
Surface
Window
Fullscreen
public classes and methods
design model
<<impl interface>>
+ MinuetoWindow
+ void setVisible(boolean arg0)
+ void close()
+ boolean isClosed()
...
<<impl>>
+ MinuetoFrame
+ MinuetoFrame(int arg0, int arg1, boolean arg2)
+ void exitOnClose(boolean arg0)
+ void setWindowPosition(int arg0, int arg1)
...
design model
<<impl interface>>
+ MinuetoKeyboardHandler
+ void handleKeyPress(int arg0)
+ void handleKeyRelease(int arg0)
+ void handleKeyType(char arg0)
<<impl>>
+ MinuetoKeyboard
+ int KEY_A
+ int KEY_B
...
<<impl>>
+ MinuetoEventQueue
+ boolean hasNext()
+ void handle()
<<impl interface>>
+ MinuetoWindow
+ void registerKeyboardHandler(MinuetoKeyboardHandler arg0, MinuetoEventQueue arg1)
+ void unregisterKeyboardHandler(MinuetoKeyboardHandler arg0, MinuetoEventQueue arg1)

API split across features

Can be split at operation level

The Usage Interface

Tailors the API to the User's Need



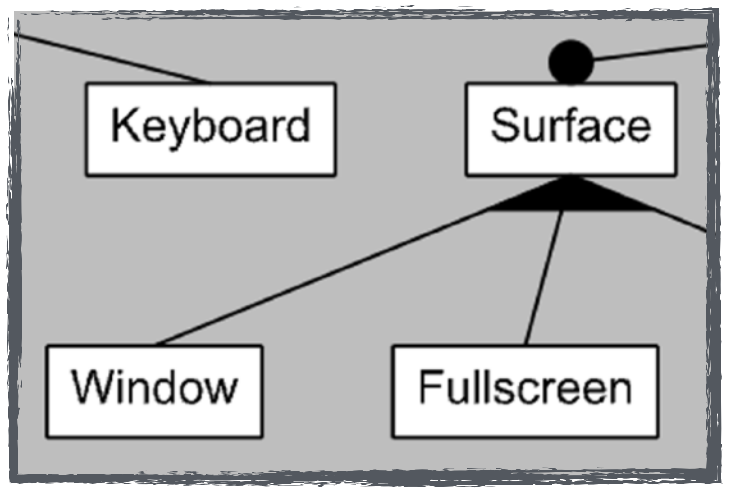
public classes and methods
design model
design model
Keyboard
Surface
Window
Fullscreen
<<impl interface>>
+ MinuetoWindow
+ void setVisible(boolean arg0)
+ void close()
+ boolean isClosed()
...
<<impl>>
+ MinuetoFrame
+ MinuetoFrame(int arg0, int arg1, boolean arg2)
+ void exitOnClose(boolean arg0)
+ void setWindowPosition(int arg0, int arg1)
...
<<impl interface>>
+ MinuetoKeyboardHandler
+ void handleKeyPress(int arg0)
+ void handleKeyRelease(int arg0)
+ void handleKeyType(char arg0)
<<impl>>
+ MinuetoKeyboard
+ int KEY_A
+ int KEY_B
...
<<impl>>
+ MinuetoEventQueue
+ boolean hasNext()
+ void handle()
<<impl interface>>
+ MinuetoWindow
+ void registerKeyboardHandler(MinuetoKeyboardHandler arg0, MinuetoEventQueue arg1)
+ void unregisterKeyboardHandler(MinuetoKeyboardHandler arg0, MinuetoEventQueue arg1)

API split across features

Can be split at operation level

The Customization Interface

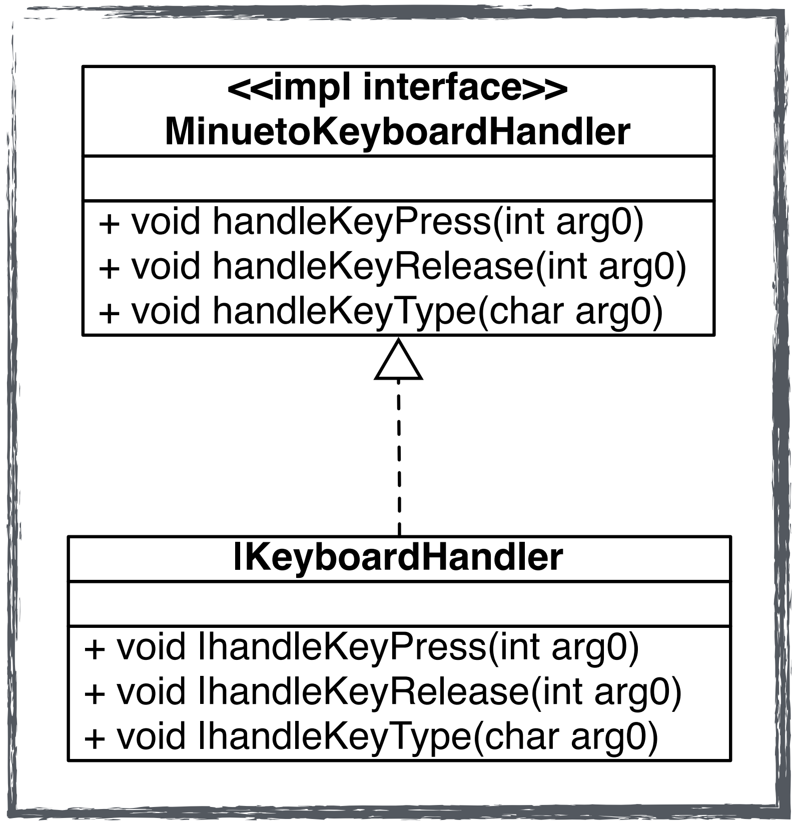
Clearly designates what user has to provide



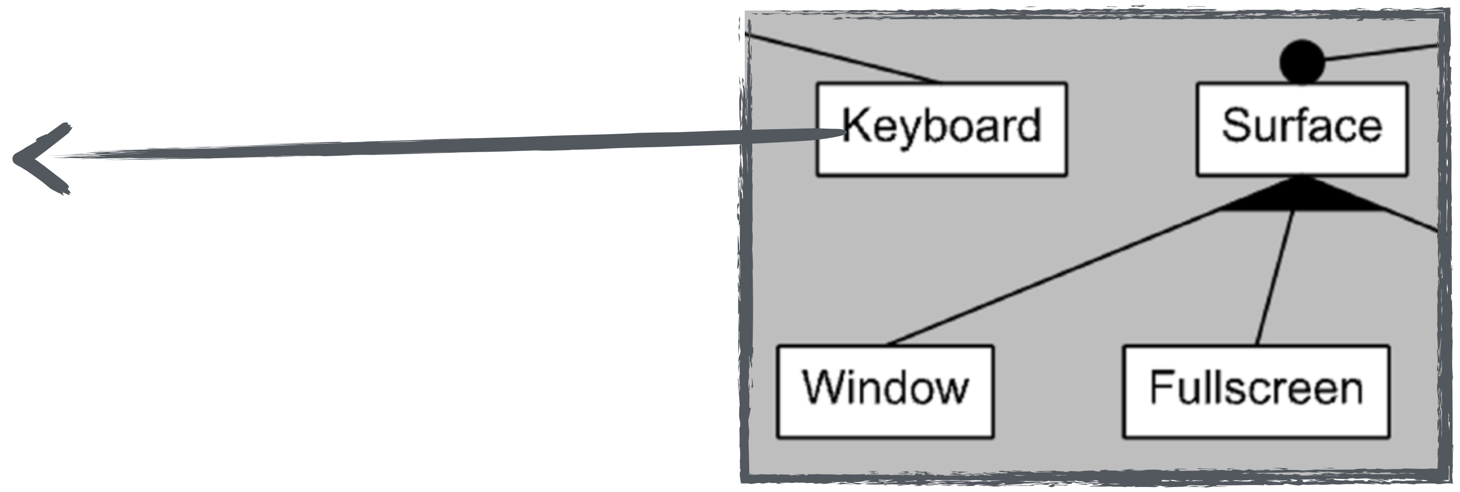
Force user to adapt correctly

The Customization Interface

Clearly designates what user has to provide



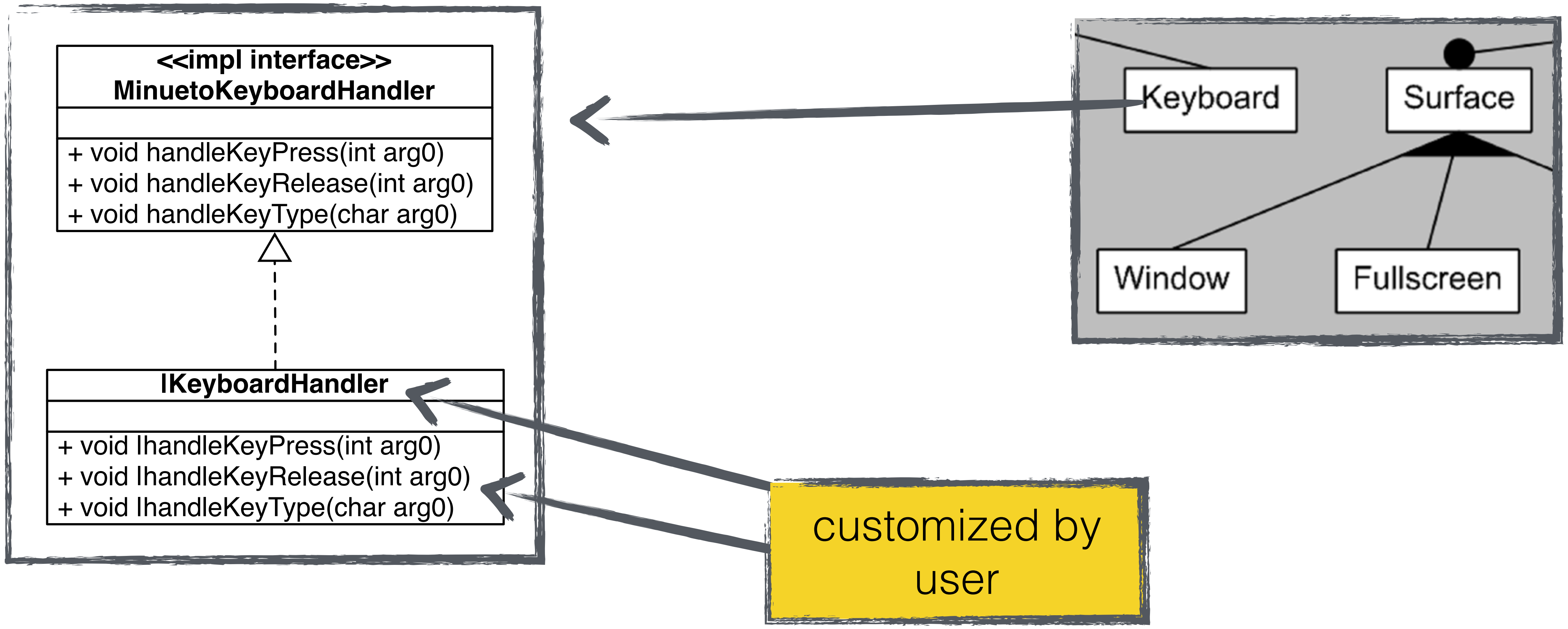
<<impl interface>>
MinuetoKeyboardHandler
+ void handleKeyPress(int arg0)
+ void handleKeyRelease(int arg0)
+ void handleKeyType(char arg0)
IKeyboardHandler
+ void IhandleKeyPress(int arg0)
+ void IhandleKeyRelease(int arg0)
+ void IhandleKeyType(char arg0)



Force user to adapt correctly

The Customization Interface

Clearly designates what user has to provide



<<impl interface>>
MinuetoKeyboardHandler
+ void handleKeyPress(int arg0)
+ void handleKeyRelease(int arg0)
+ void handleKeyType(char arg0)
IKeyboardHandler
+ void IhandleKeyPress(int arg0)
+ void IhandleKeyRelease(int arg0)
+ void IhandleKeyType(char arg0)
Keyboard
Surface
Window
Fullscreen
customized by user

Force user to adapt correctly

TouchCORE Tool Demo

Shows how Minueto framework concern is reused

Selecting the desired features from the high-level view of the framework

Trade-off analysis depending on impacts on non-functional goals

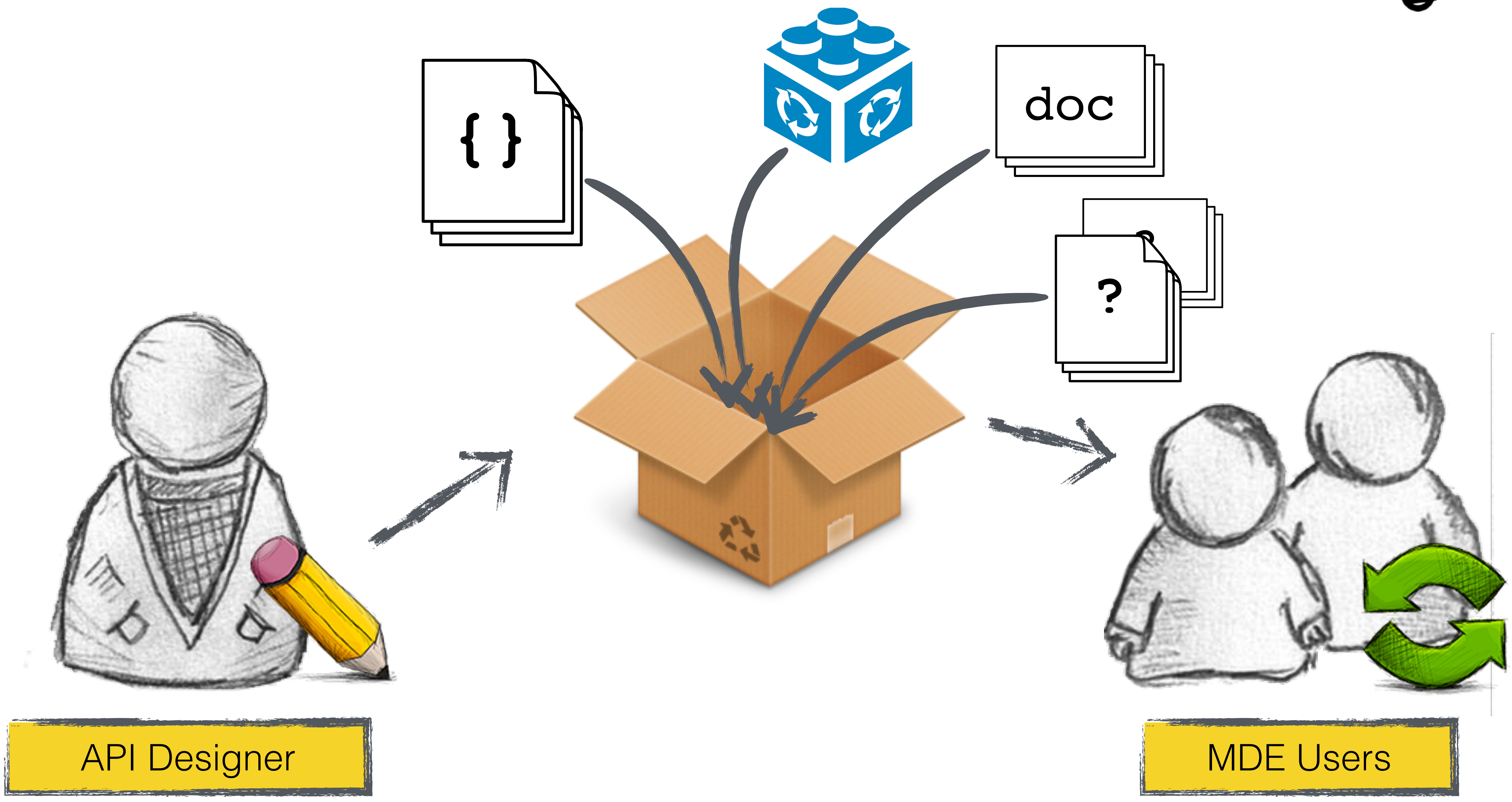
Produces subset of API based on chosen functionality

Customizing the chosen functionality (adapt to reuse reuse context)

Using the chosen functionality in a design model

View Demo Recording: phd_defence_demo.mov

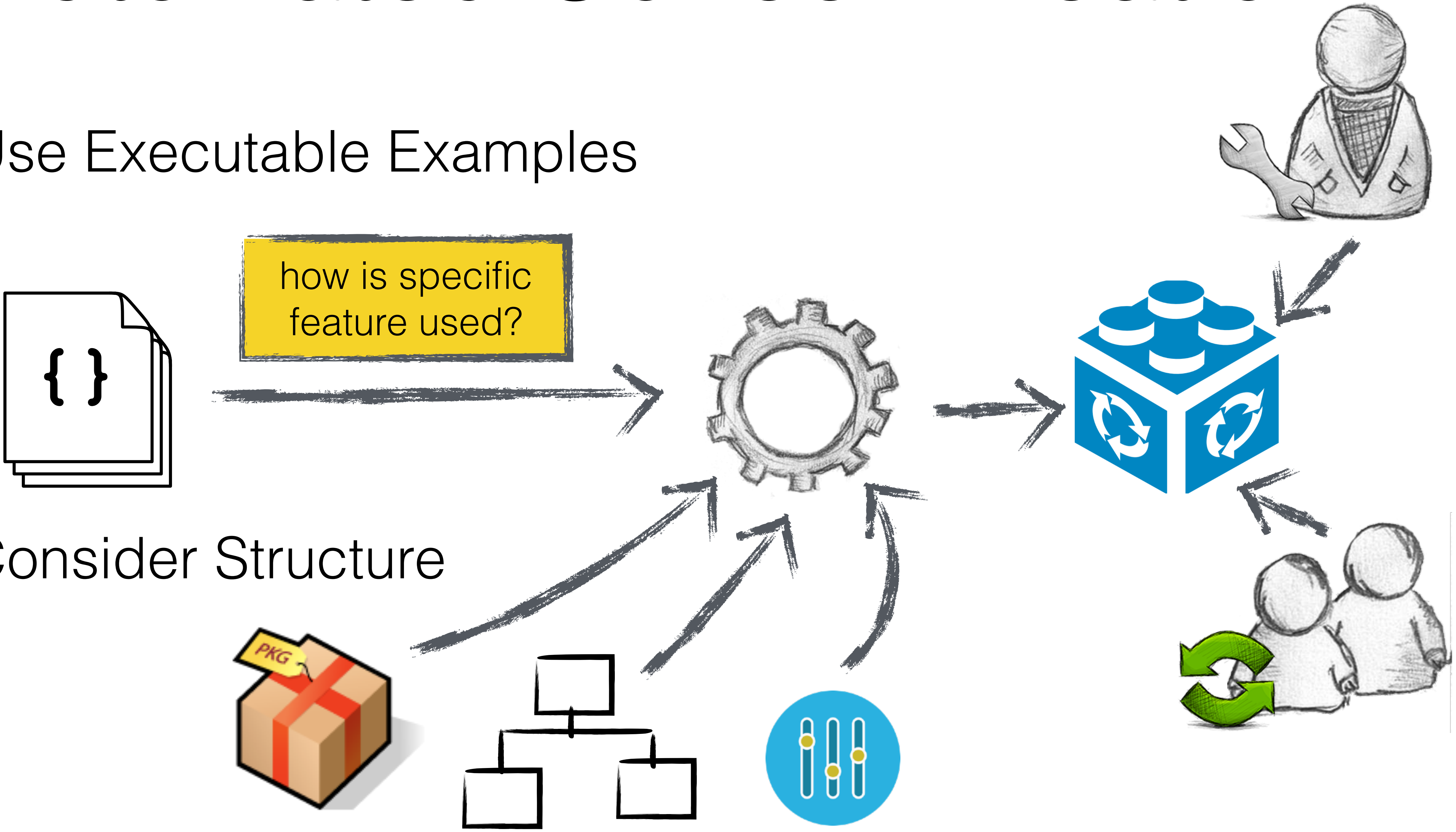
Concernification



Automated Concernification

Use Executable Examples

Consider Structure



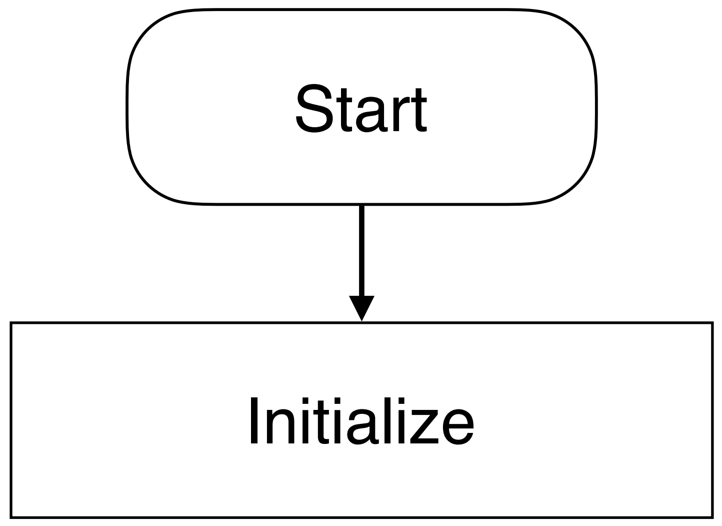
Extract feature model

Help with initial concern interface creation

Automated Concernification

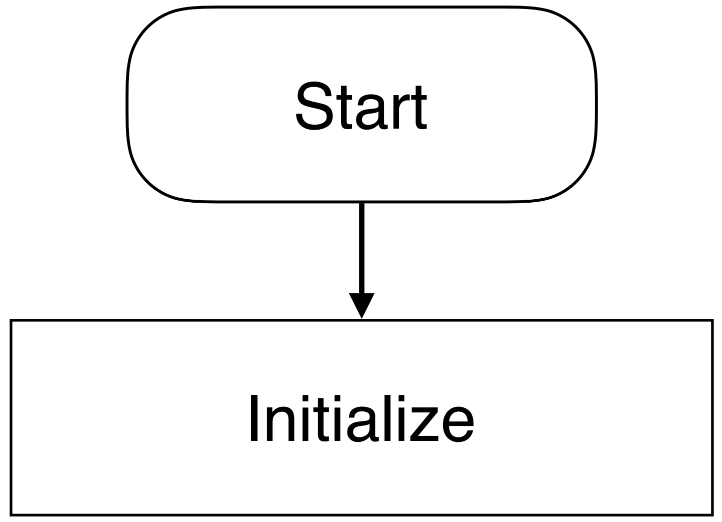
Uses Directed Acyclic Graph

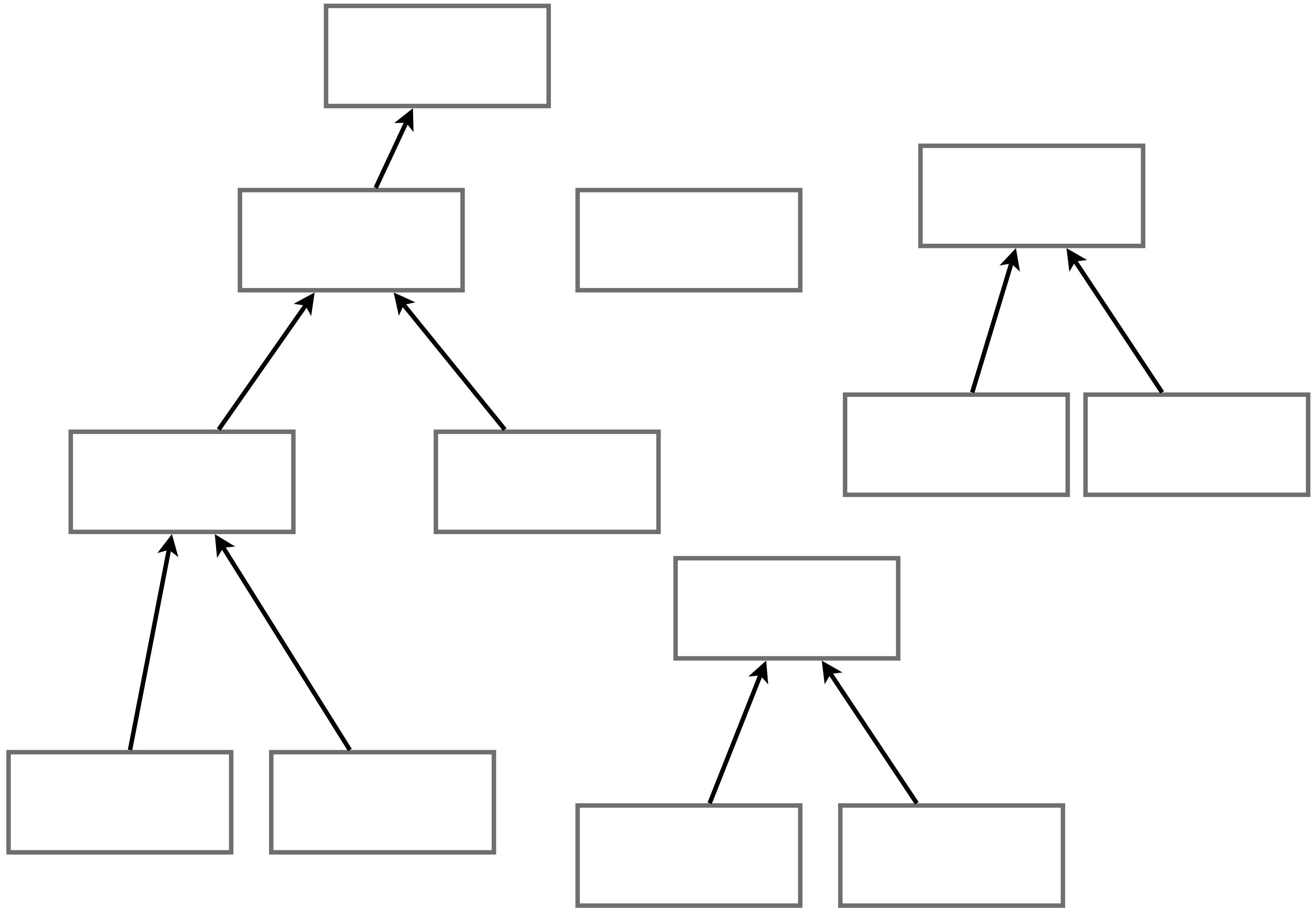
Automated Concernification



Uses Directed Acyclic Graph

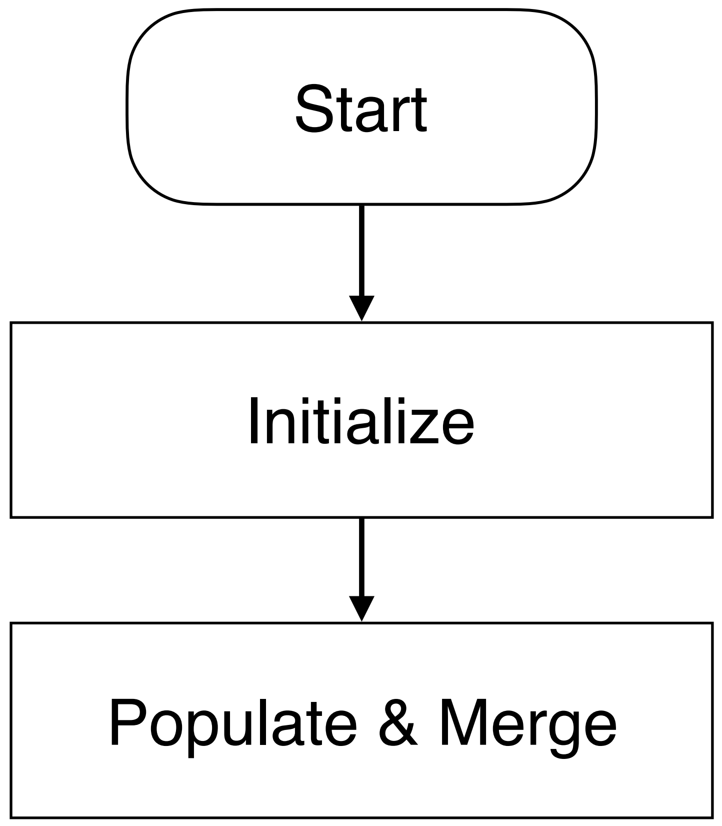
Automated Concernification

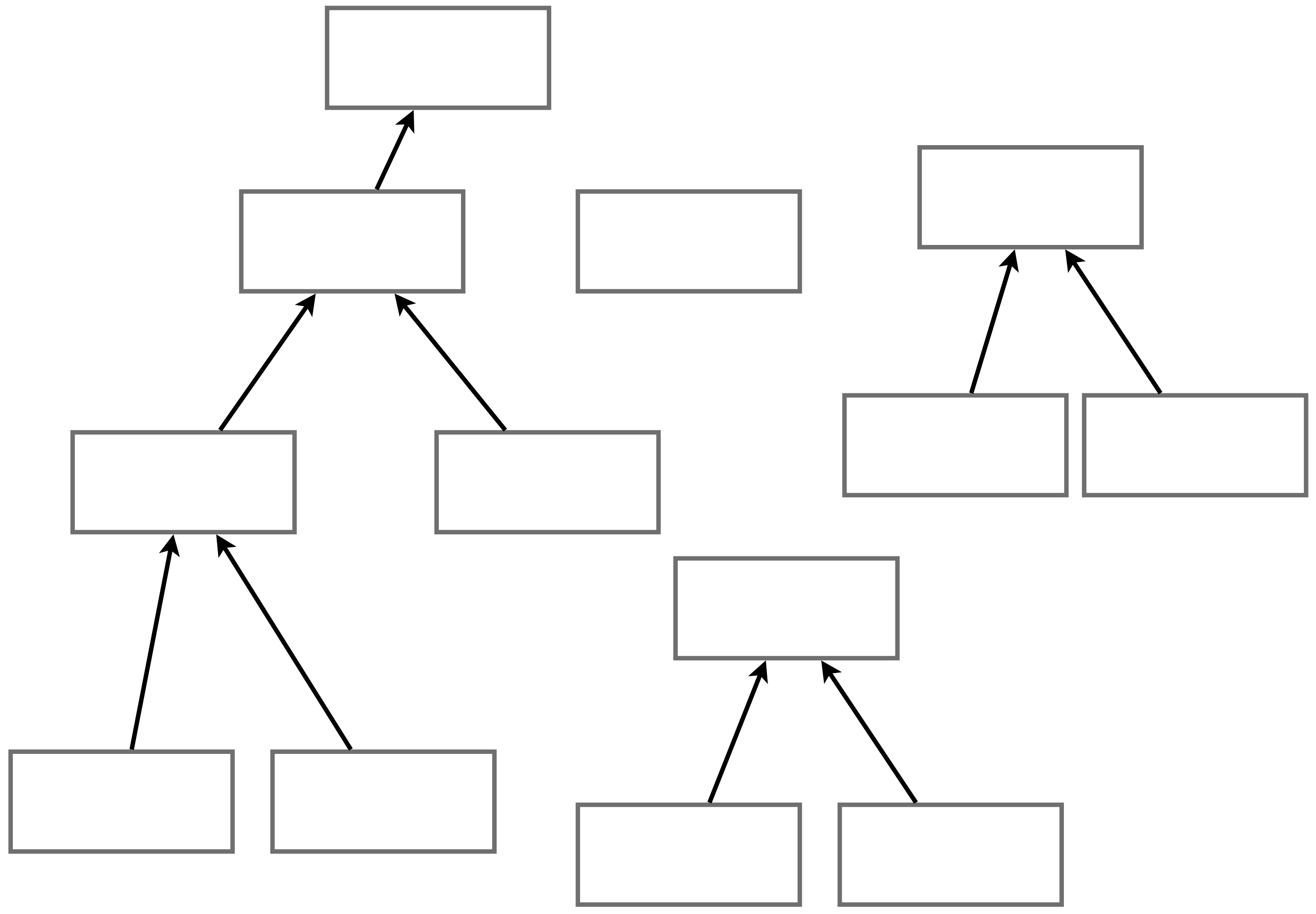




Uses Directed Acyclic Graph

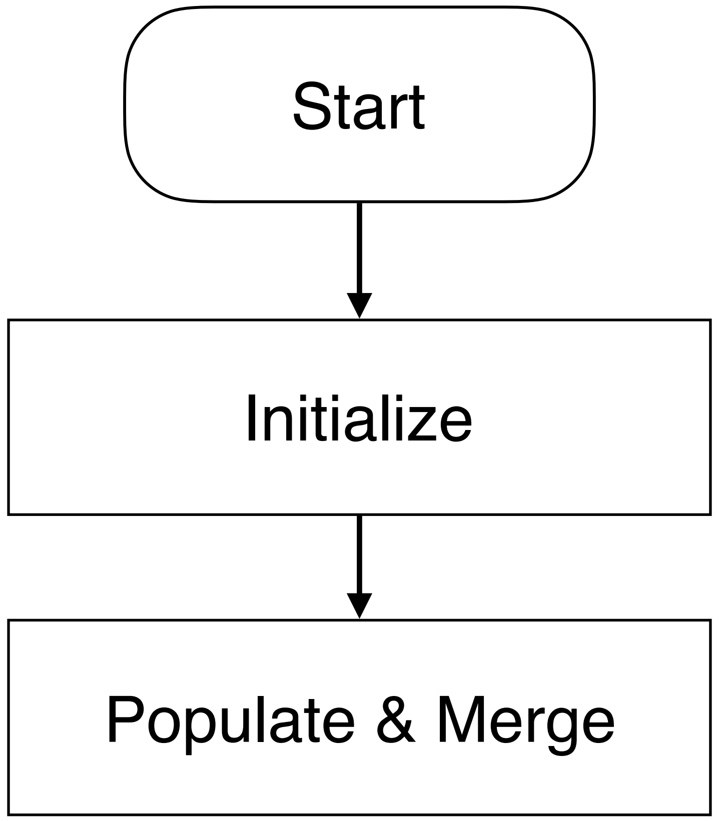
Automated Concernification

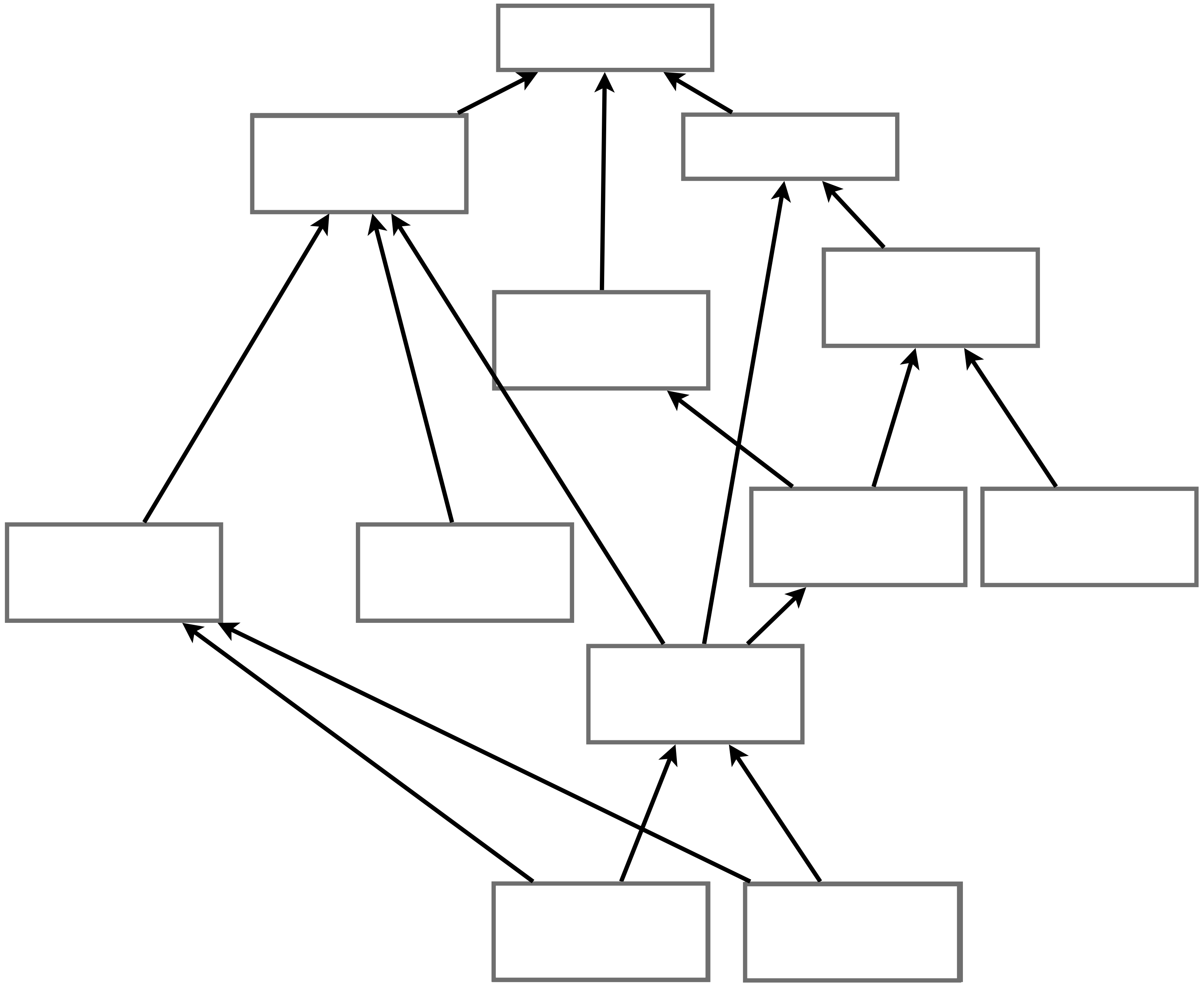




Use structure and example usage

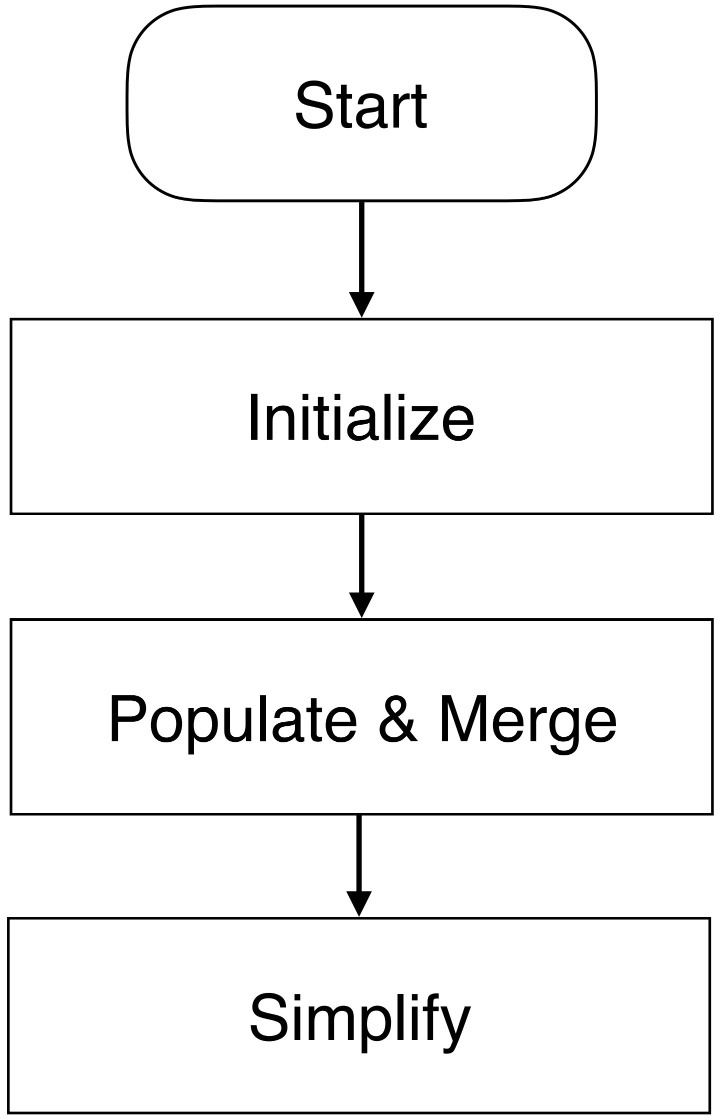
Automated Concernification

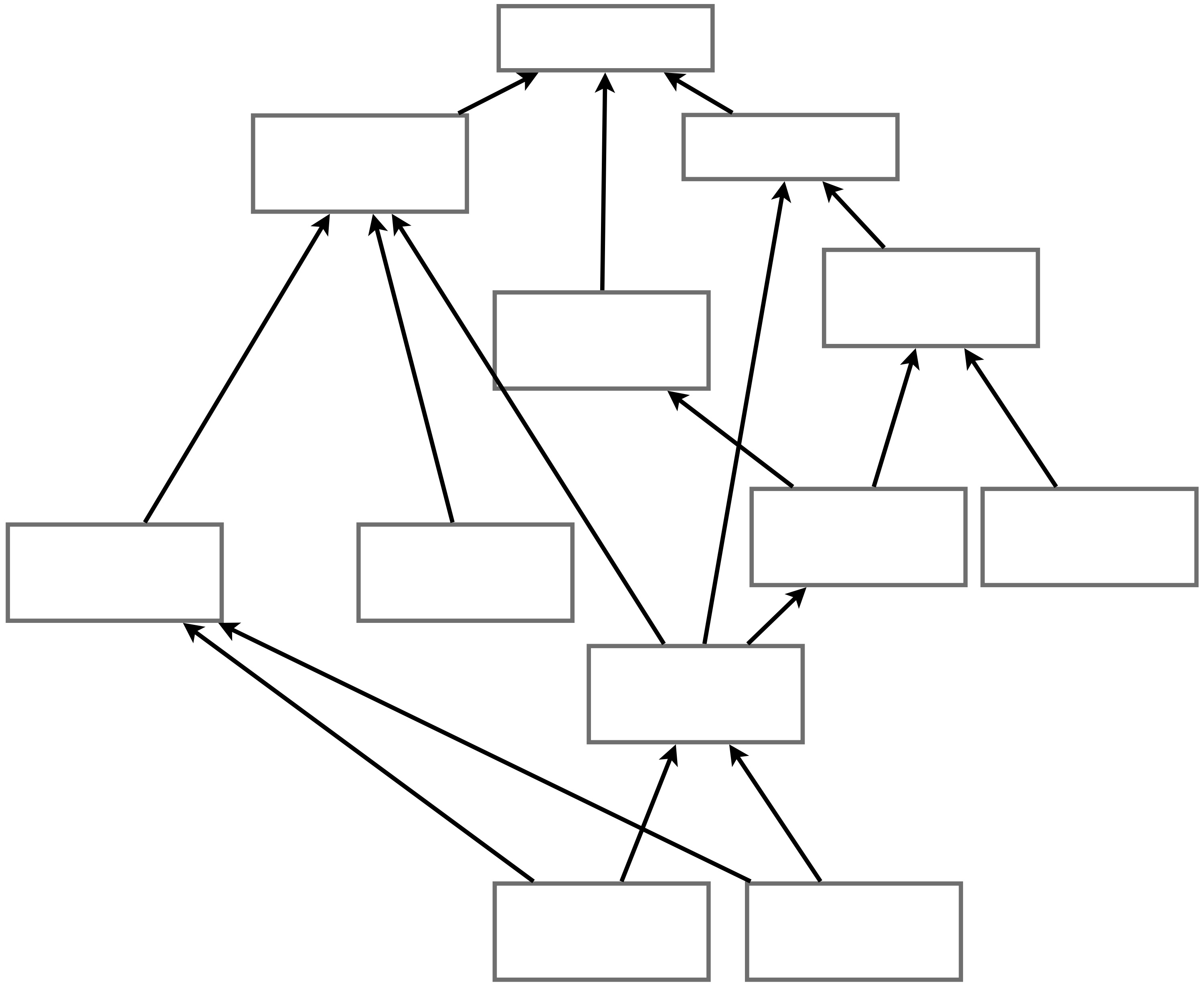




Use structure and example usage

Automated Concernification

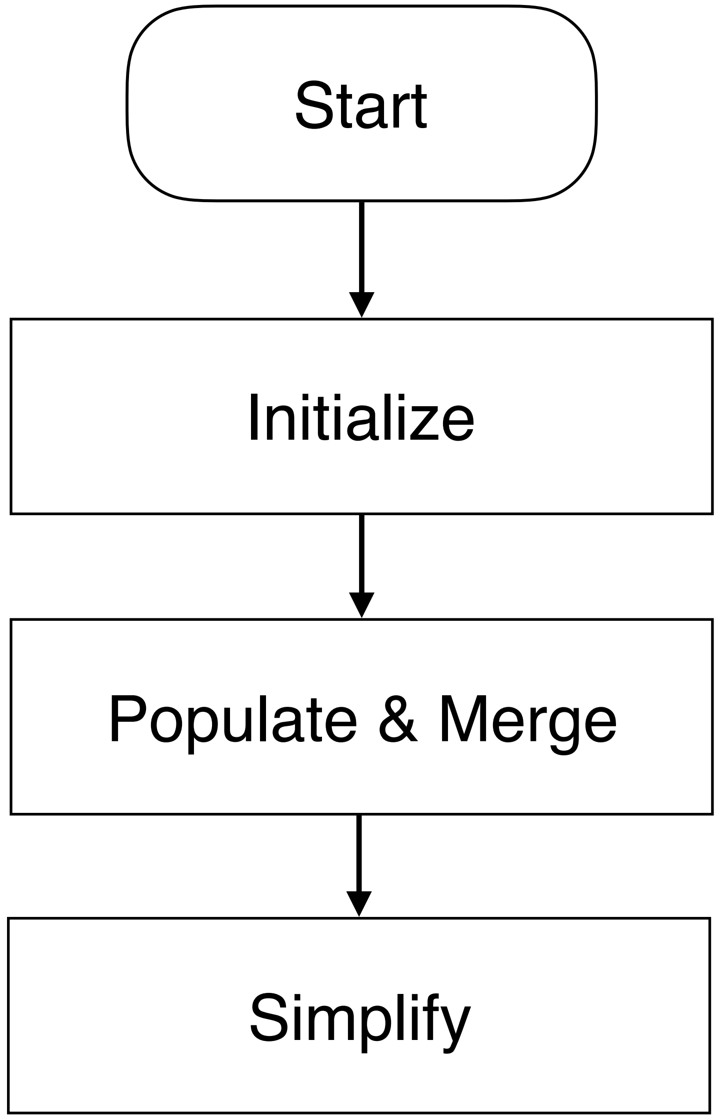


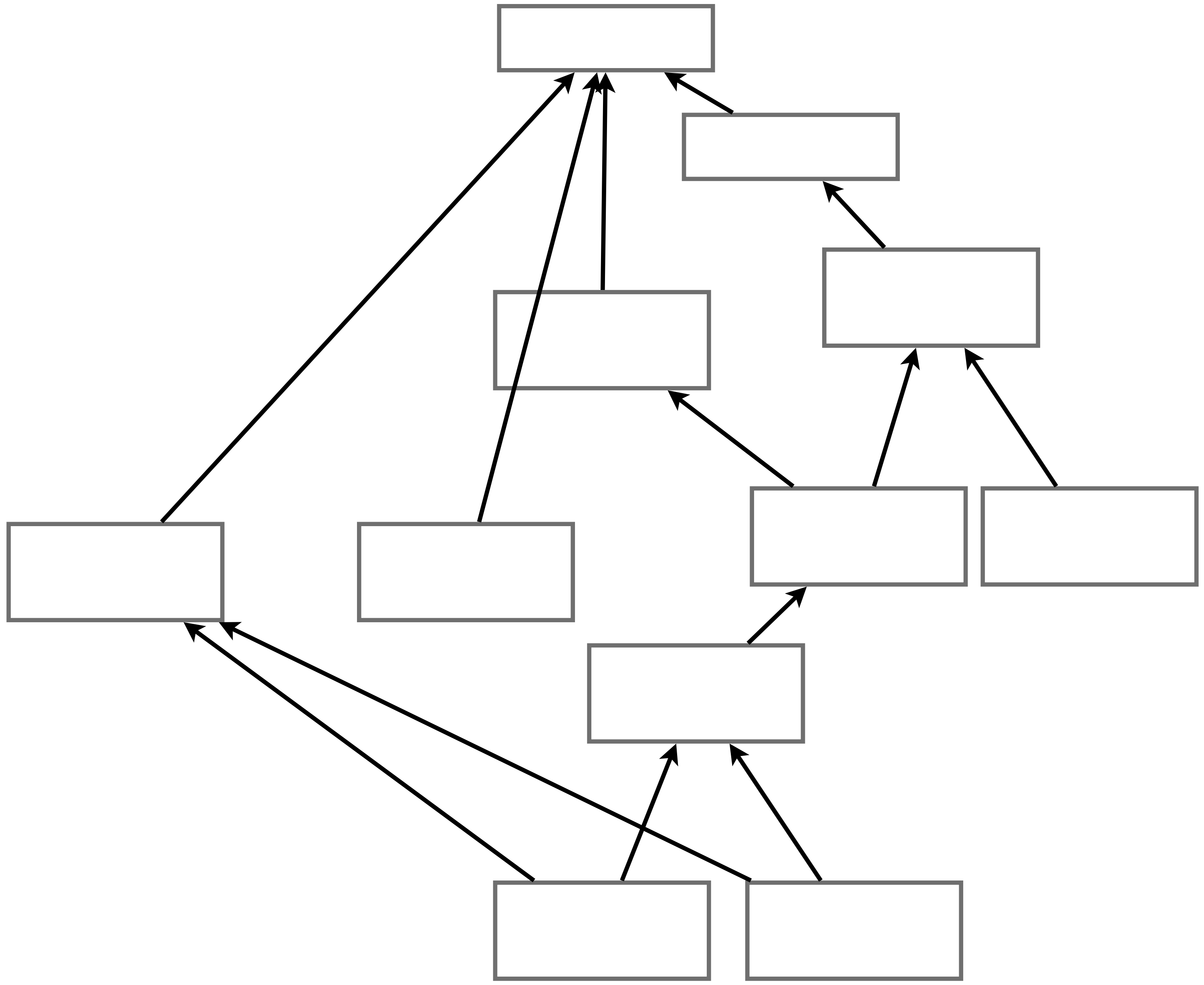


Maintain integrity of API

Can be fine-tuned by designer after

Automated Concernification

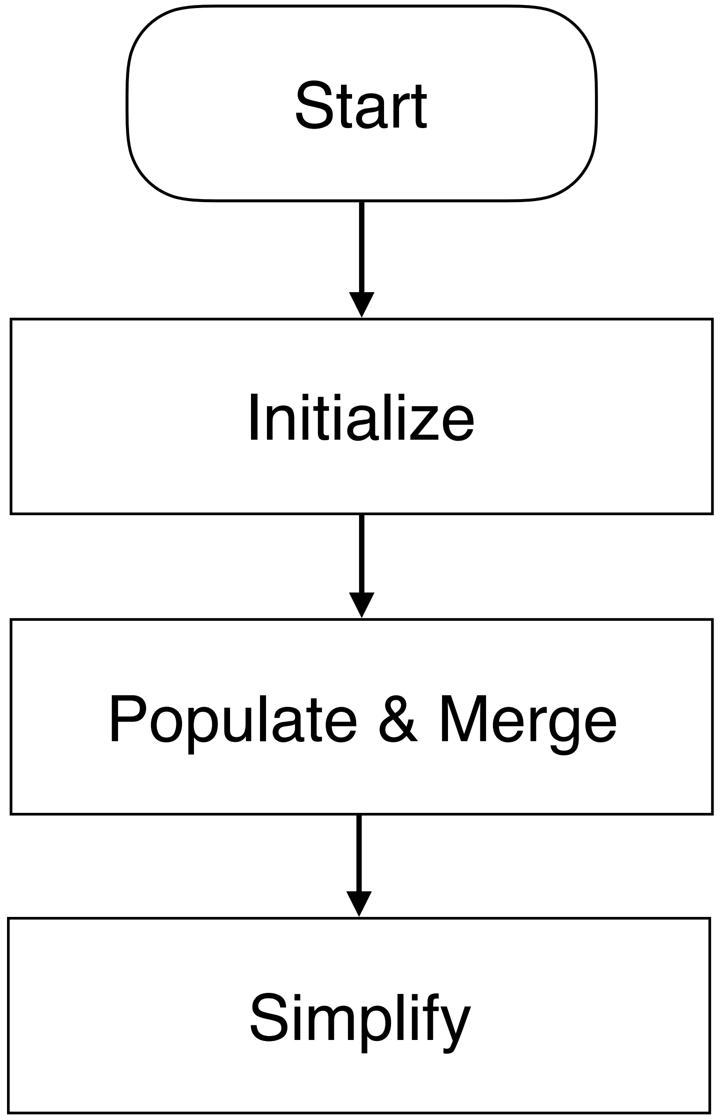


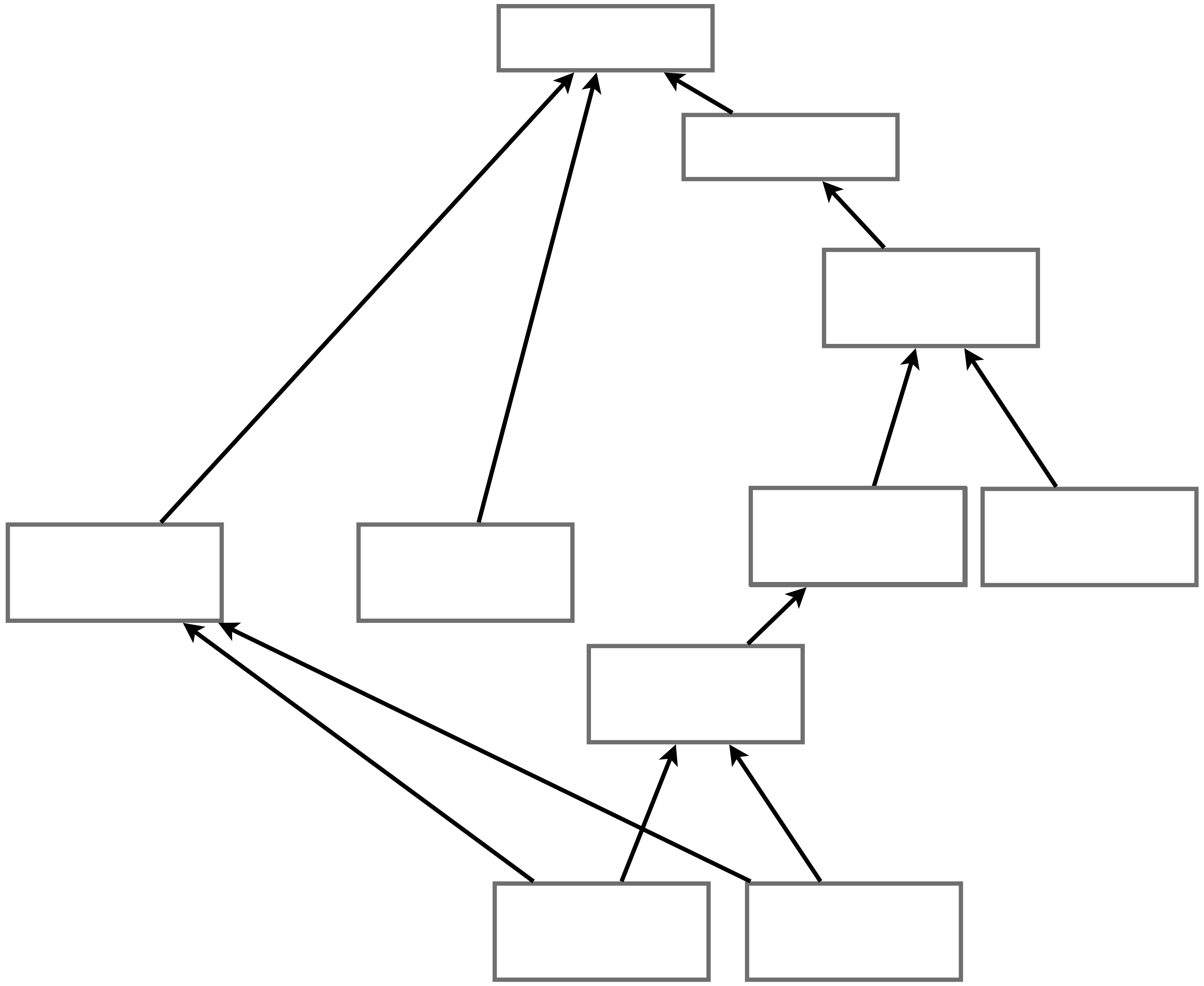


Maintain integrity of API

Can be fine-tuned by designer after

Automated Concernification

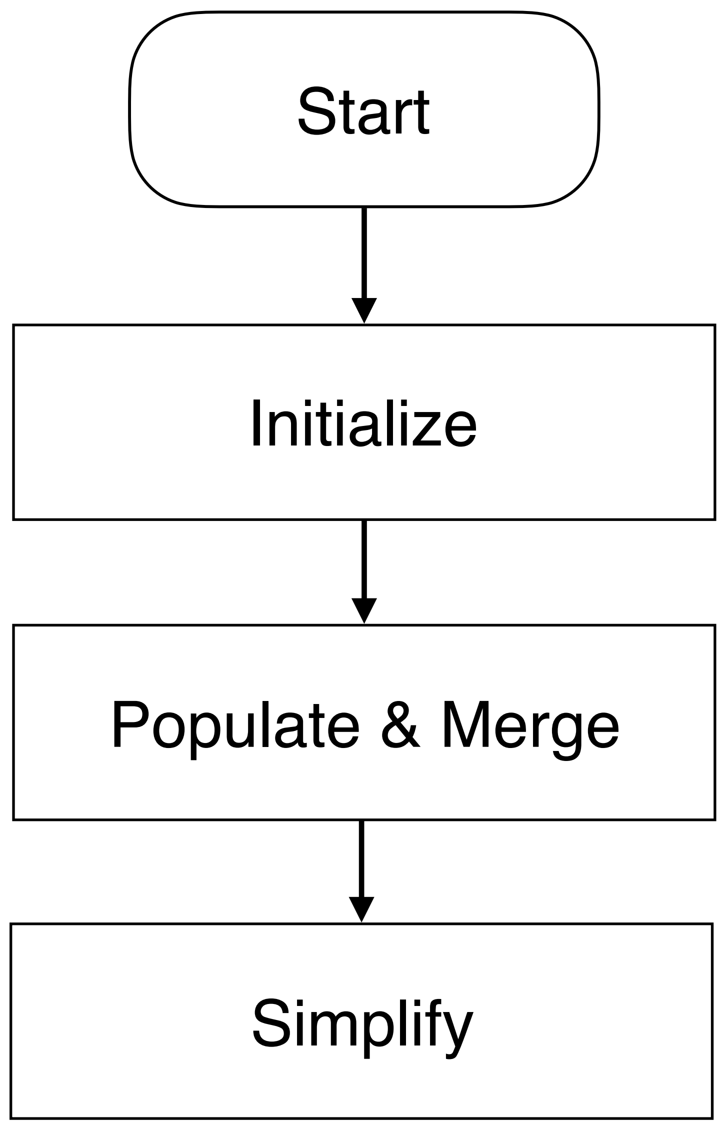


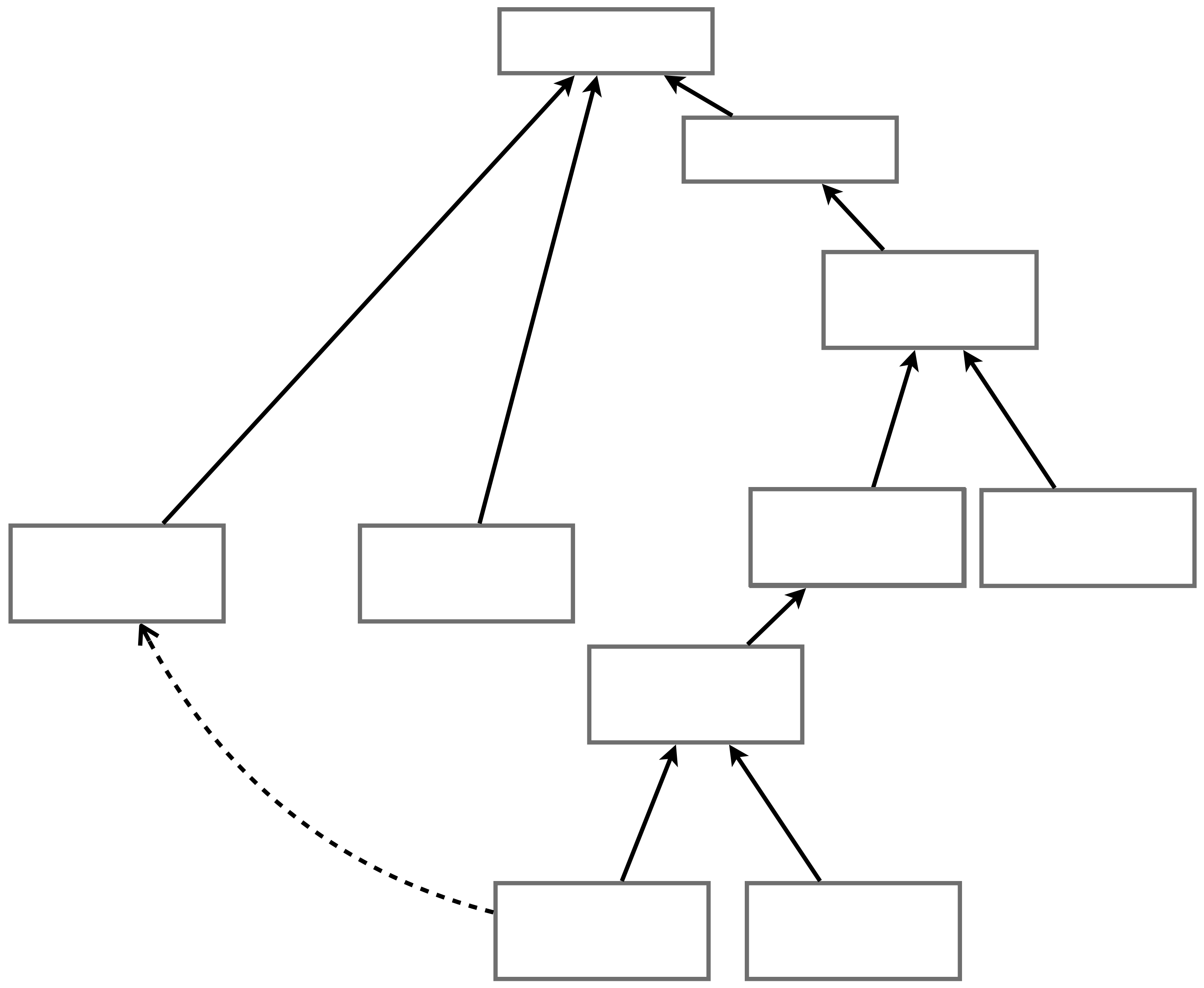


Maintain integrity of API

Can be fine-tuned by designer after

Automated Concernification





Maintain integrity of API

Can be fine-tuned by designer after

Automated Concernification

Validated on three frameworks

Automated Concernification

Workflow Concern

Validated on three frameworks

Automated Concernification

Workflow Concern

Minueto

Validated on three frameworks

Automated Concernification

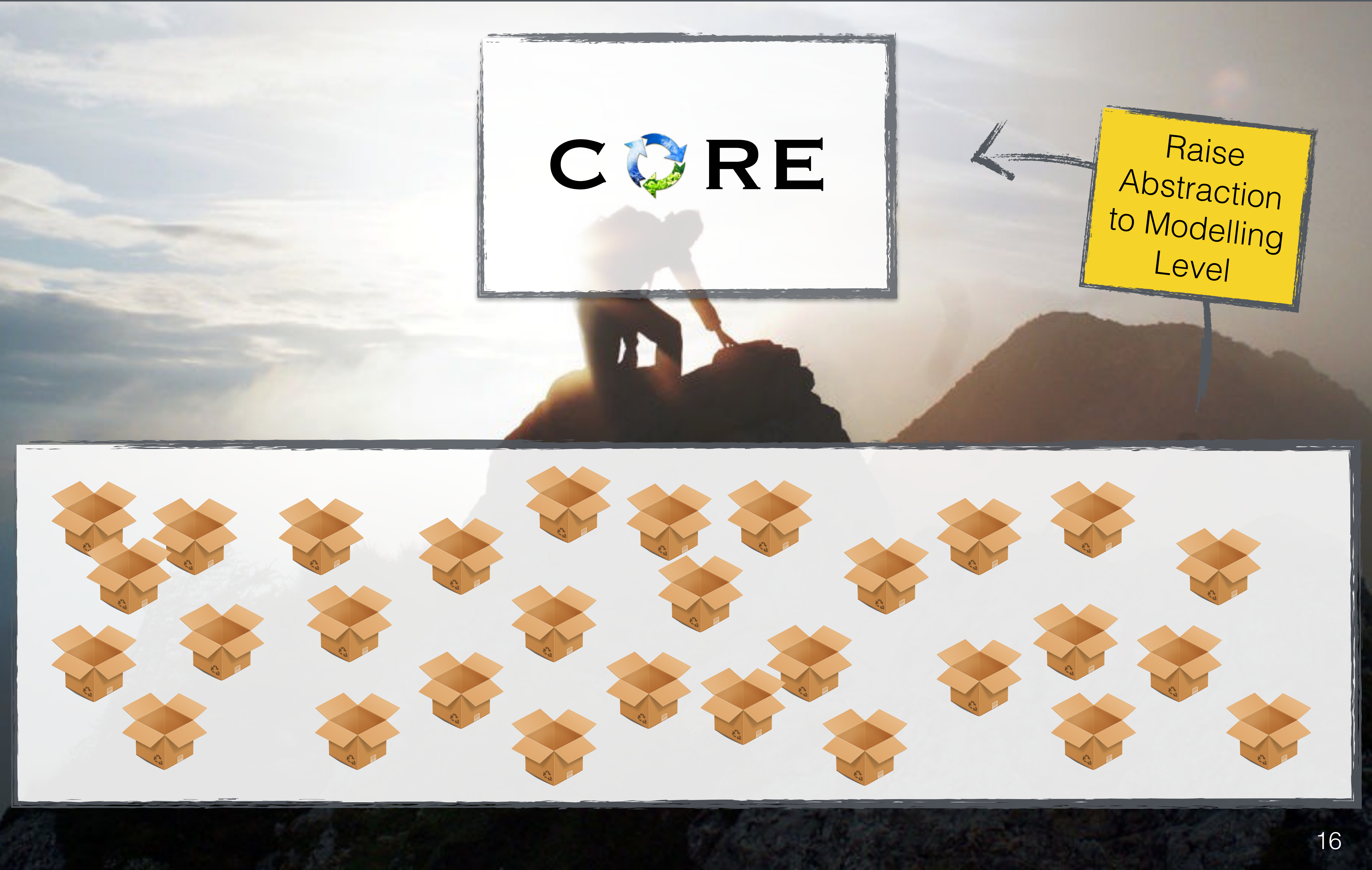
Workflow Concern

Minueto

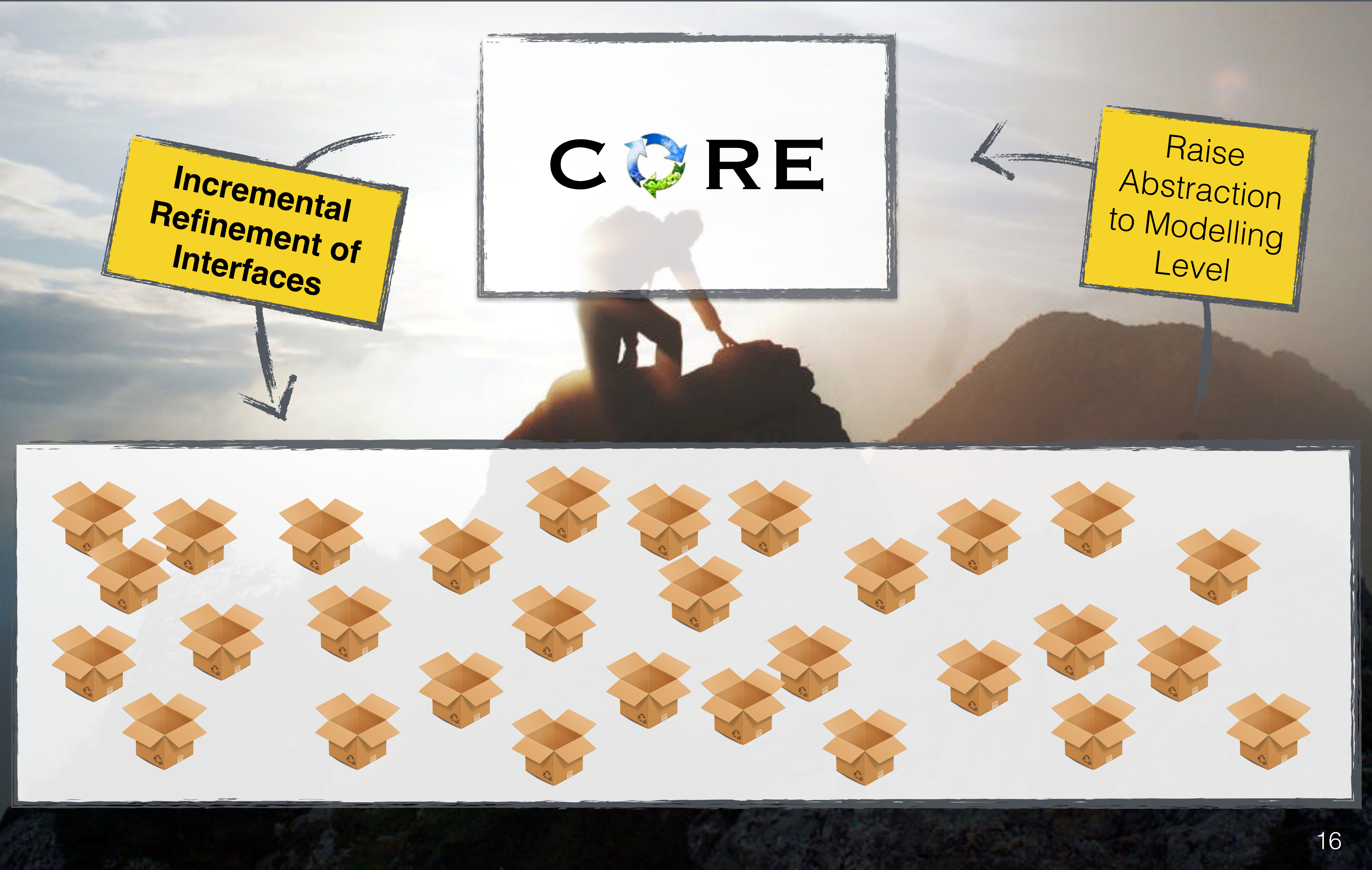
Android Notifications

Validated on three frameworks

Bridging the Gap



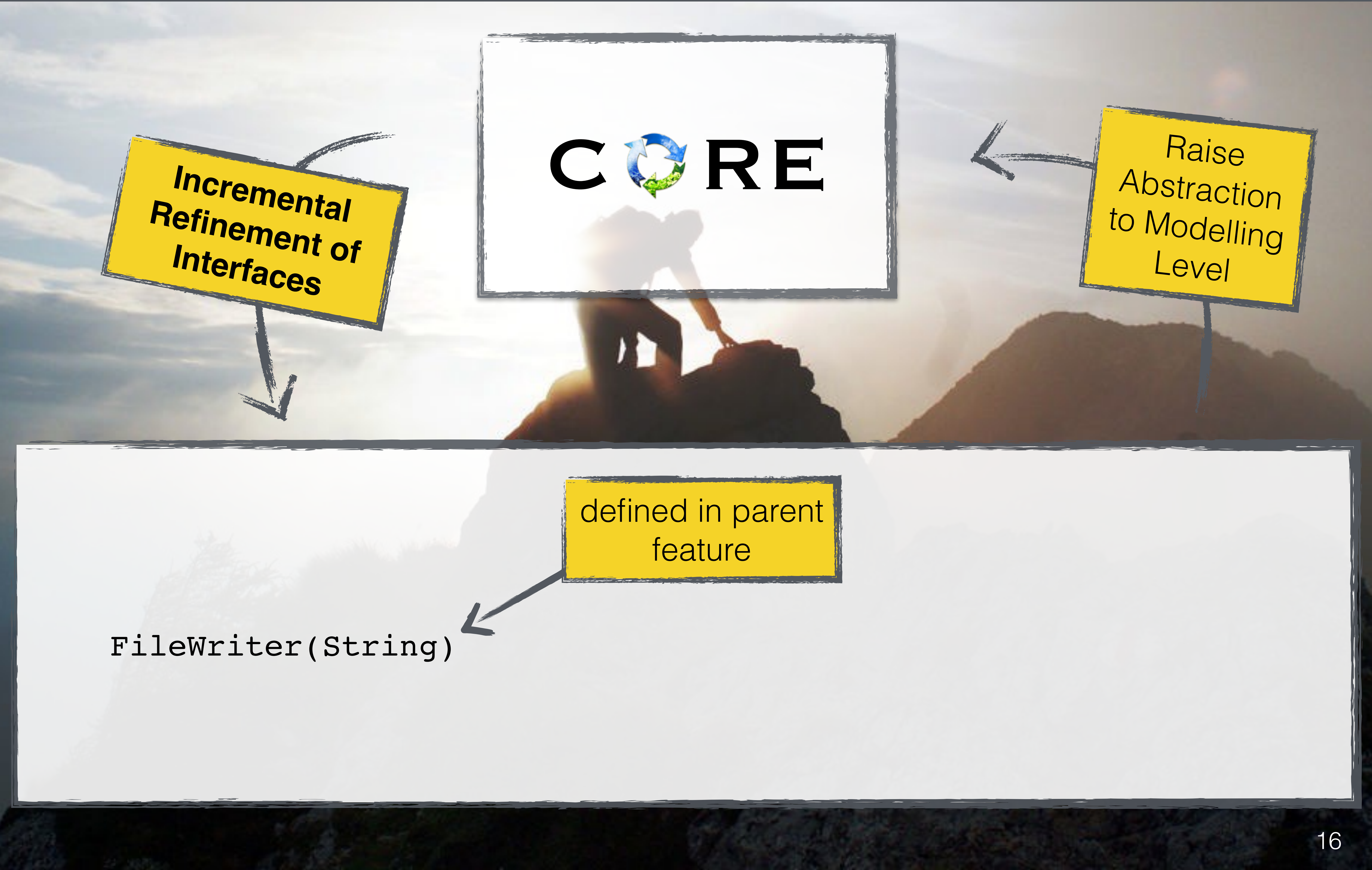
Bridging the Gap


CORE
Incremental Refinement of Interfaces
Raise Abstraction to Modelling Level

Bridging the Gap

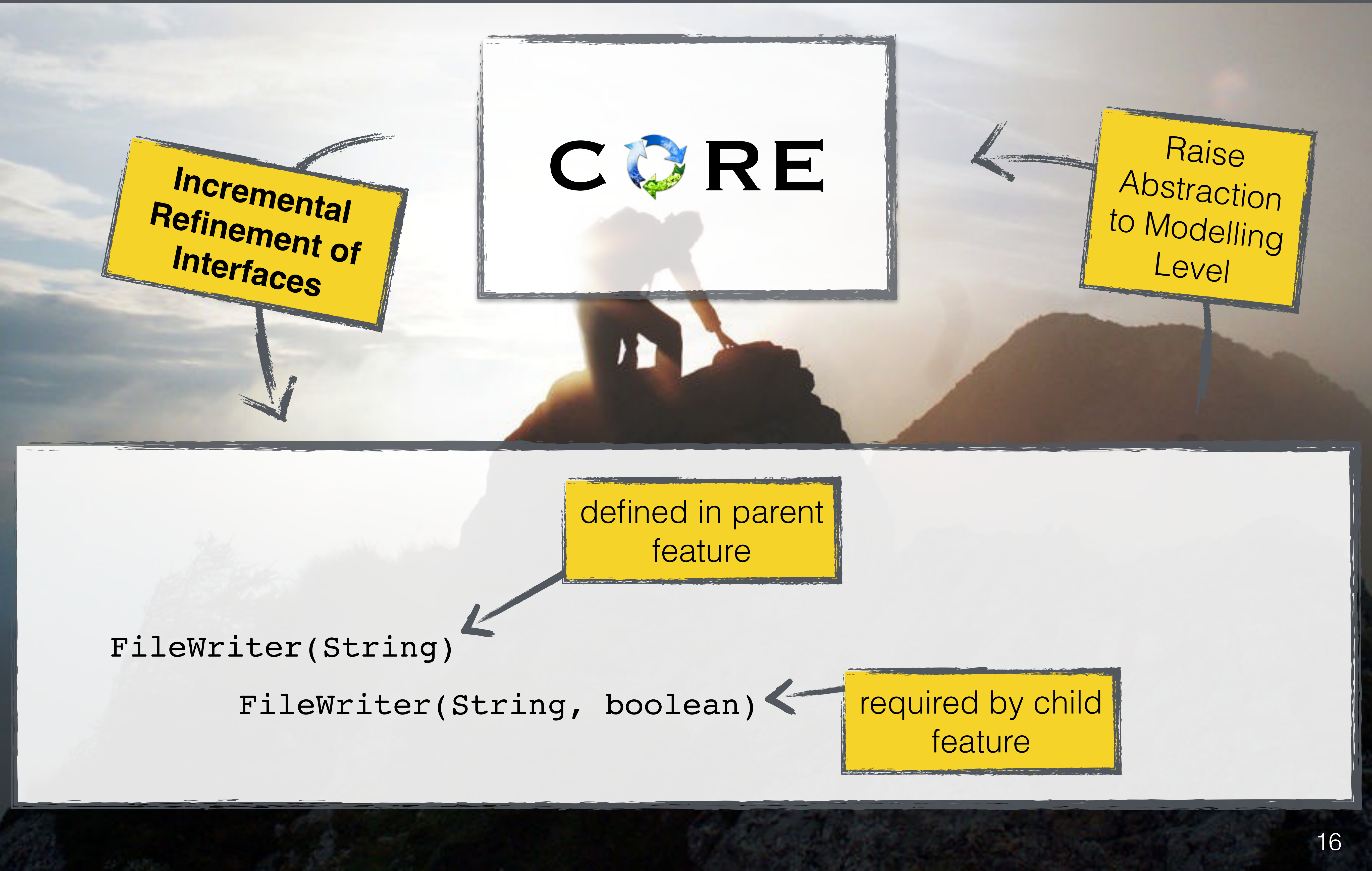
CORE
Incremental Refinement of Interfaces
Raise Abstraction to Modelling Level

Bridging the Gap


CORE
Incremental Refinement of Interfaces
Raise Abstraction to Modelling Level
defined in parent feature

```
FileWriter(String)
```

Bridging the Gap



CORE

Incremental Refinement of Interfaces

Raise Abstraction to Modelling Level

defined in parent feature

```
FileWriter(String)
      FileWriter(String, boolean)
```

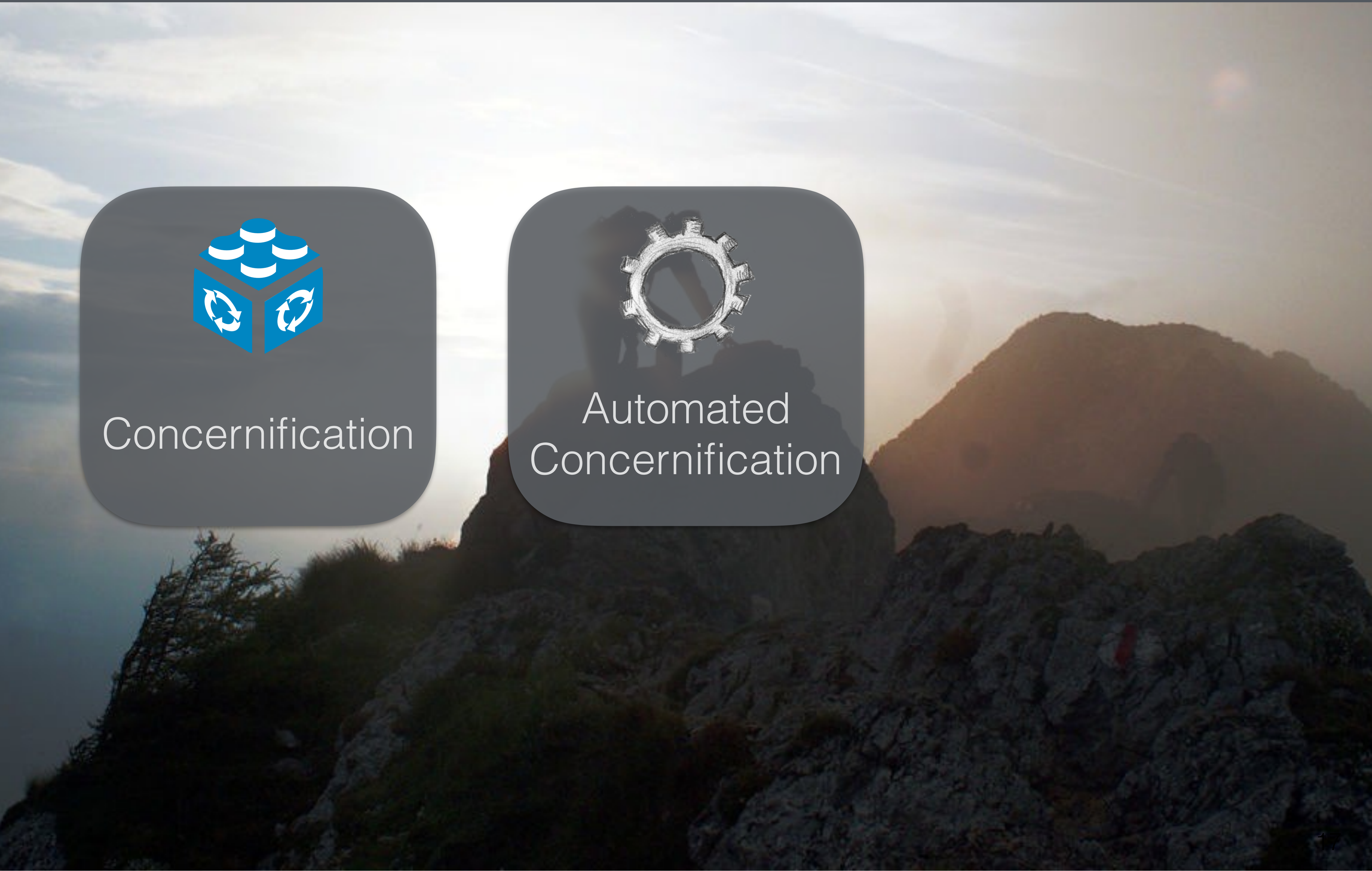
required by child feature

Summary

Summary

Concernification

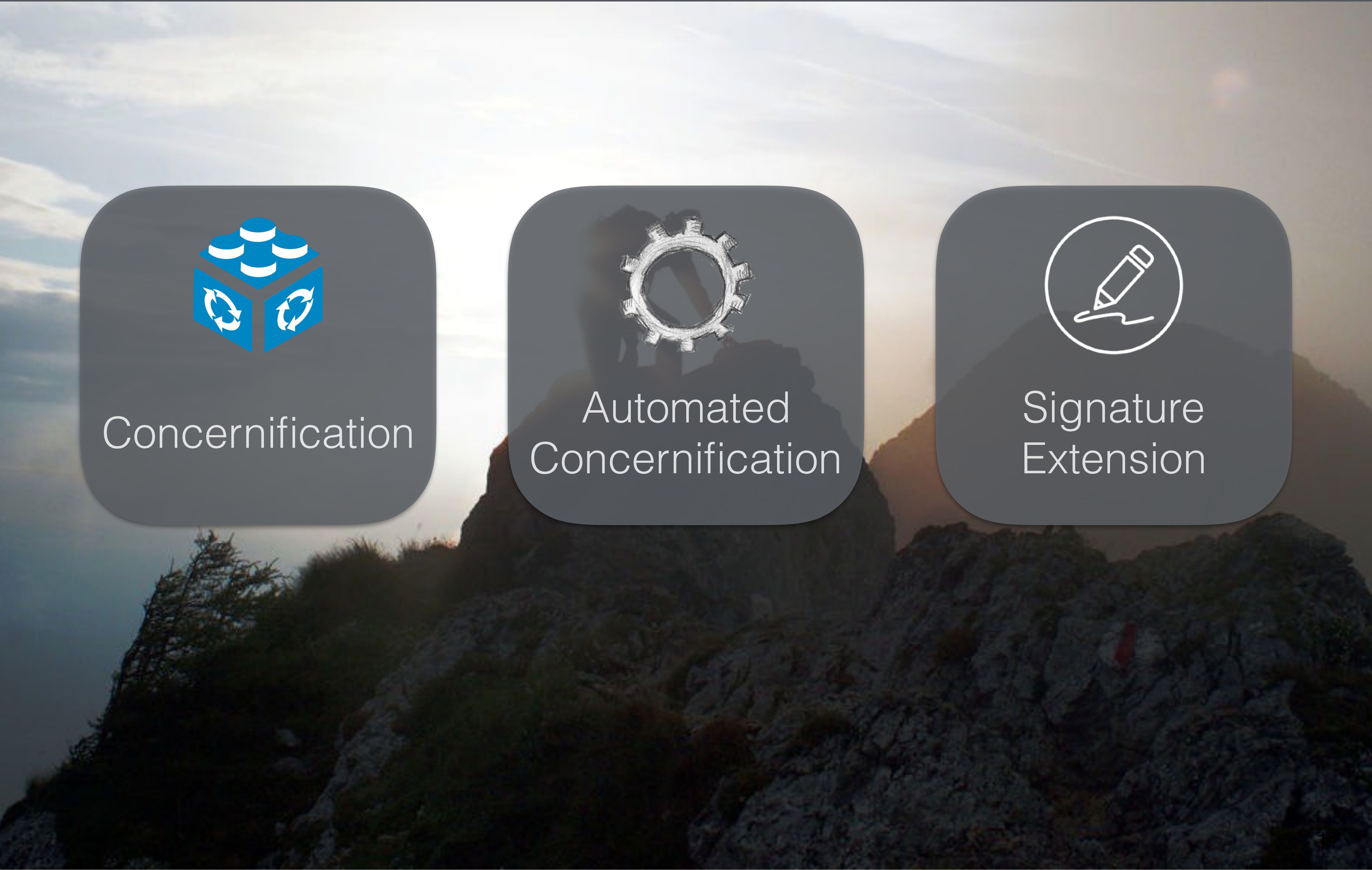
Summary



Concernification

Automated Concernification

Summary



Concernification

Automated Concernification

Signature Extension

Future Work

Future Work

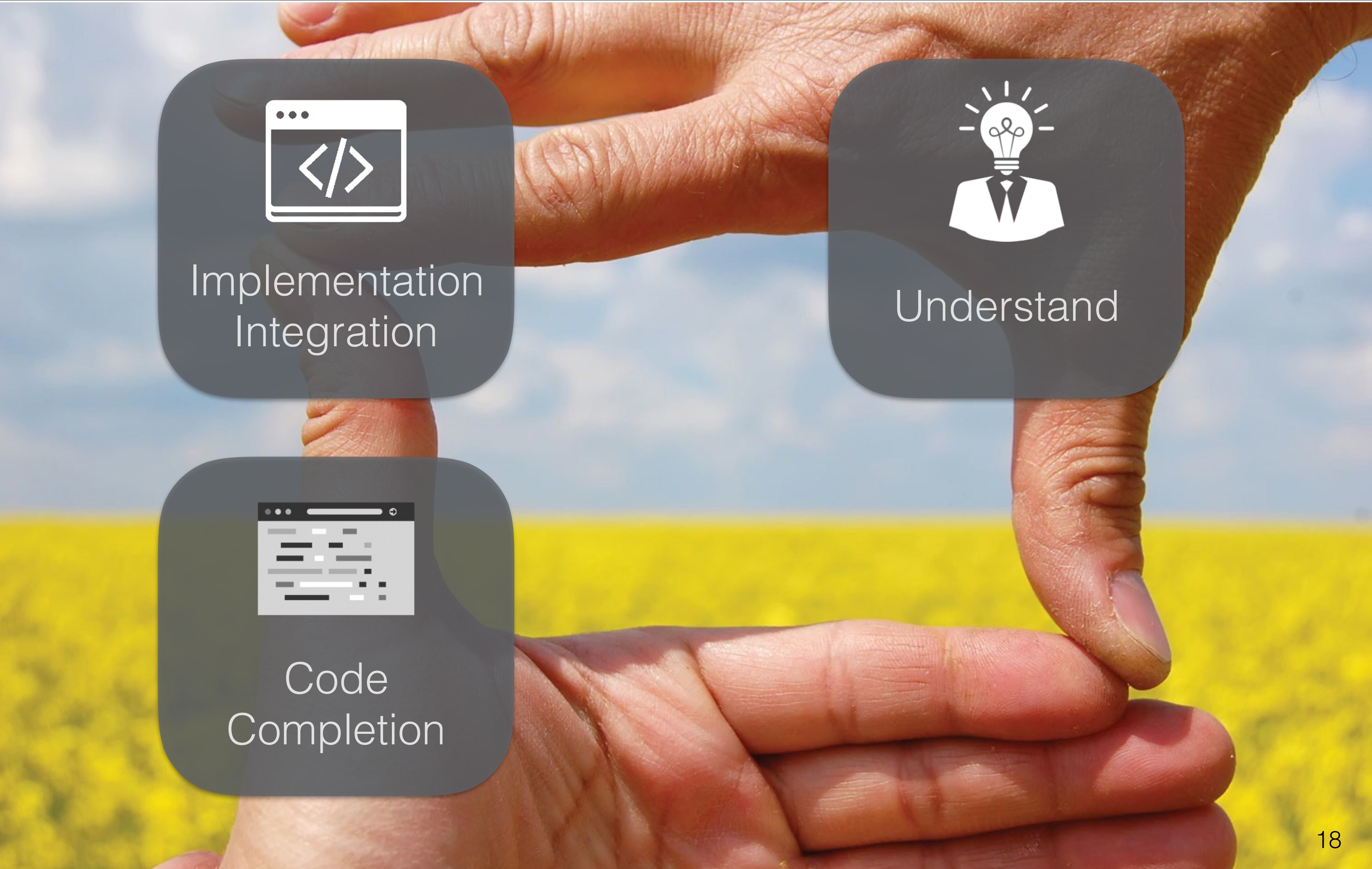
Implementation
Integration

Future Work

Implementation
Integration

Understand

Future Work

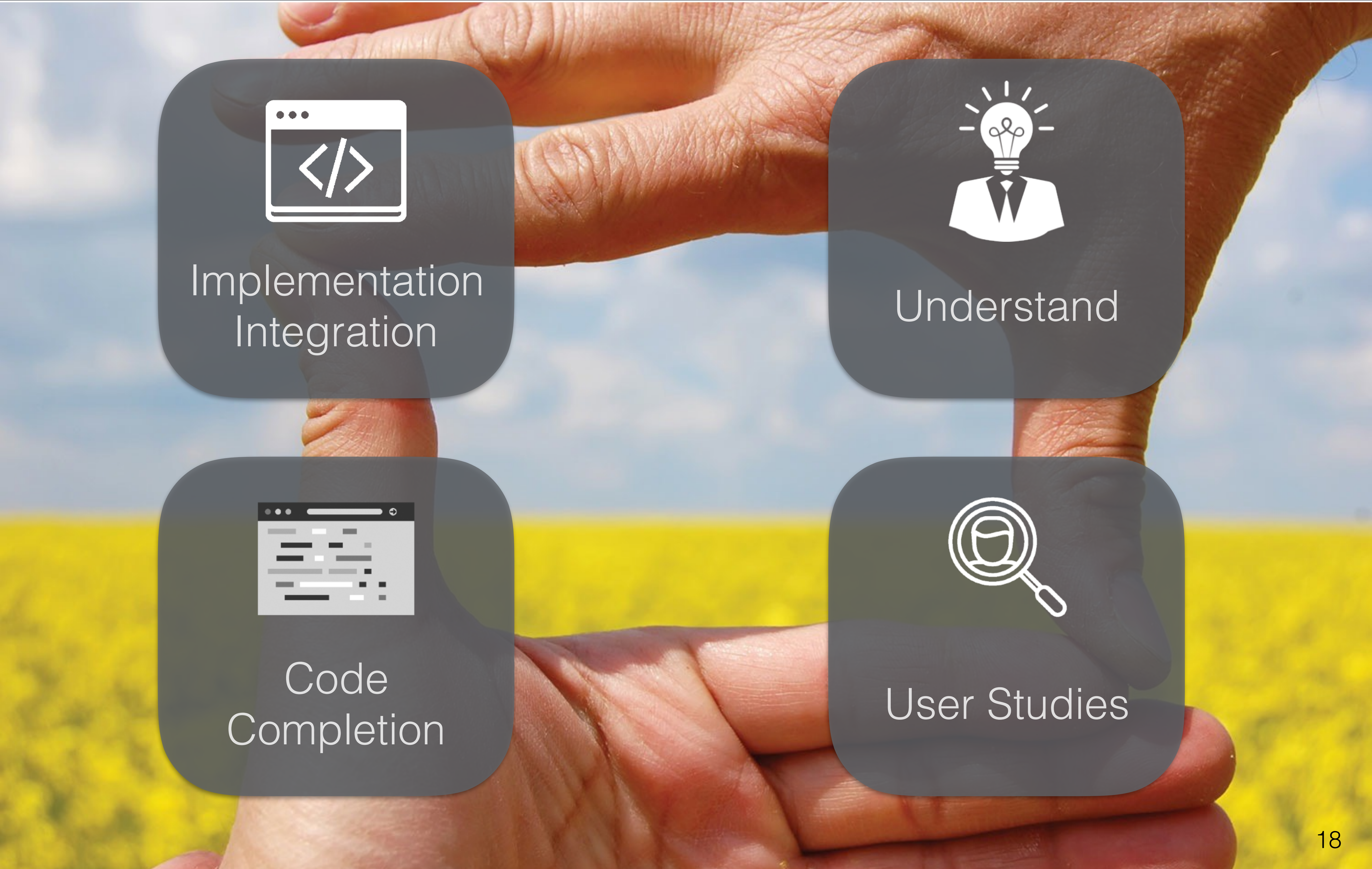


Implementation Integration

Understand

Code Completion

Future Work



Implementation Integration

Understand

Code Completion

User Studies

Summary

Key Contributions

Concernification

- concern interfaces for existing functionality
- tool support in TouchCORE
- qualitative study with Minueto developers

Automated Concernification

- algorithm to automate the creation of concern interfaces for existing frameworks
- implementation of algorithm
- validation of algorithm on three frameworks

Signature Extension

- identification and description of four difficult situations and empirical study on Java Platform API
- signature extension approach and support for class diagrams in CORE
- application of signature extension to two concerns