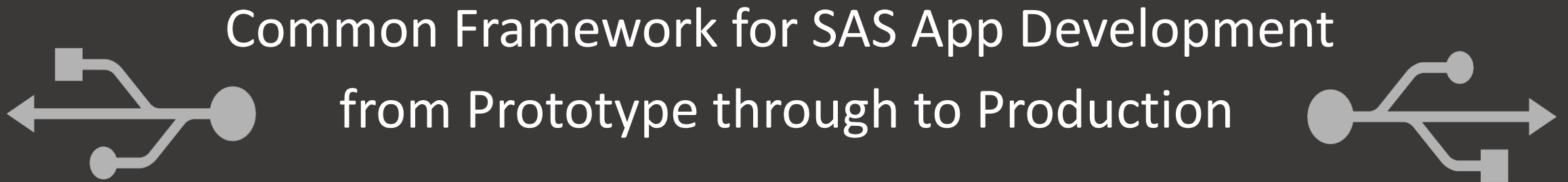


Interfacing with Interfaces



Overview

Problem: tightly-coupled code

Solution: define interfaces

Problem: plumbing for prototype

Solution: Open Source interface adapters

Problem: different platforms

Solution: implement abstract interface

sas-appfitter project

Interface (*v*) - connect in a compatible way

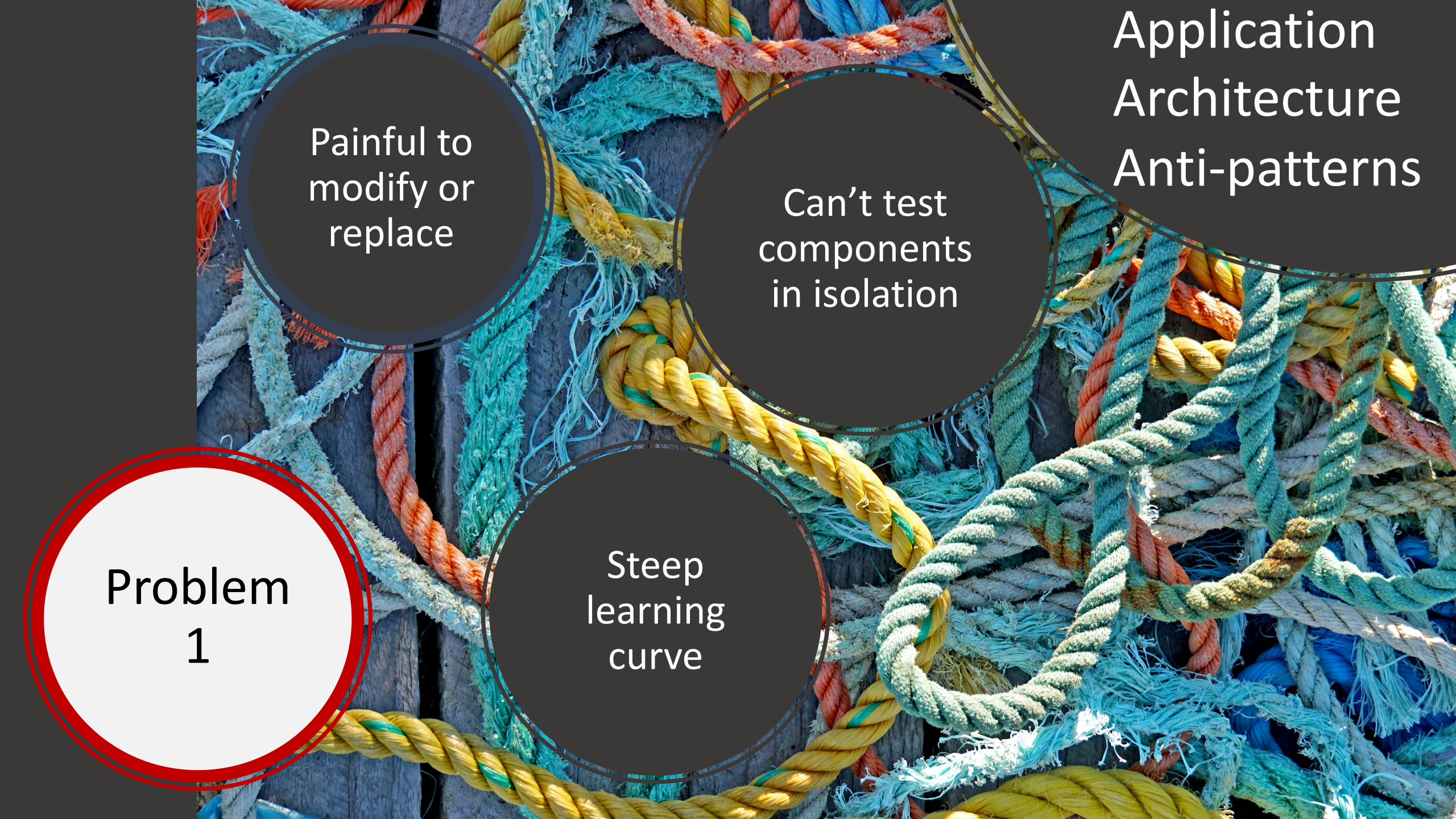
Interface (*n*) – common boundary across which components of a system communicate with one another e.g. GUI, API

Interface (*n, object-oriented programming*) – abstract definition of an object's inputs, outputs, and methods i.e. generic blueprint



Application
Architecture
Anti-patterns

Problem
1



Application
Architecture
Anti-patterns

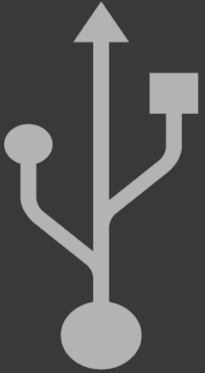
Painful to
modify or
replace

Can't test
components
in isolation

Problem
1

Steep
learning
curve

Define interfaces between components up-front



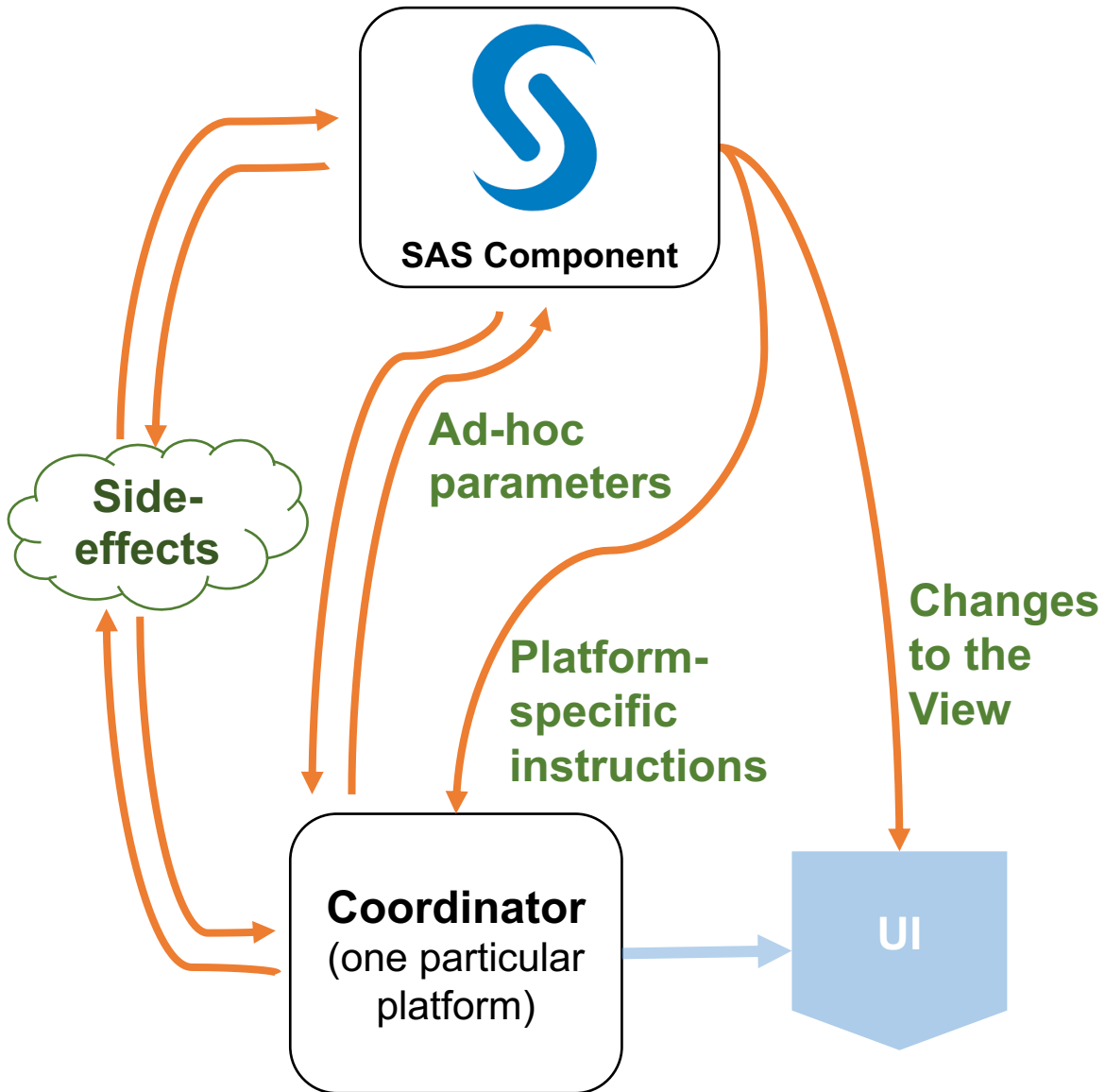
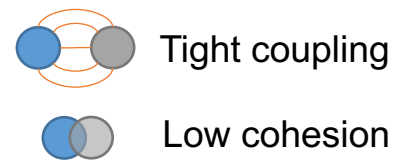
Solution 1

Inputs and outputs are agreed at the start,
documenting component definitions

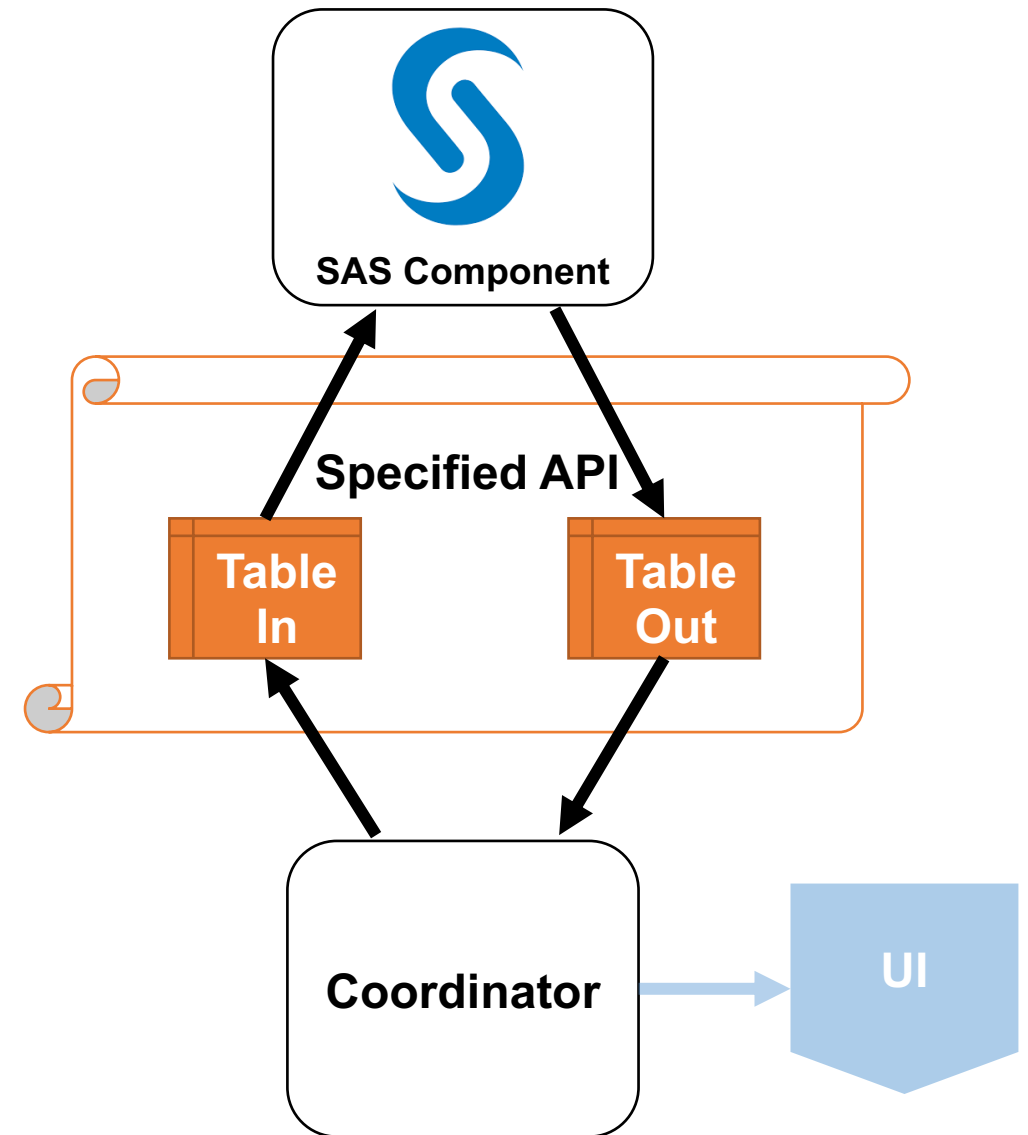
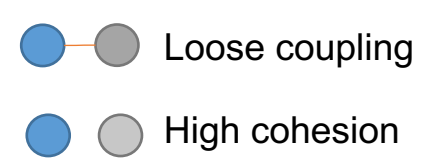
Component testing and emulation are simplified,
allowing test-driven (robust) and parallel (quick)
development

Separation of components by design leads to better
portability

Undefined boundaries (avoid this, it can get messy)



Boundary-first design (predictable, interchangeable)



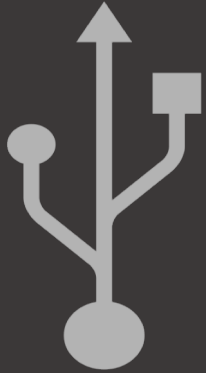
*“JavaScript? Java development is too expensive.
Besides, we are only allowed to use SAS”*

– once heard from a clinical programming manager

Problem
2

- SAS programmers may need training to understand and use other platforms
 - Development requires extensive effort and dependence on IT
 - Work restricted to validated environments
- ➔ Result: resistance to development of proofs of concept in Pharma

Open Source Adapters to Interface SAS to App

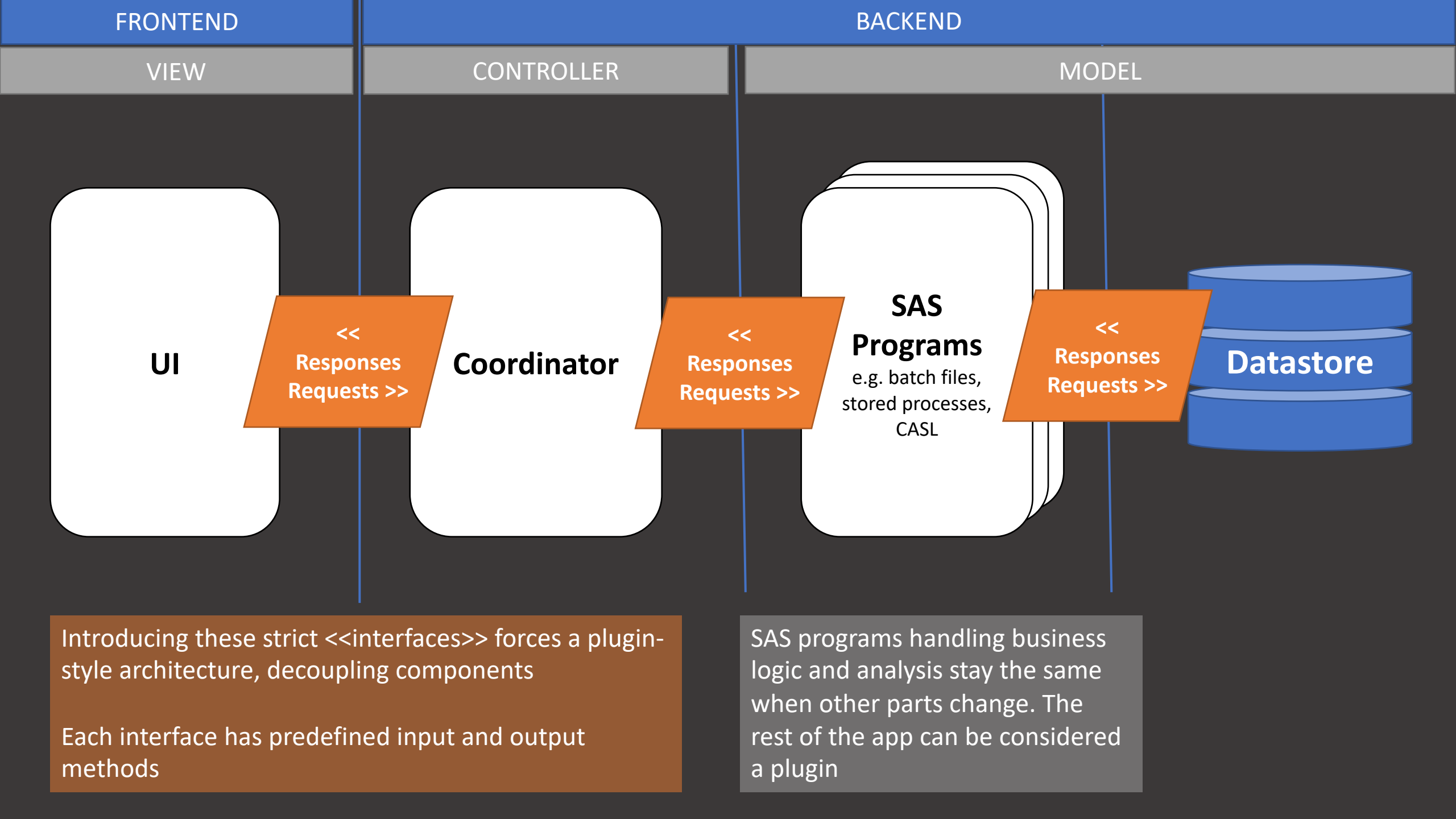


Solution 2

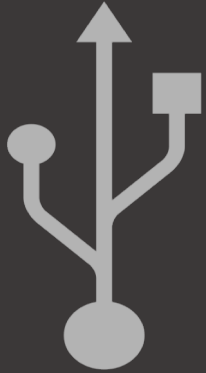
Separation by design between SAS (which handles business logic and secure data access) and the rest of the application

Proprietary SAS/business logic is safe since Client has no visibility of the back-end

If needed, the entire front end could be replaced without requiring any modification to SAS programs



Open Source adapters to interface SAS to App



Solution 2

A good example is Boemski's HTML5 Adapter for SAS (H54S)

- SAS Stored Processes Web-app framework
- SAS Dataset as native communication format
- Connects SAS to JavaScript

H54S available free on GitHub under GPLv3.
Open Source licenses deserve more understanding
and less fear in this industry



Environmental
Constraints Preventing
Proof-of-Concept

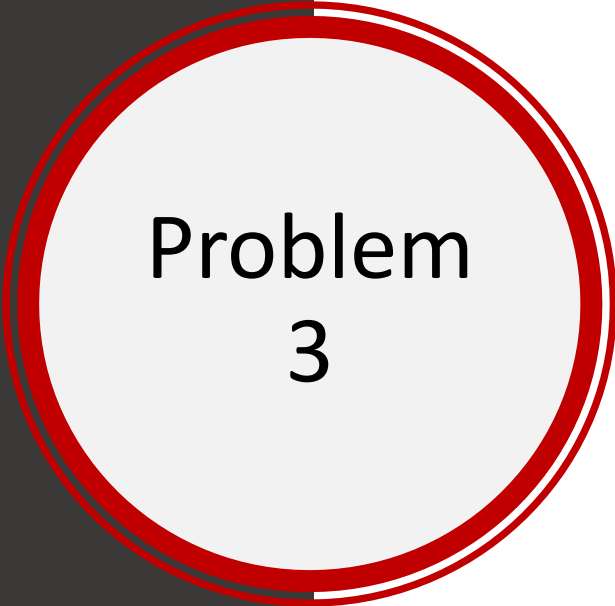
Wrong SAS Licenses

No JavaScript developers

No Stored Process server

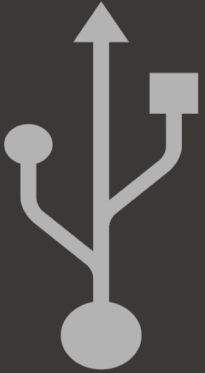
Virtual desktop with no admin rights

Scaling up would require rewrites



Problem
3

Toolbox of Adapters Implementing an Abstract Interface



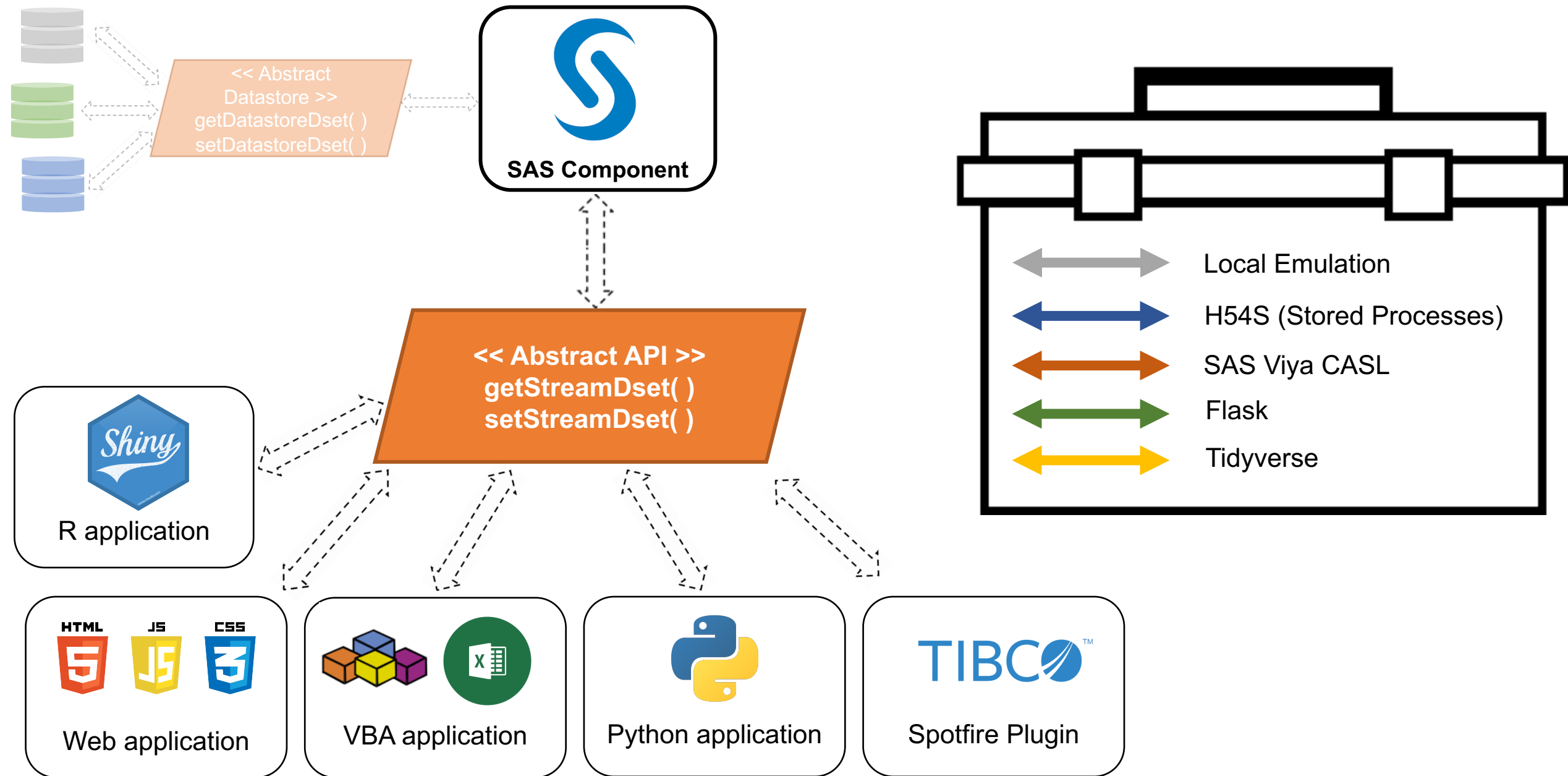
What if the same SAS program could work with multiple adapter types?

Multiple ways of connecting to SAS could share a common abstract interface

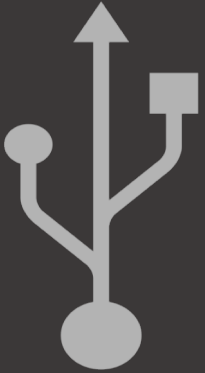
Solution
3

Open Source collaboration makes better use of the existing adapters while sharing improvements to the as collection

Toolbox of Adapters Implementing an Abstract Interface



**All adapters implement the same interface,
same code works with different connections**



Solution 3

```
getStreamDset(obj=, outdset=)
```

```
setStreamDset(obj=, indset=)
```

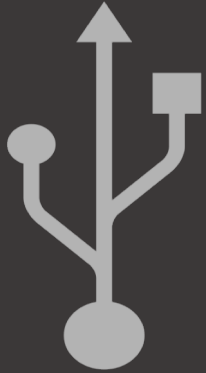
```
setupDatastore(name=, targ=)
```

```
teardownDatastore(name=)
```

```
getDatastoreDset(name=, obj=, outdset=)
```

```
setDatastoreDset(name=, obj=, indset=)
```

Develop a quick prototype, develop it further



Solution 3

Different connection types (like ODBC) can share common syntax

Other parts of the application can be treated as plugins. The entire front end could be swapped out without affecting the back-end SAS

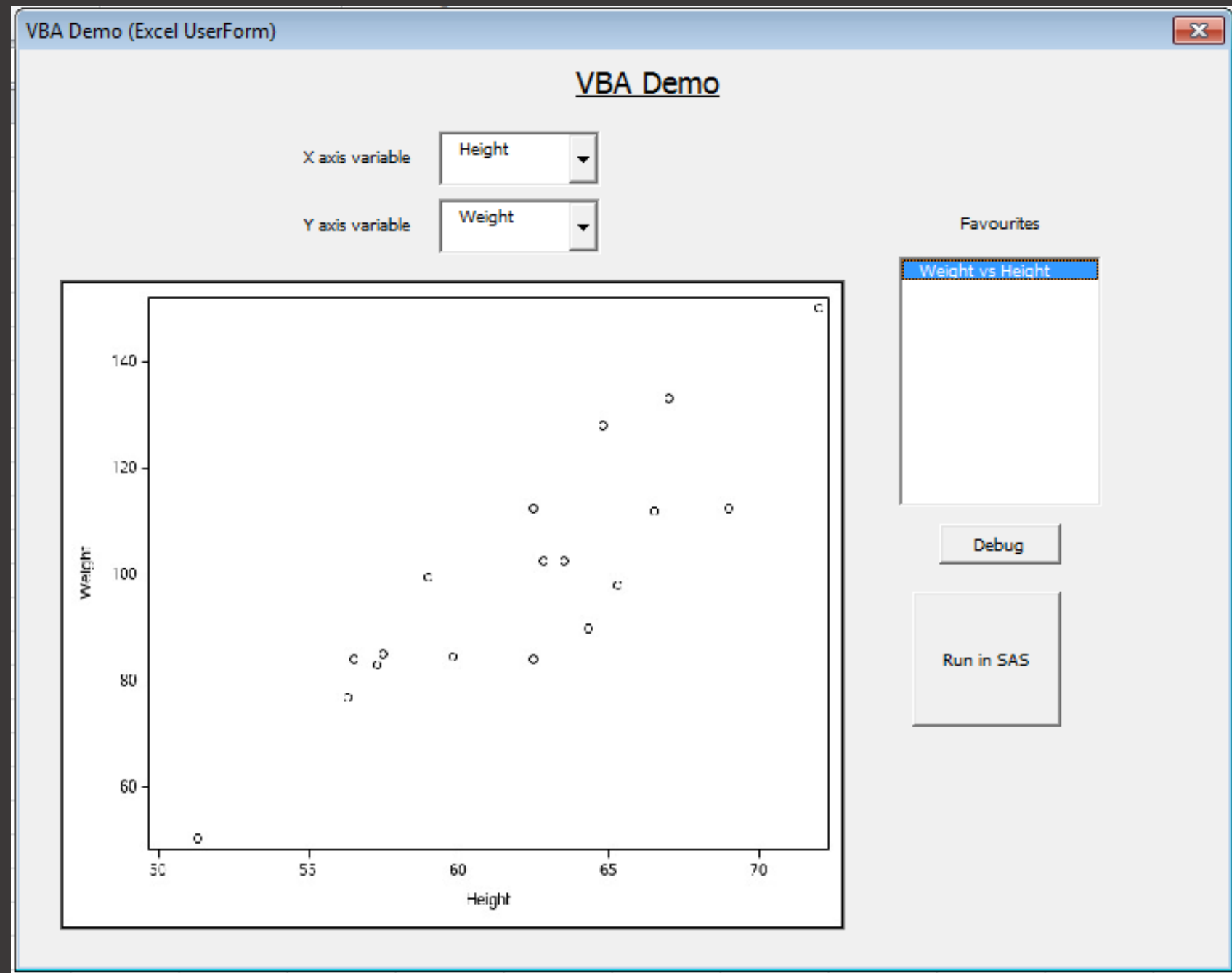
Save time by building on an existing framework



sas-appfitter
interface

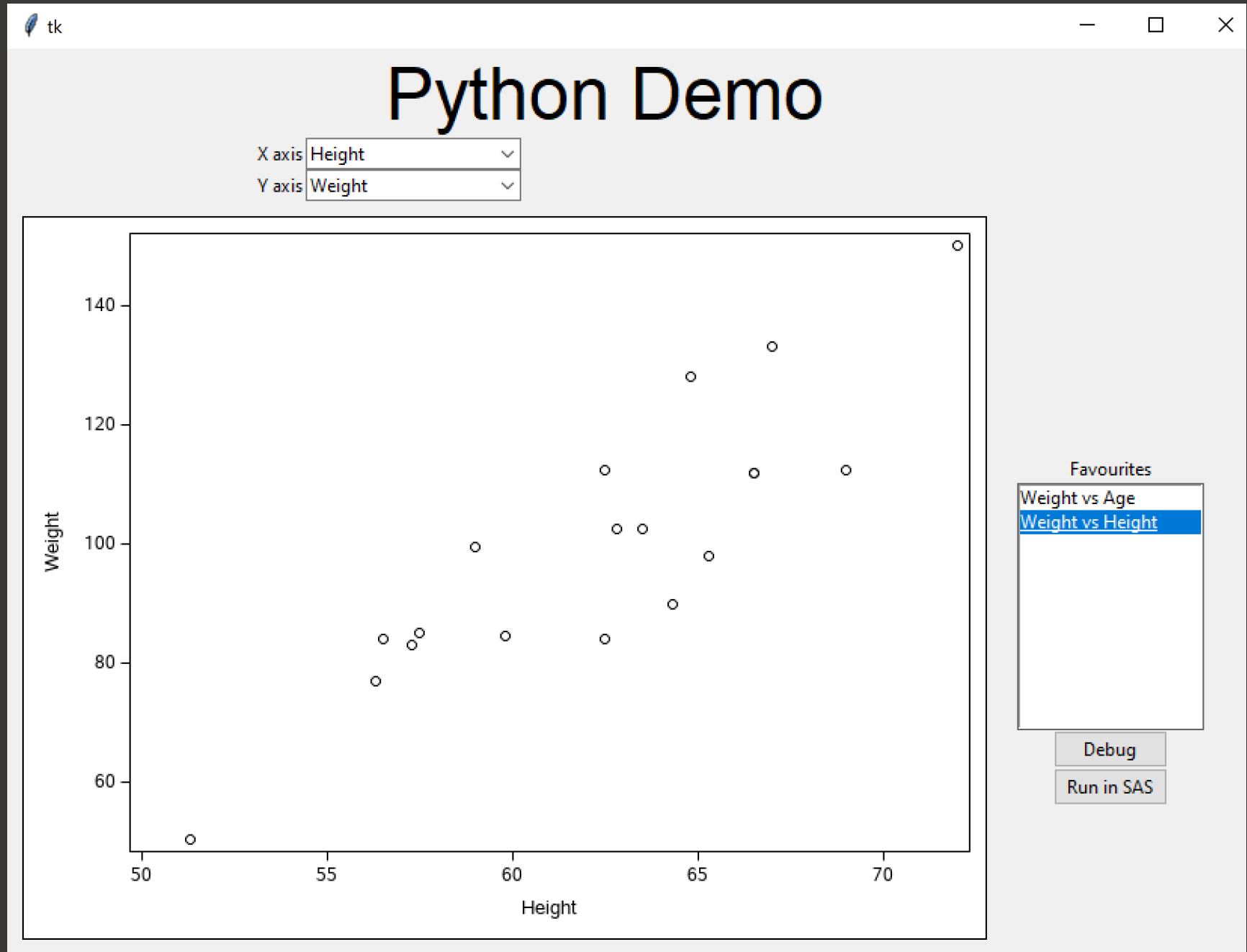
↔

Local Emulation





sas-appfitter
interface
↔
Local Emulation



Demo API payload returned by SAS Process (JSON serialised from SAS dataset)

```
{  
  "MyData": [  
    {  
      "status": 201,  
      "plottitle": "Weight vs Height",  
      "plotdesc": "This is an example plot generated by SAS",  
      "plotpath": "C:\\myapp\\output\\exampleplot.jpg"  
    }  
  ]  
}
```

Overview

Problem: tightly-coupled code

Solution: define interfaces

Problem: plumbing for prototype

Solution: Open Source interface adapters

Problem: different platforms

Solution: implement abstract interface

sas-appfitter project



Takeaways

Interface-first application development leads to well-defined architecture consisting of reusable components

Use Open Source frameworks! There is no point reinventing the wheel. Collaboration leads to superior solutions



Special thanks to

- Nikola Markovic
- Allan Bowe
- Giuseppe Di Monaco

Interface-first application development leads to well-defined architecture consisting of reusable components

Use Open Source frameworks! There is no point reinventing the wheel. Collaboration leads to superior solutions



Thank you for your attention

Jeremy Teoh

jeremyyteoh@gmail.com

[linkedin.com/in/jeremyteoh/](https://www.linkedin.com/in/jeremyteoh/)

sas-appfitter

github.com/TeMeta/sas-appfitter

Interface-first application
development leads to well-defined
architecture consisting of reusable
components

Use Open Source frameworks! There
is no point reinventing the wheel.
Collaboration leads to superior
solutions