CON3449

The Open Monolith

Keeping Your Codebase (and Your Headaches) Small

Michael Bar-Sinai
@michbarsinai
mbarsinai.com

Matthew Dunlap
@disbliss
sbgrid.org/about/staff/

@dataverseorg

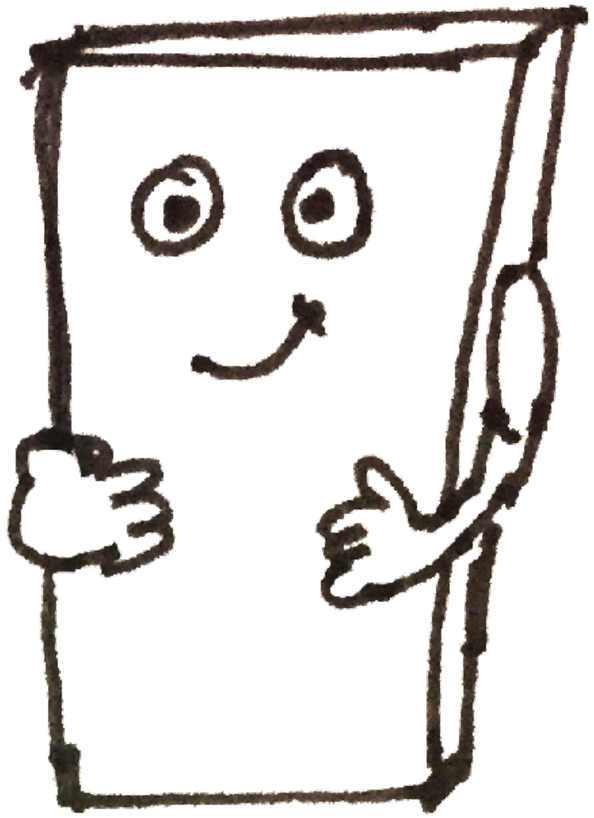
Large, monolithic codebase is hard to work with.

Large, monolithic codebase is hard to work with.

also, microservices are the new cool toy

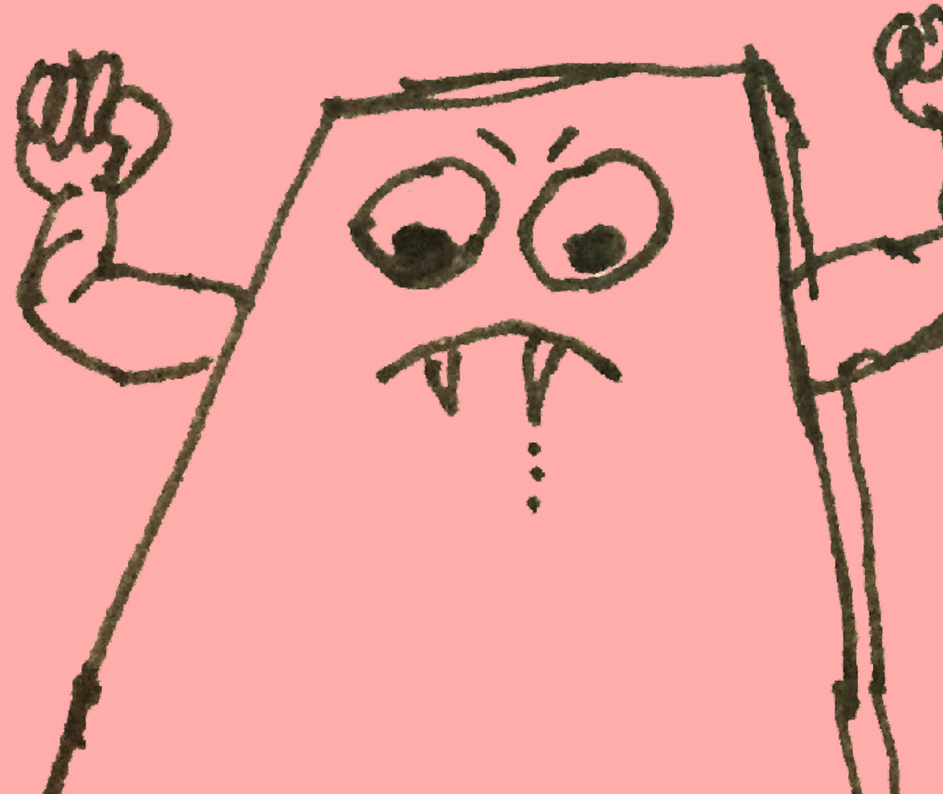
Monolith is Fun…

- Easy setup
- Entire system is versioned together
- **Static Typing** provides strong guarantees for compatibility between sub-systems.

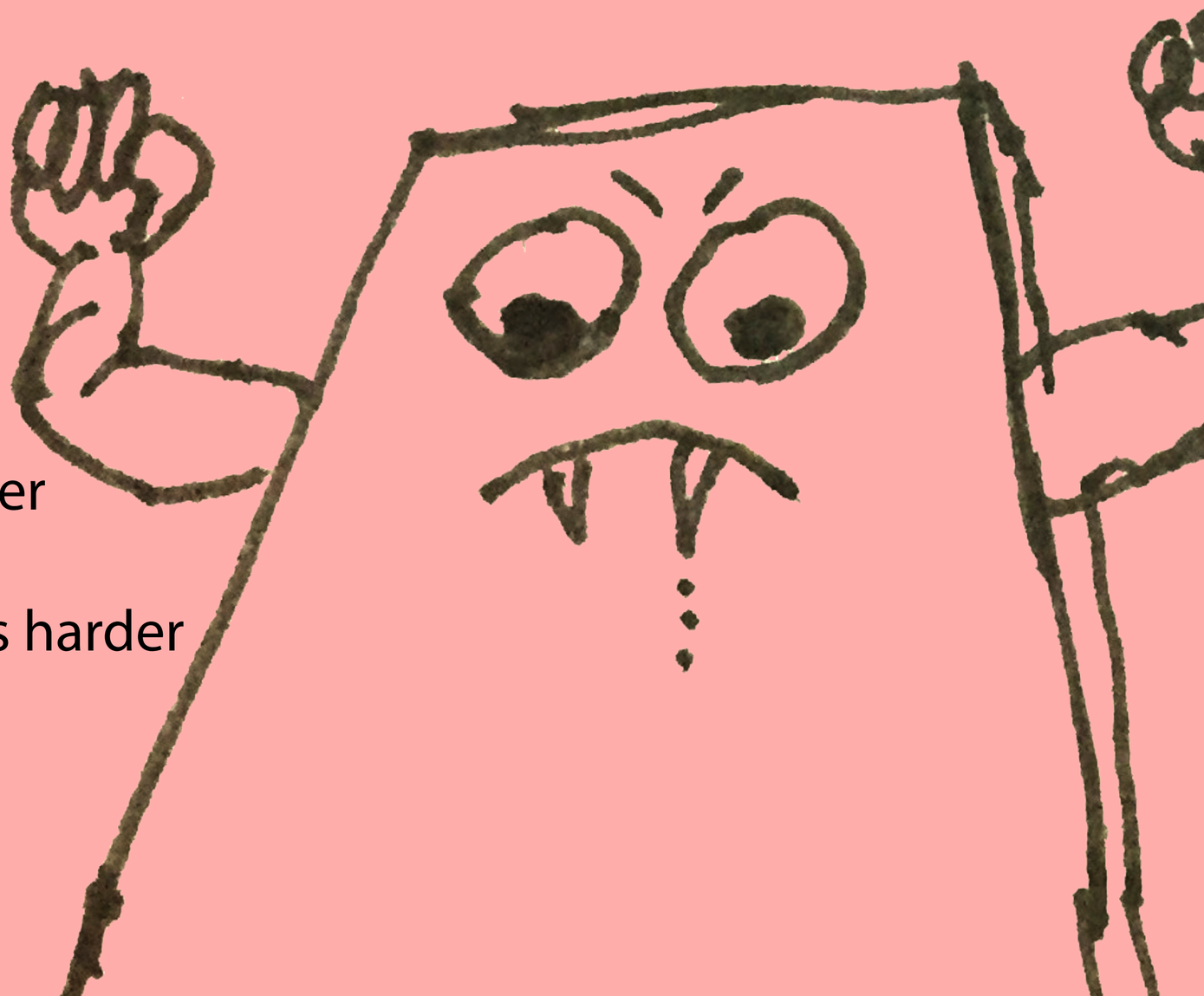


…Until it becomes Too Large

- Adding new features becomes slower
- Updating existing features becomes harder
- Regressions become common



…Until it becomes Too Large



- Adding new features becomes slower
- Updating existing features becomes harder
- Regressions become common

Microservices are No Silver Bullet

- Application Re-write
- Re-arrangement of the production environment
- Complexity still there, but distributed
- Added overhead for network, versioning, deployment
- More resilient, better throughput, etc.

Is there a middle way between monolith and microservices?

Can we break the codebase,
but not shatter it to little pieces?

Harvard Dataverse

https://dataverse.harvard.edu

Dataverse

About User Guide Support Sign Up Log In

Harvard Dataverse

Metrics 2,721,202 Downloads

Contact Share

Share, archive, and get credit for your data. Find and cite data across all research fields.

Search this dataverse... Find Advanced Search

+ Add Data

Dataverses (2,346)

Datasets (74,659)

Files (352,782)

Dataverse Category

Research Project (712)

Researcher (670)

Organization or Institution (194)

Journal (123)

Research Group (62)

More...

Metadata Source

1 to 10 of 77,005 Results

Sort

Replication Data for: Pivotal Politics and the Ideological Content of Landmark Laws"

Oct 2, 2017 - Journal of Public Policy Dataverse

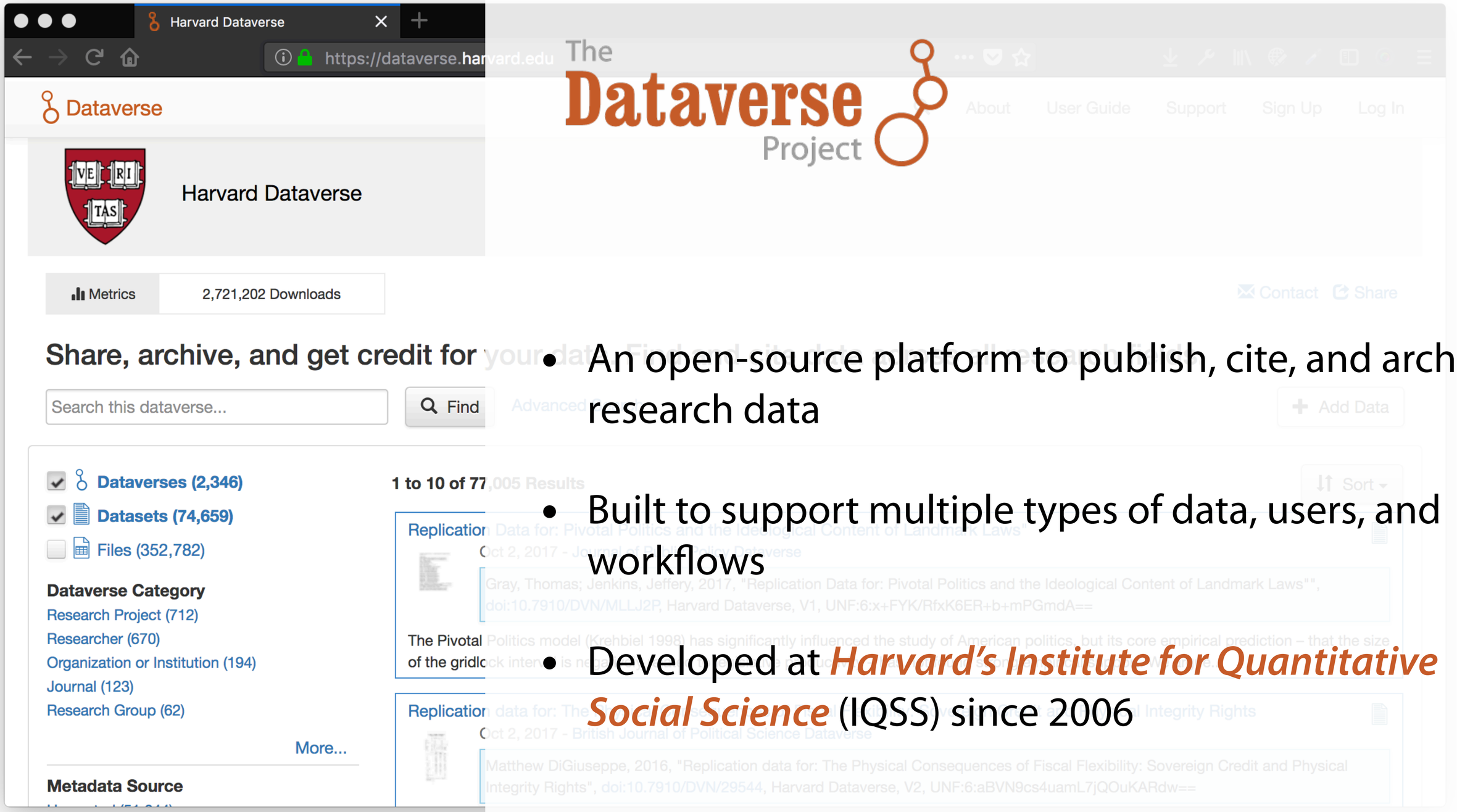
Gray, Thomas; Jenkins, Jeffery, 2017, "Replication Data for: Pivotal Politics and the Ideological Content of Landmark Laws"", doi:10.7910/DVN/MLLJ2P, Harvard Dataverse, V1, UNF:6:x+FYK/RfxK6ER+b+mPGmdA==

The Pivotal Politics model (Krehbiel 1998) has significantly influenced the study of American politics, but its core empirical prediction – that the size of the gridlock interval is negatively related to legislative productivity – has not found strong empirical support. We argue...

Replication data for: The Physical Consequences of Fiscal Flexibility: Sovereign Credit and Physical Integrity Rights

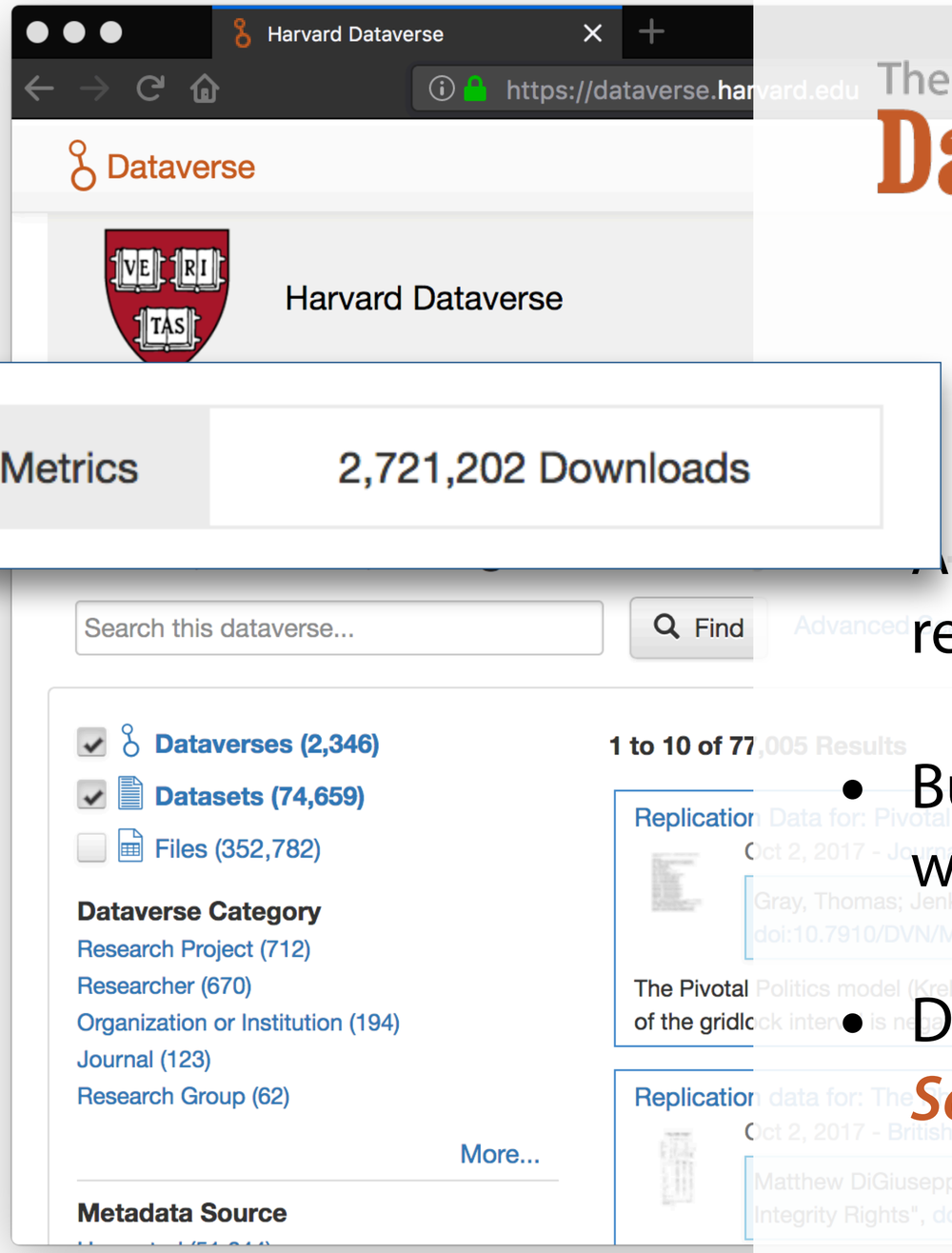
Oct 2, 2017 - British Journal of Political Science Dataverse

Matthew DiGiuseppe, 2016, "Replication data for: The Physical Consequences of Fiscal Flexibility: Sovereign Credit and Physical Integrity Rights", doi:10.7910/DVN/29544, Harvard Dataverse, V2, UNF:6:aBVN9cs4uamL7jQOuKARdw==



The Dataverse Project

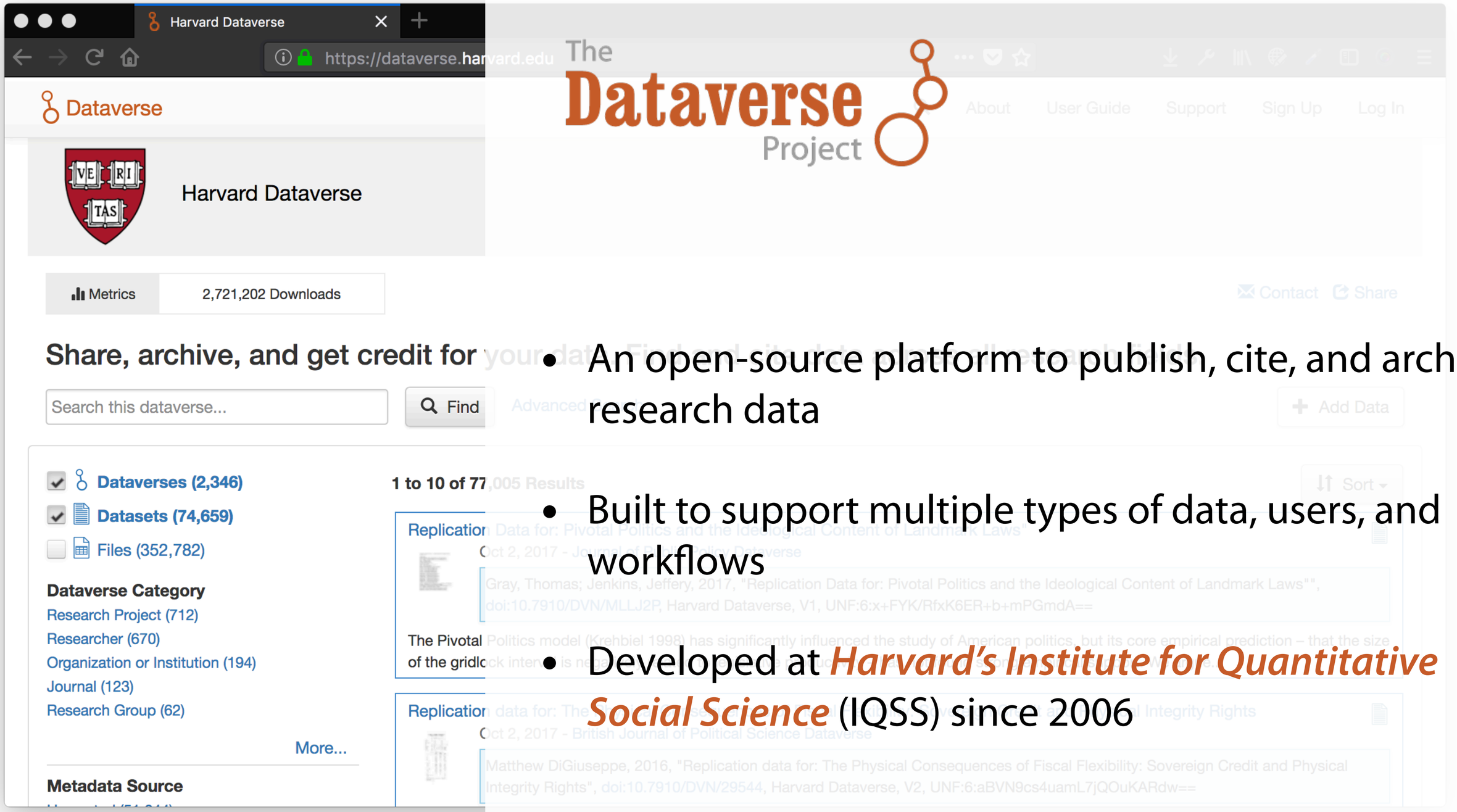
- An open-source platform to publish, cite, and archive research data
- Built to support multiple types of data, users, and workflows
- Developed at ***Harvard's Institute for Quantitative Social Science*** (IQSS) since 2006



The Dataverse Project

An open-source platform to publish, cite, and archive research data

- Built to support multiple types of data, users, and workflows
- Developed at ***Harvard's Institute for Quantitative Social Science*** (IQSS) since 2006



The Dataverse Project

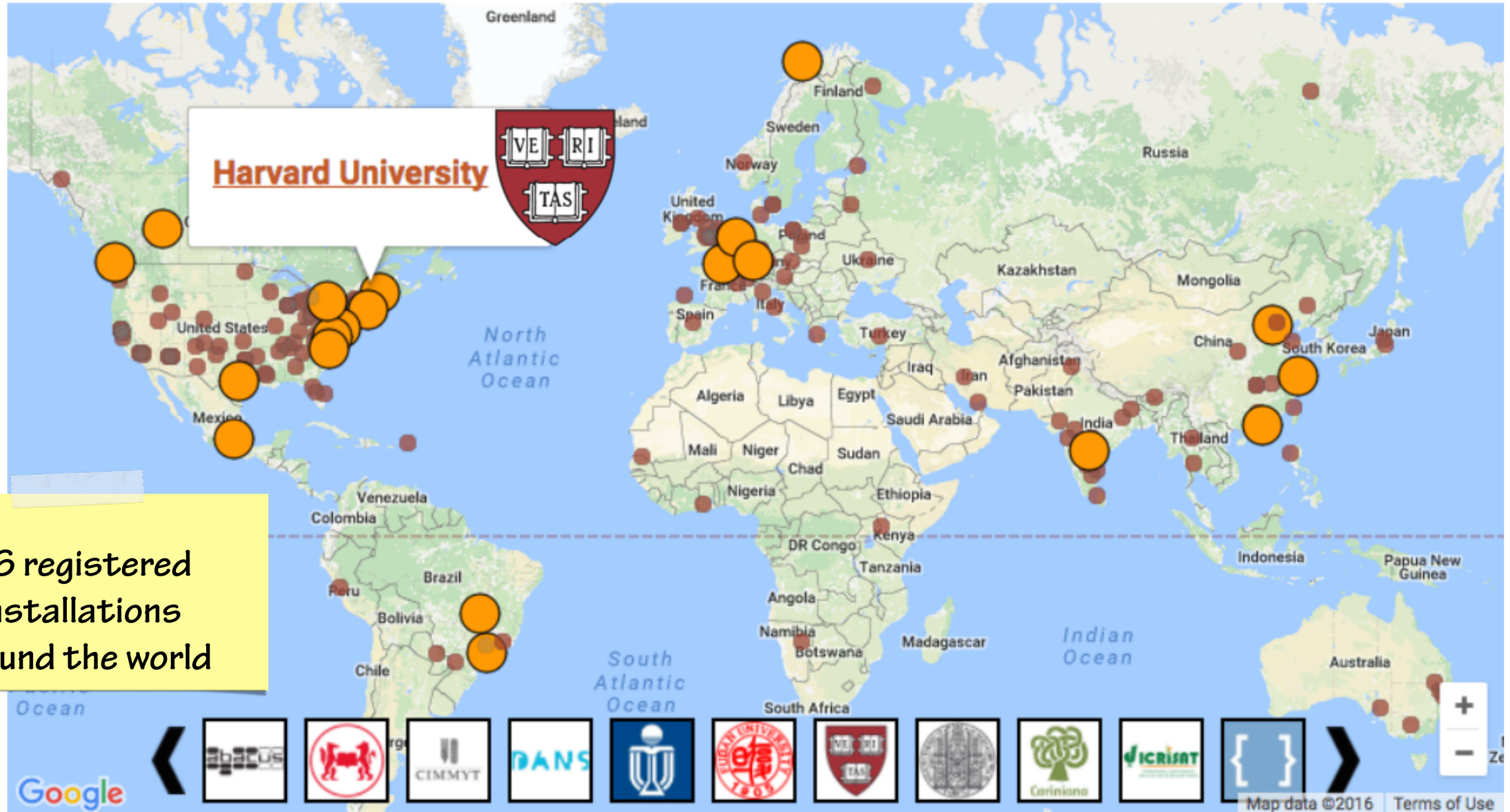
- An open-source platform to publish, cite, and archive research data
- Built to support multiple types of data, users, and workflows
- Developed at ***Harvard's Institute for Quantitative Social Science*** (IQSS) since 2006

Dataverse Core Features

Dataverse Core Features

- Persistent IDs / URLs
- Automatically Generated Citations with attribution
- Domain-specific Metadata
- Data and metadata Versioning
- File Storage:
  - local
  - OpenStack
  - AWS S3
- Permissions
- Access Controls
- Terms of Use
- Publishing Workflows
- Private URLs
- Upload/Download via browser
- Upload/Download via rsync
- Upload/Download via Dropbox
- Multiple Sign In options
  - Native
  - Shibboleth
- OAuth (ORCID)
- Dataverses within Dataverses
- Branding
- Widgets
- APIs
  - SWORD
  - Native
- Harvesting (OAI-PMH)
  - Client
  - Server

Dataverse Community



26 registered installations around the world

Dataverse Community

- Hundreds of members of the Dataverse Community - developers, researchers, librarians, data scientists
  - Dataverse Google Group
  - Dataverse Community Calls
  - Dataverse Community Meeting
- 14,000+ Registered researchers on the Harvard Dataverse alone

Supporting the Dataverse Community

- **Installation-based needs**
  - Branding and customizations
  - Language support
- **Subject-based needs**
  - See next slides

Subject-based Needs

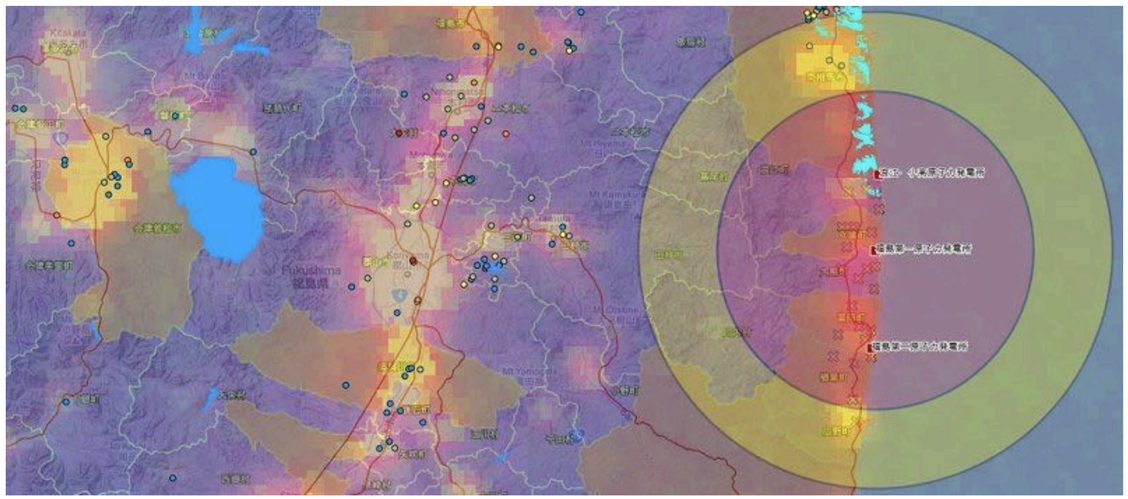
Quantitative Social Science

WorldMap
HARVARD UNIVERSITY

- Original use case
- Tabular Data / TwoRavens
- WorldMap

Zelig

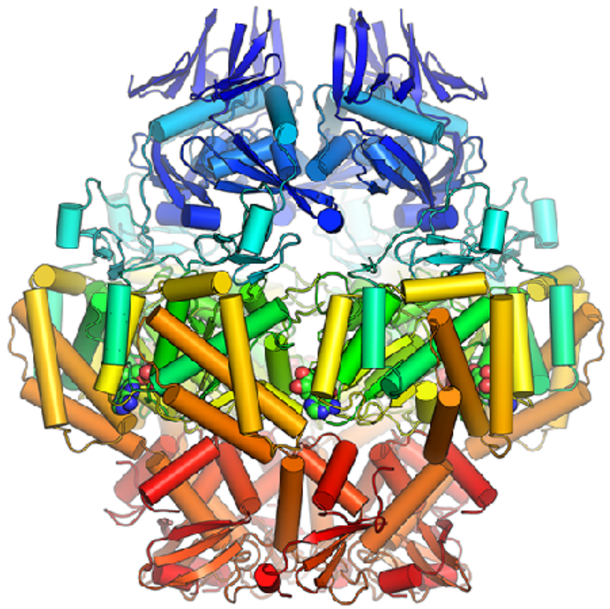
TwoRavens

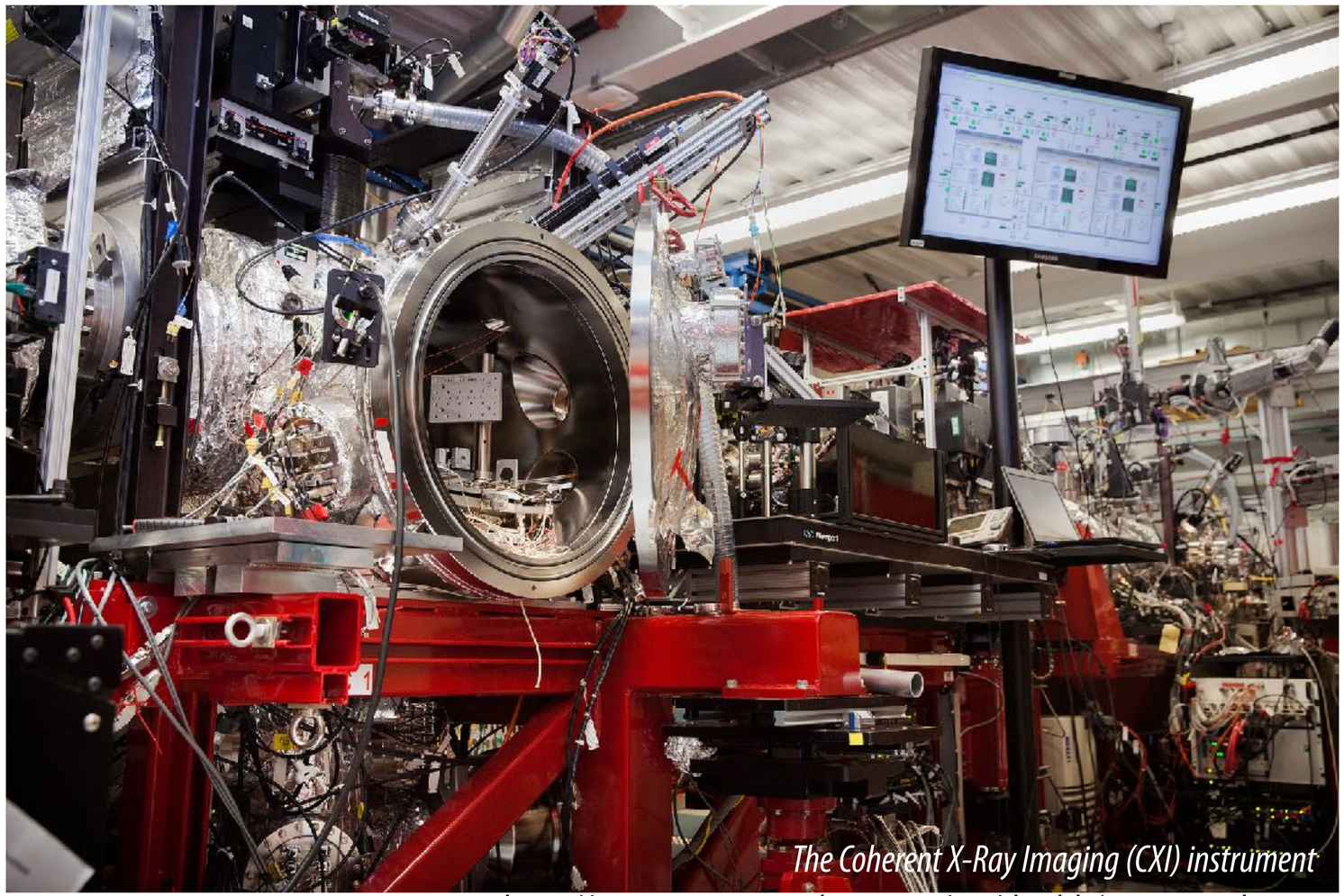


Subject-based Needs

Structural Biology

- Large data uploads
- Visualization support



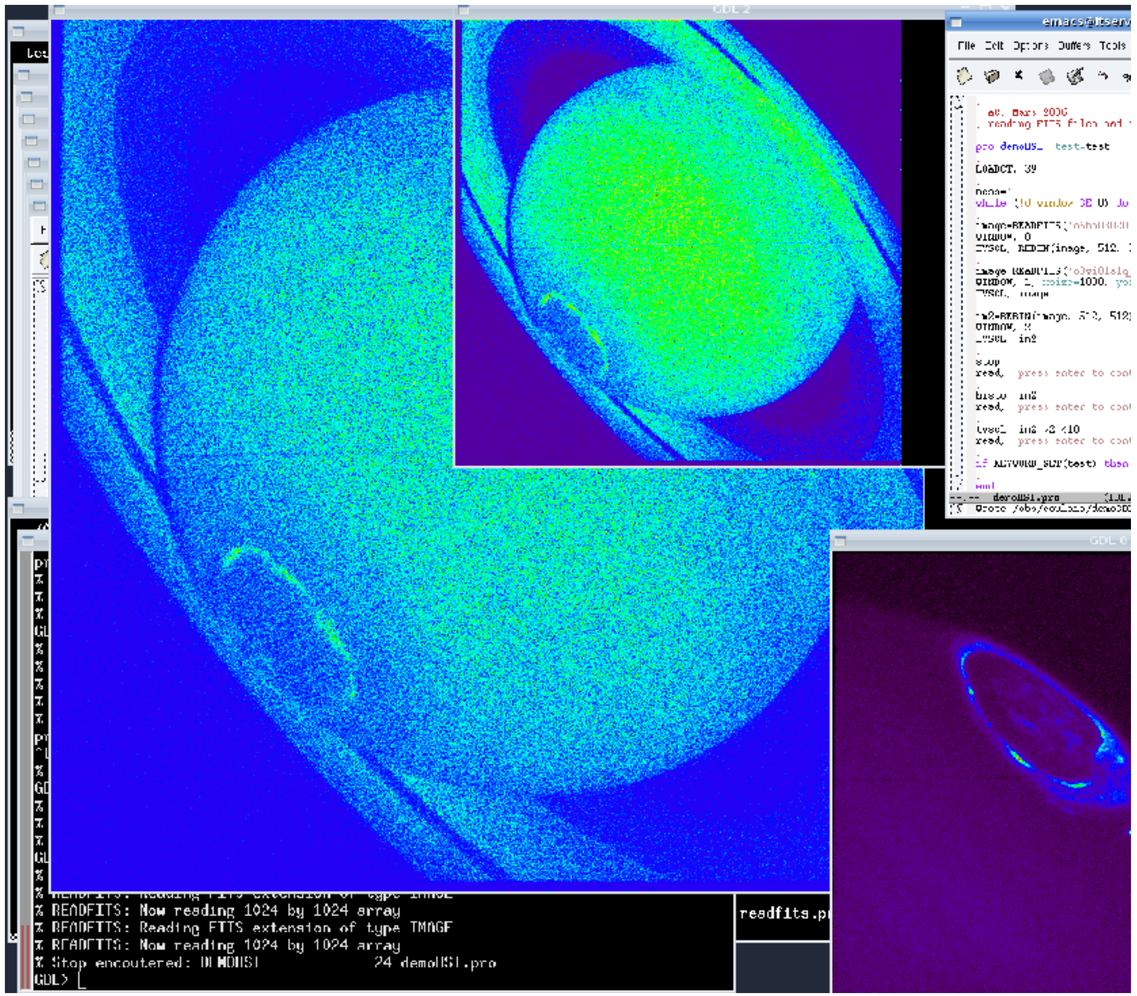


The Coherent X-Ray Imaging (CXI) instrument

https://sciencesprings.wordpress.com/tag/slac-lcls/#jp-carousel-39067

Subject-based Needs

Other Fields



- Astronomy: FITS format
- Expanding file format support
- Redhat system performance data

(Some) Collaborations

- **SBGrid Data** Large Data and Support

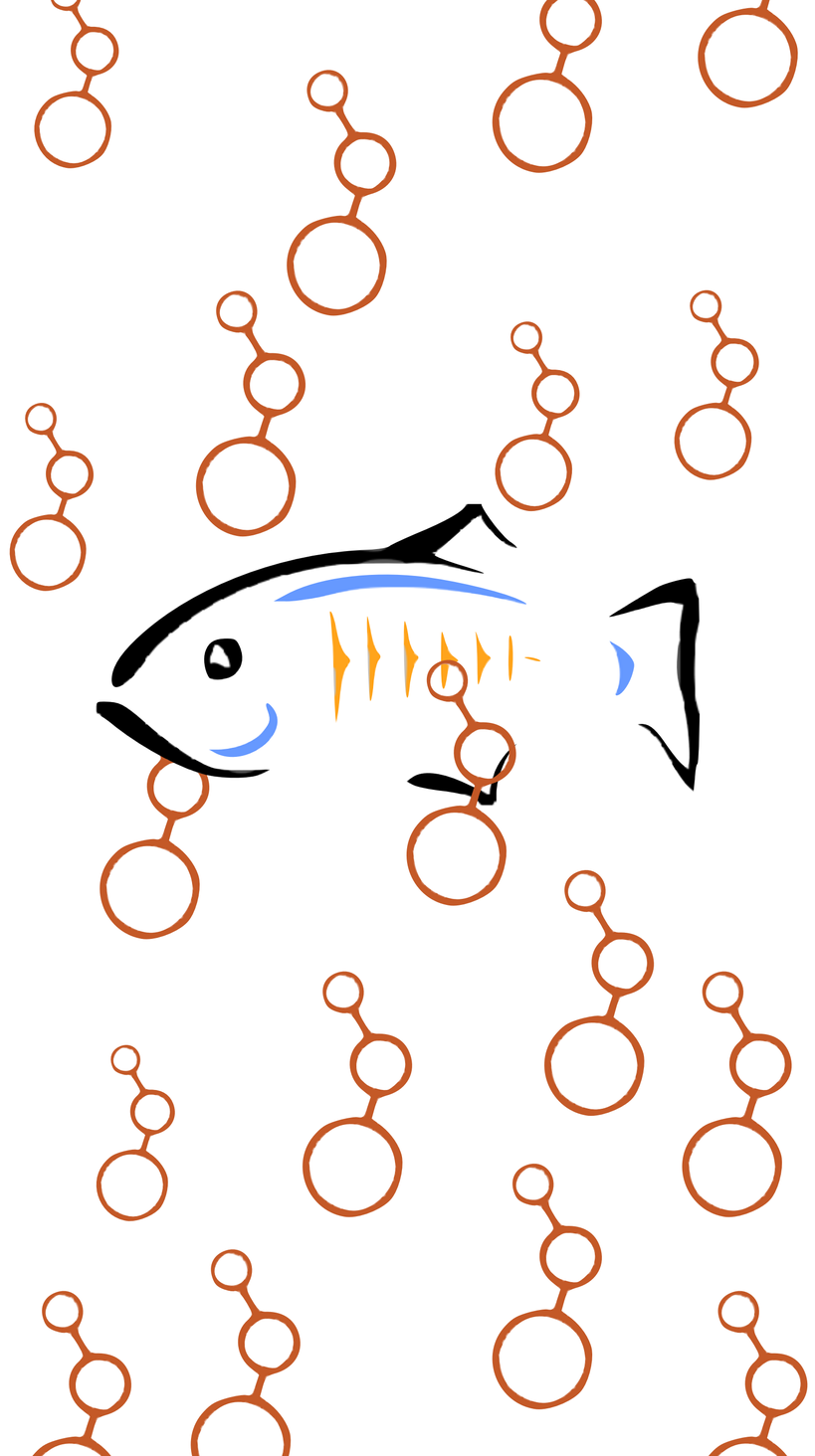
- **Massachusetts Open Cloud** Big Data Storage and Compute Access (OpenStack)

- **DANS/CIMMYT** Handles Support

- **ResearchSpace** API Java Client Library
- **W3C PROV** Provenance (soon)

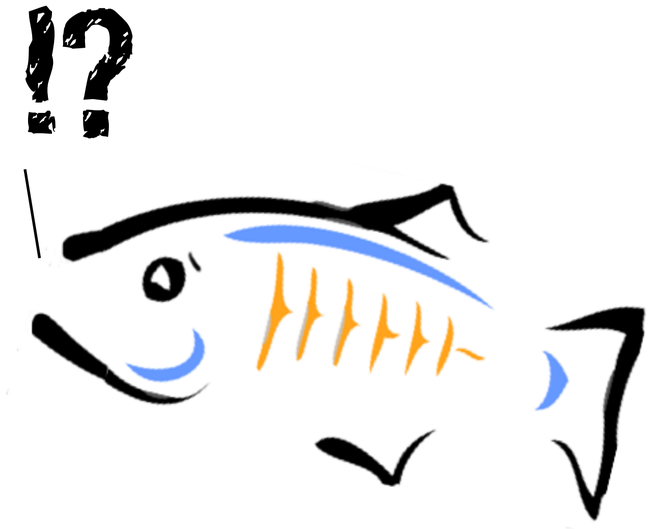
Technology

- Glassfish Server 4.1
- Java SE8
- Java EE7
  - Presentation: JSF (PrimeFaces), RESTful AP
  - Business: EJB, Transactions, Asynchronous, Timers, Adapted Command Pattern
  - Persistence: JPA (Entities), Bean Validation
- Storage: Postgres, Solr, File System / Swift / S3



Dataverse Codebase

```
App:    100,173 SLoC
Test:    16,081 SLoC
Total: 116,254 SLoC
generated using David A. Wheeler's 'SLOCCount'.
```



⚠️ Most functionality implemented in main codebase

HELP

- Goal:
  Offload Work from Core Team to Willing Community Members.

  - Lower bar needed to support community involvement

  - Facilitate Installation Customization

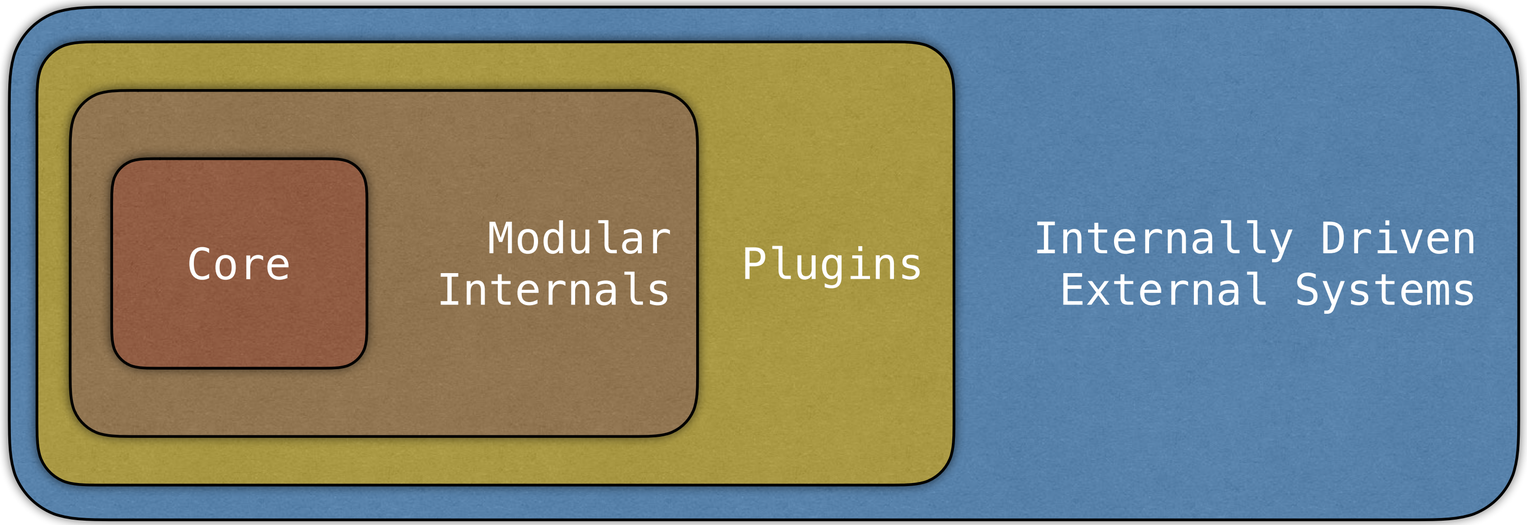
HELP

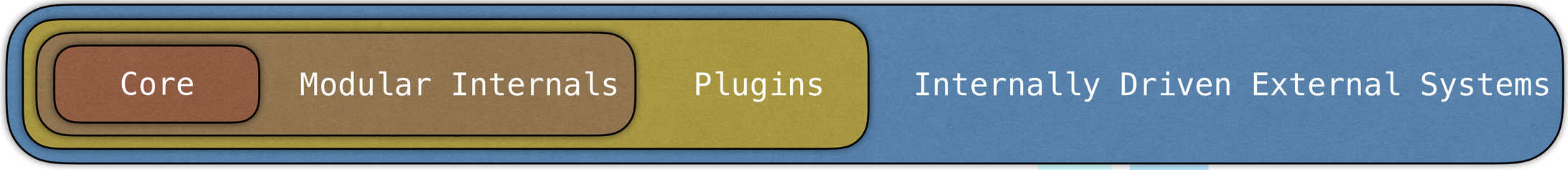
- Goal:
  Offload Work from Core Team to Willing Community Members.

  - Lower bar needed to support community involvement
  - Facilitate Installation Customization

WE NEED MODULARITY.

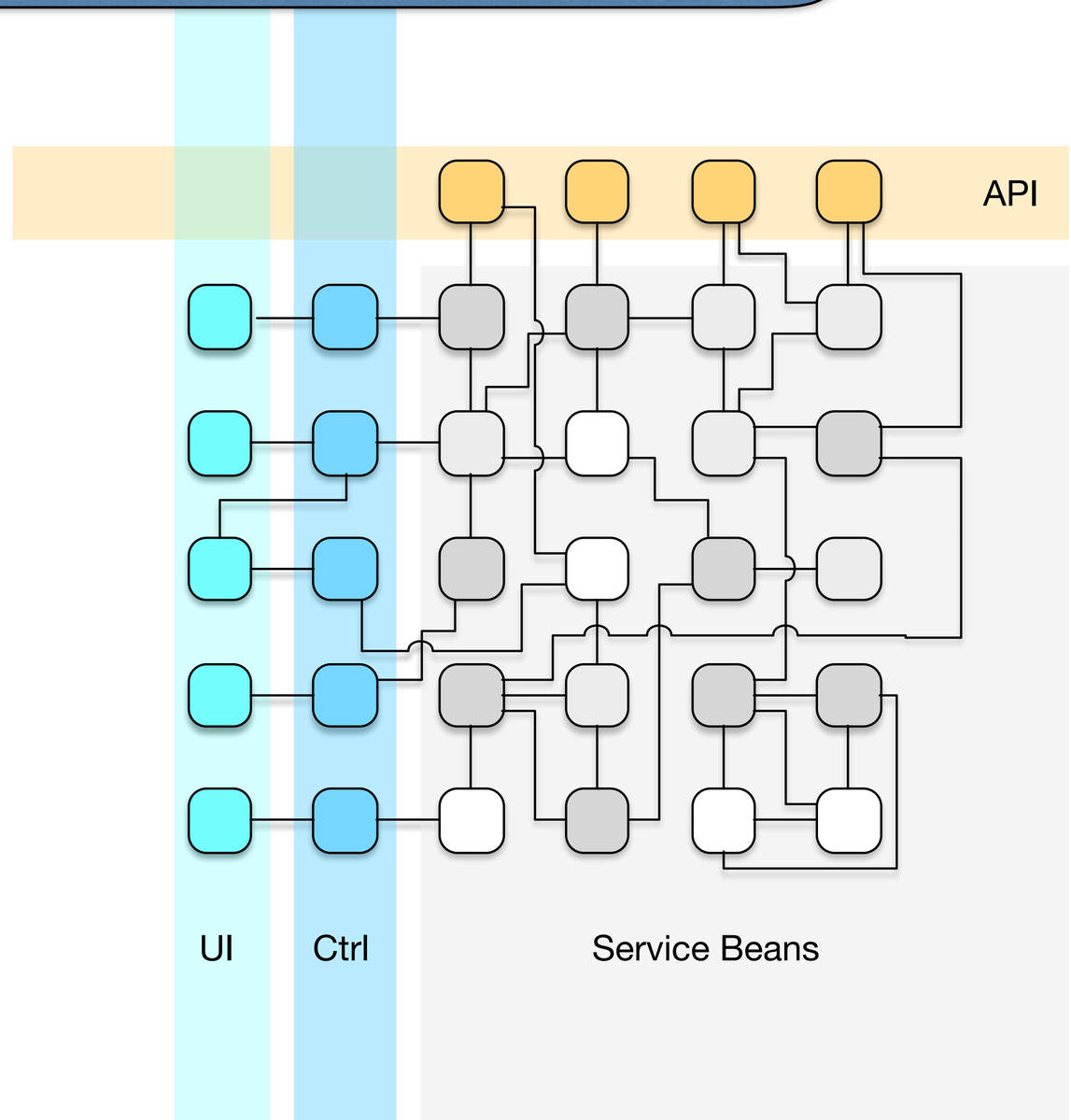
Modularity Levels

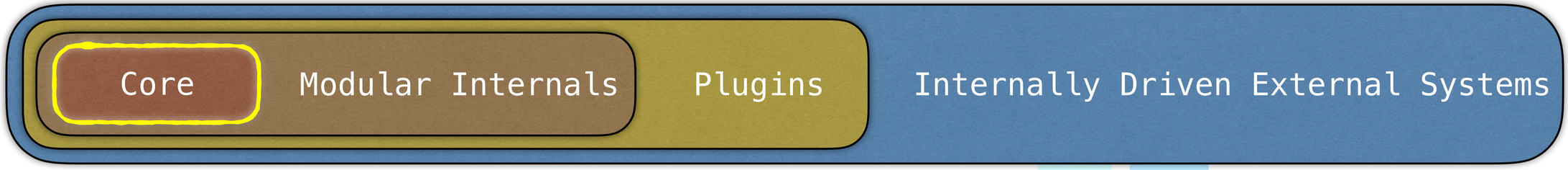




Core

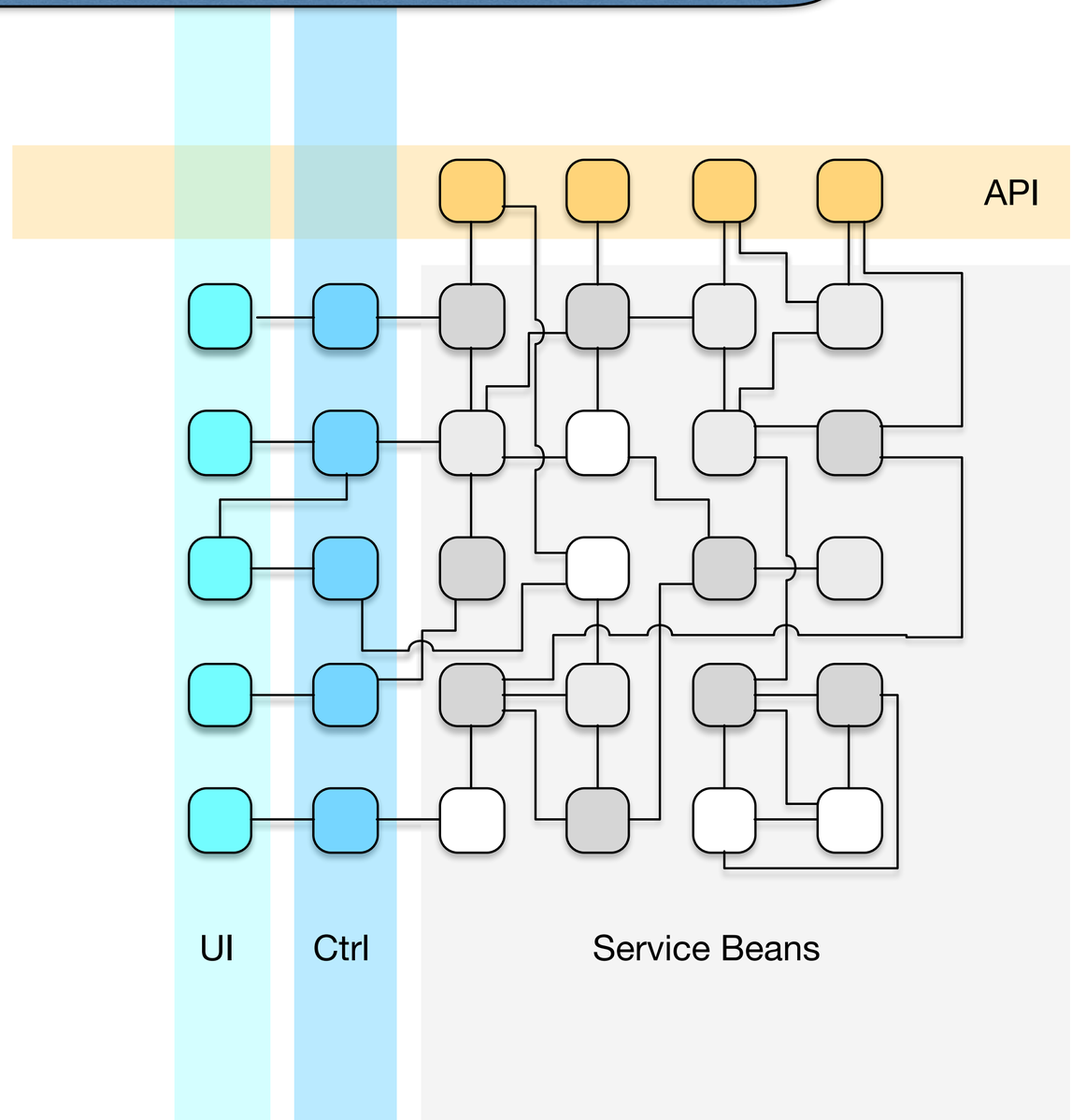
- Layered Code
- Lots of inter-related parts, high-connectivity between various beans.
- Not really modular

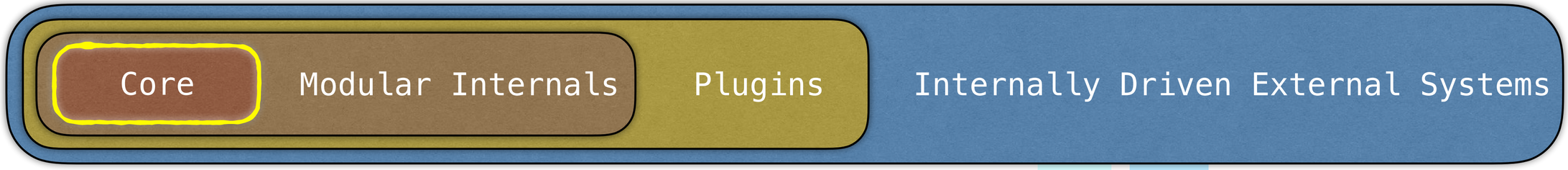




Core

- Layered Code
- Lots of inter-related parts, high-connectivity between various beans.
- Not really modular

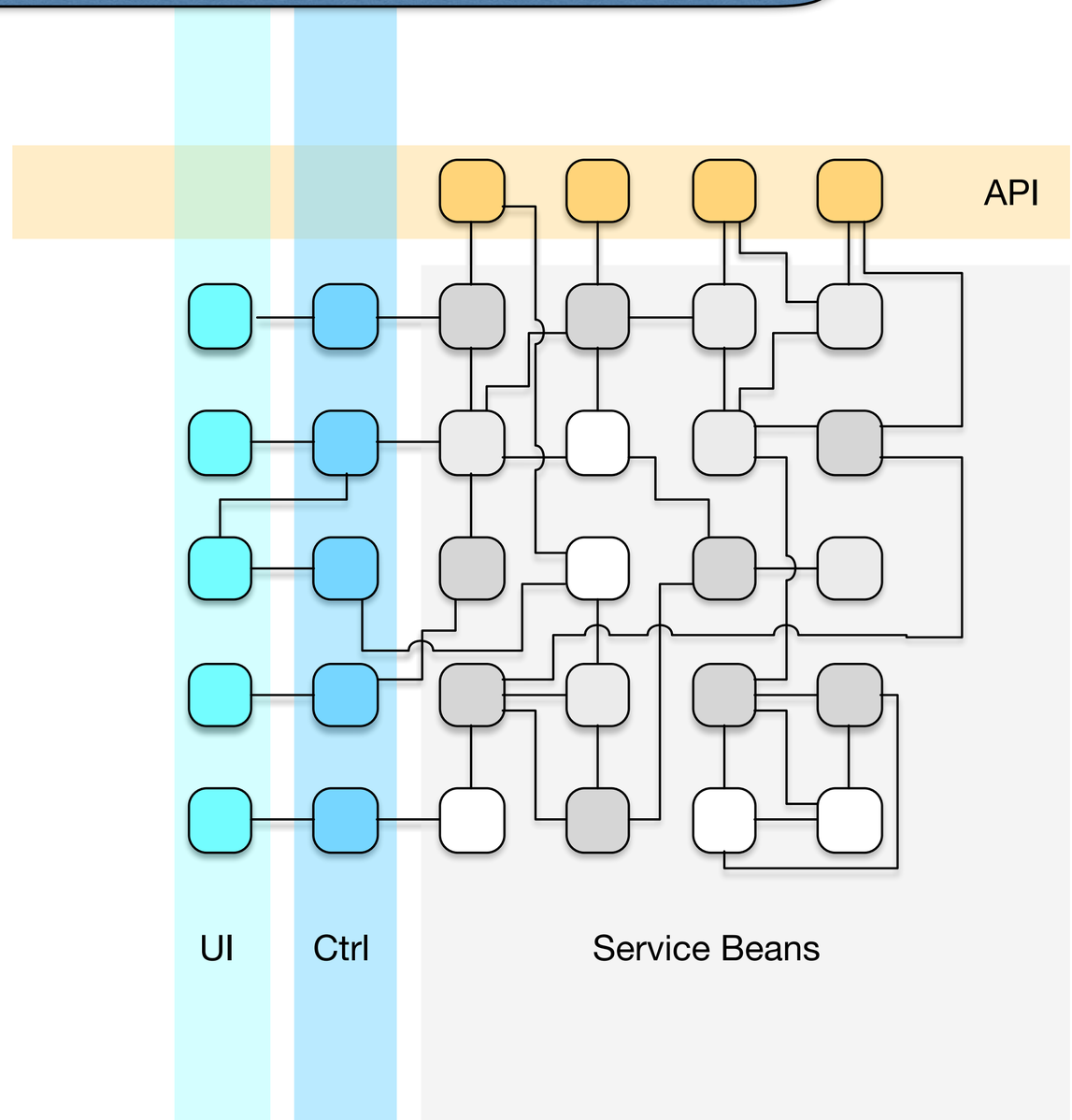


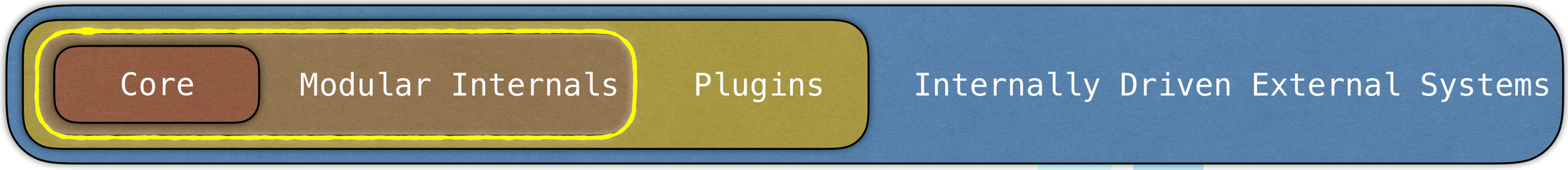


Core

- Layered Code
- Lots of inter-related parts, high-connectivity between various beans.
- Not really modular

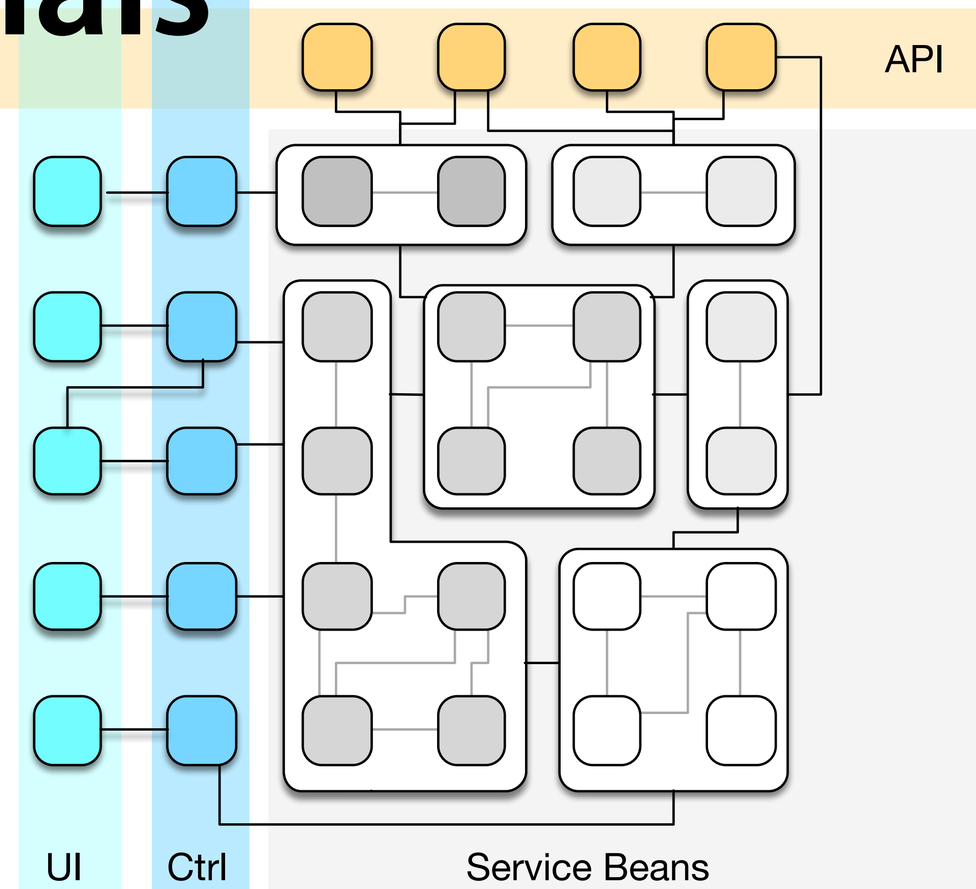
✘ Extension via fork-and-forget





Modular Internals

- Functionalities are implemented in *cohesive modules* with clear interface to the rest of the code
- Even though same code base, modules *don't get to make assumptions*
- Multiple patterns for this
- Jigsaw Helps



The Service Provider Interface (SPI)

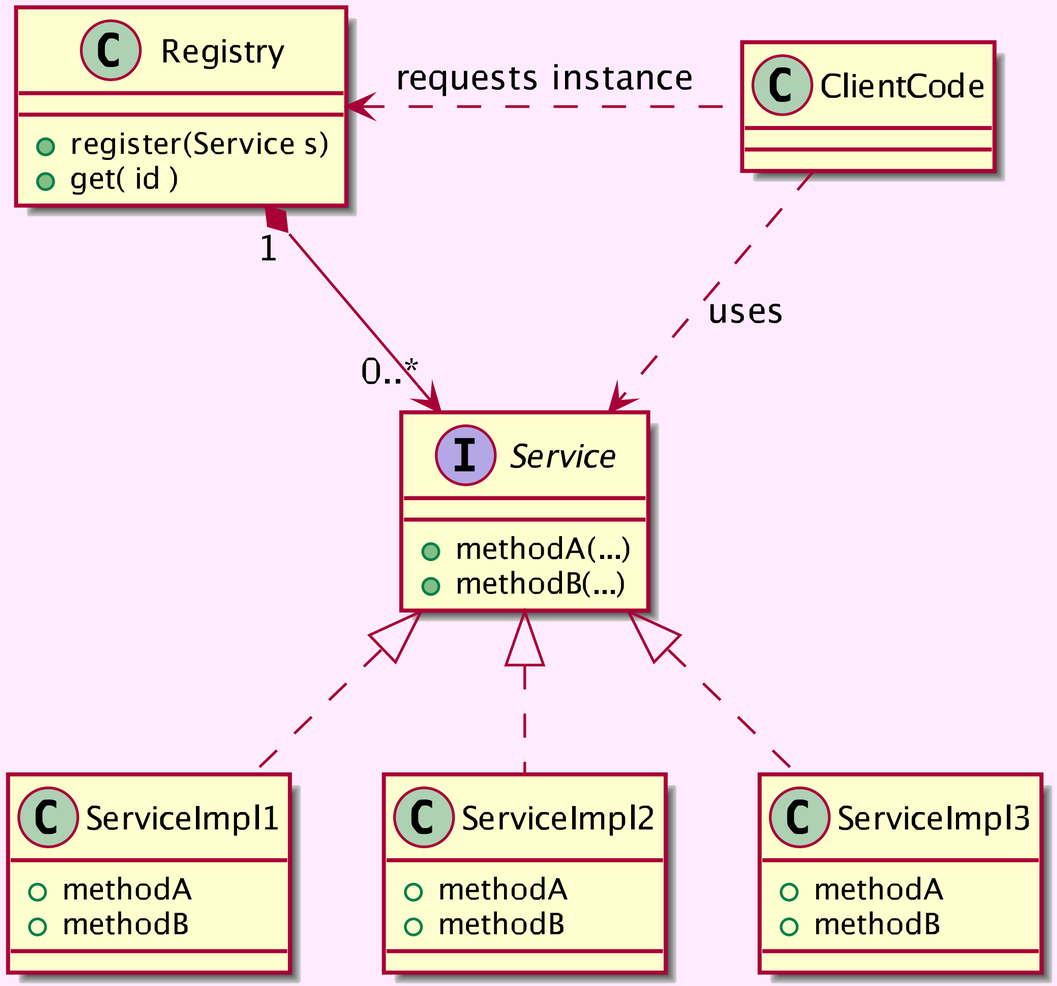
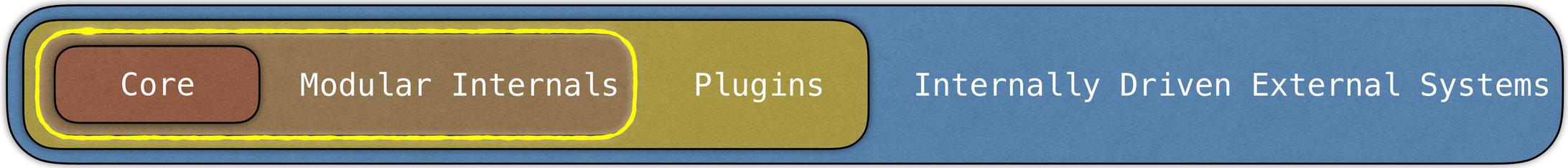


Diagram created using PlantUML

SPIs in Dataverse 4.x

```java
@PostConstruct
public void startup() {

    // First, set up the factories
    try {
        registerProviderFactory( new BuiltinAuthenticationProviderFactory(builtinUserServiceBean, passwordValidatorService) );
        registerProviderFactory( new ShibAuthenticationProviderFactory() );
        registerProviderFactory( new OAuth2AuthenticationProviderFactory() );

    } catch (AuthorizationSetupException ex) {
        logger.log(Level.SEVERE, "Exception setting up the authentication provider factories: " + ex.getMessage(), ex);
    }
```

❤️🧊 *actual production code*



Modular Internals

Codebase

- Size remains roughly the same

✔ Extension via pull-requests

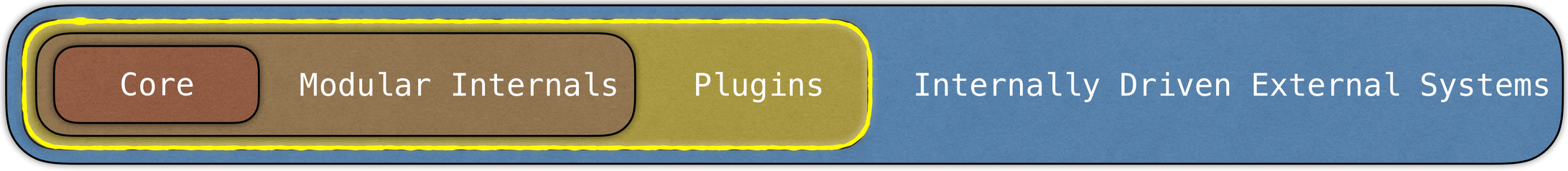
Headache

✔ Can **flip the ignorance switch** on module internals when not within a module

✔ Improves **code comprehension**

✔ **Less merge conflicts** between team members / collaborators

✔ **Internal** change

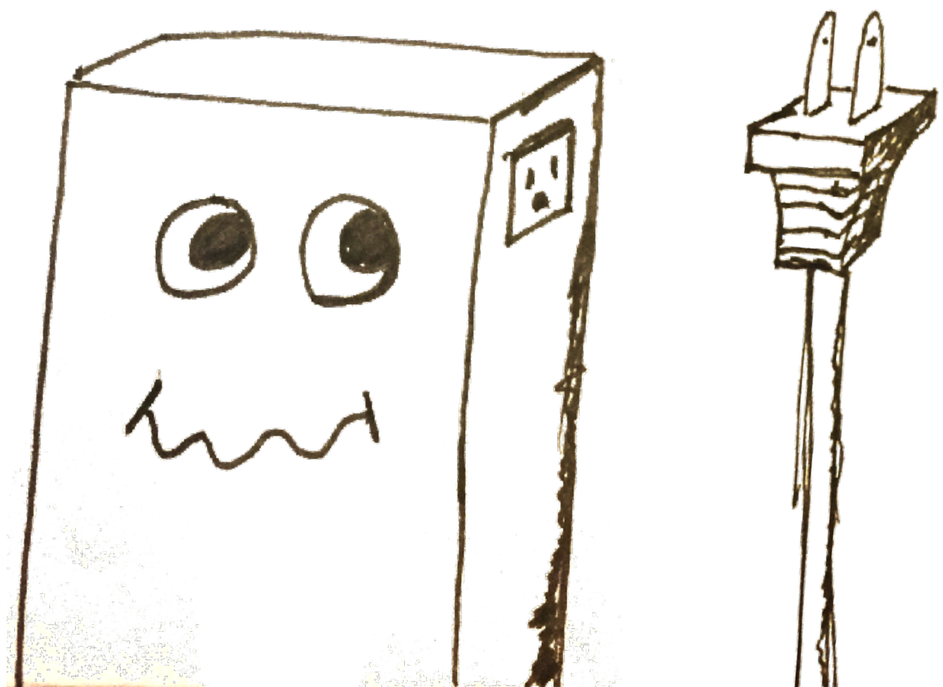


Plugins

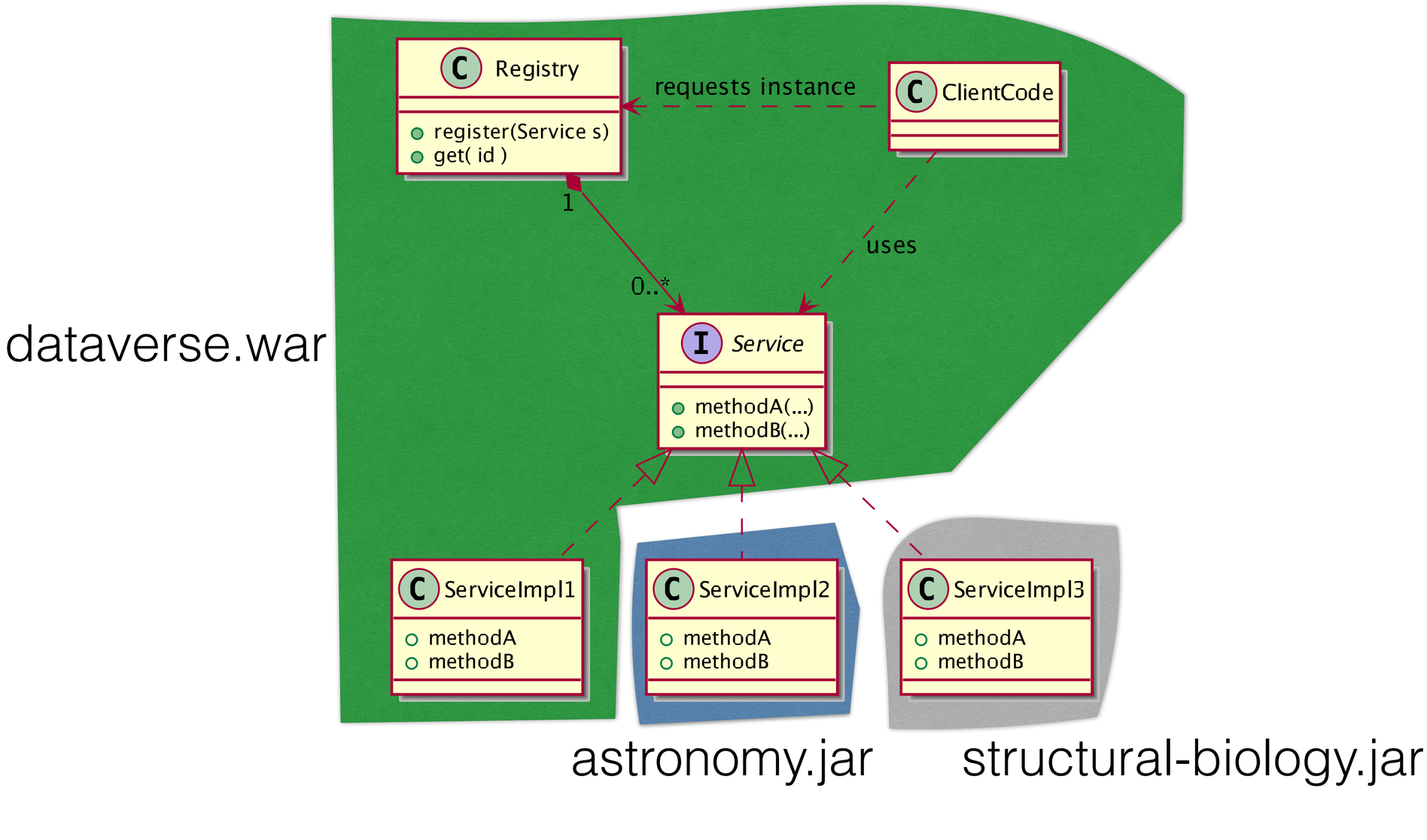
- Third parties implement SPIs, pack them in .jar files, and declare which services they provide
  Using META-INF/services or Project Jigsaw modules

- Core application team adds functionality to dynamically load implementation from .jar files

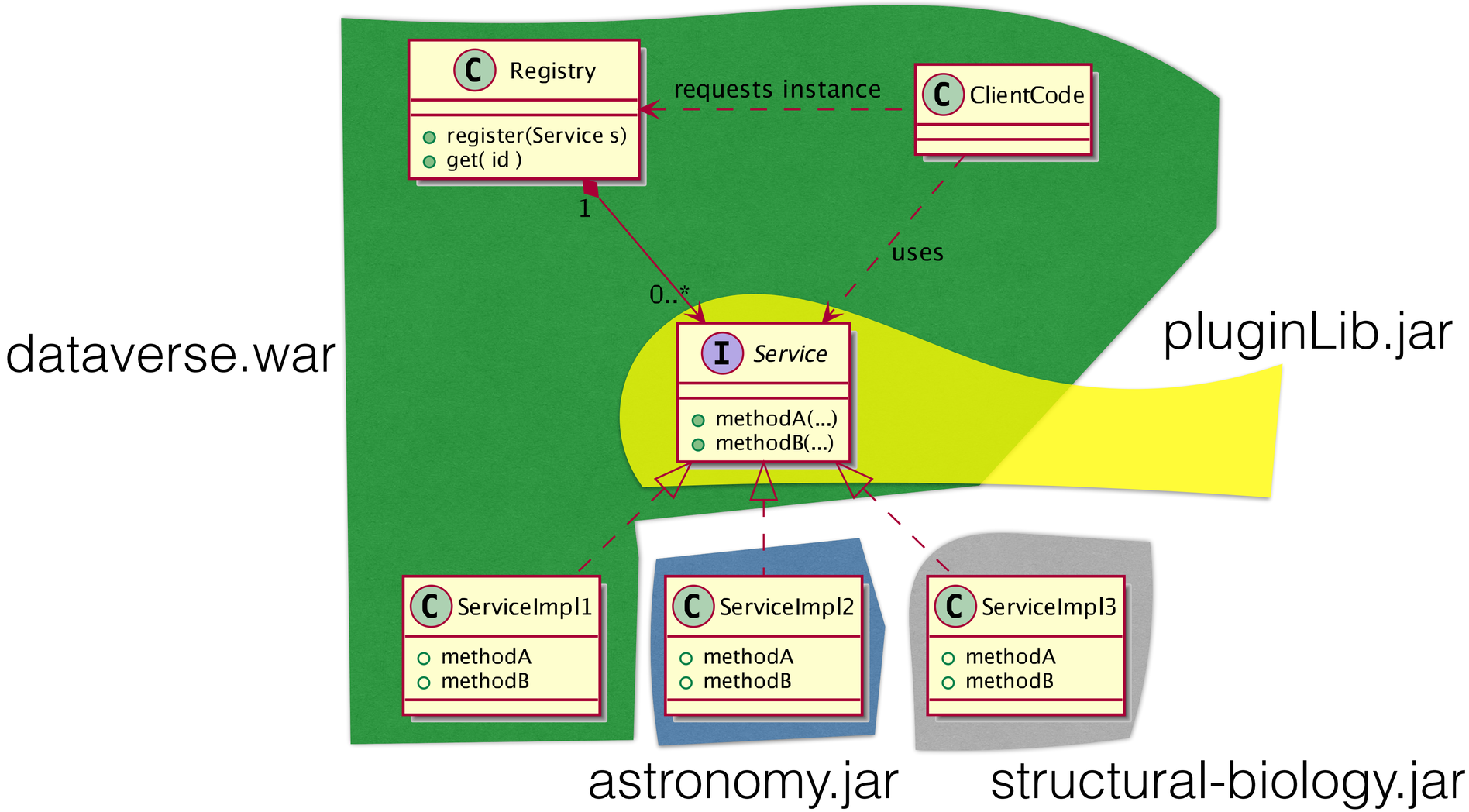
- Admins add .jar files to installations' classpath



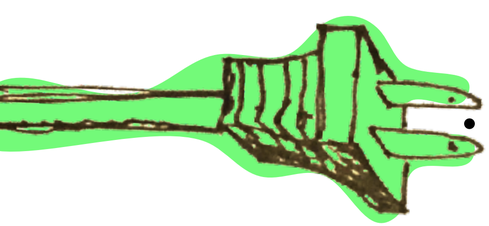
SPI + PlugIns


Registry
register(Service s)
get(id)
requests instance
ClientCode
1
0..*
uses
Service
methodA(...)
methodB(...)
ServiceImpl1
methodA
methodB
ServiceImpl2
methodA
methodB
ServiceImpl3
methodA
methodB
dataverse.war
astronomy.jar
structural-biology.jar

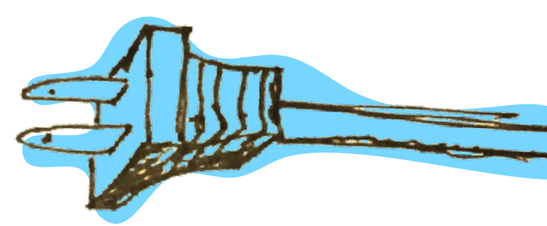
SPI + PlugIns


Registry
register(Service s)
get(id)
requests instance
ClientCode
1
0..*
uses
Service
methodA(...)
methodB(...)
ServiceImpl1
methodA
methodB
ServiceImpl2
methodA
methodB
ServiceImpl3
methodA
methodB
dataverse.war
pluginLib.jar
astronomy.jar
structural-biology.jar

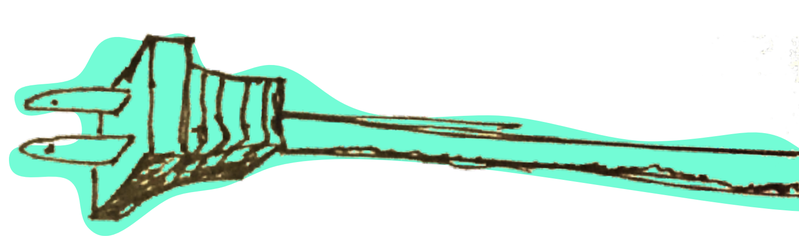
Writing Plugins



- Put classes available to plugins on a new .jar



  - Alas, that's another public API you need to maintain

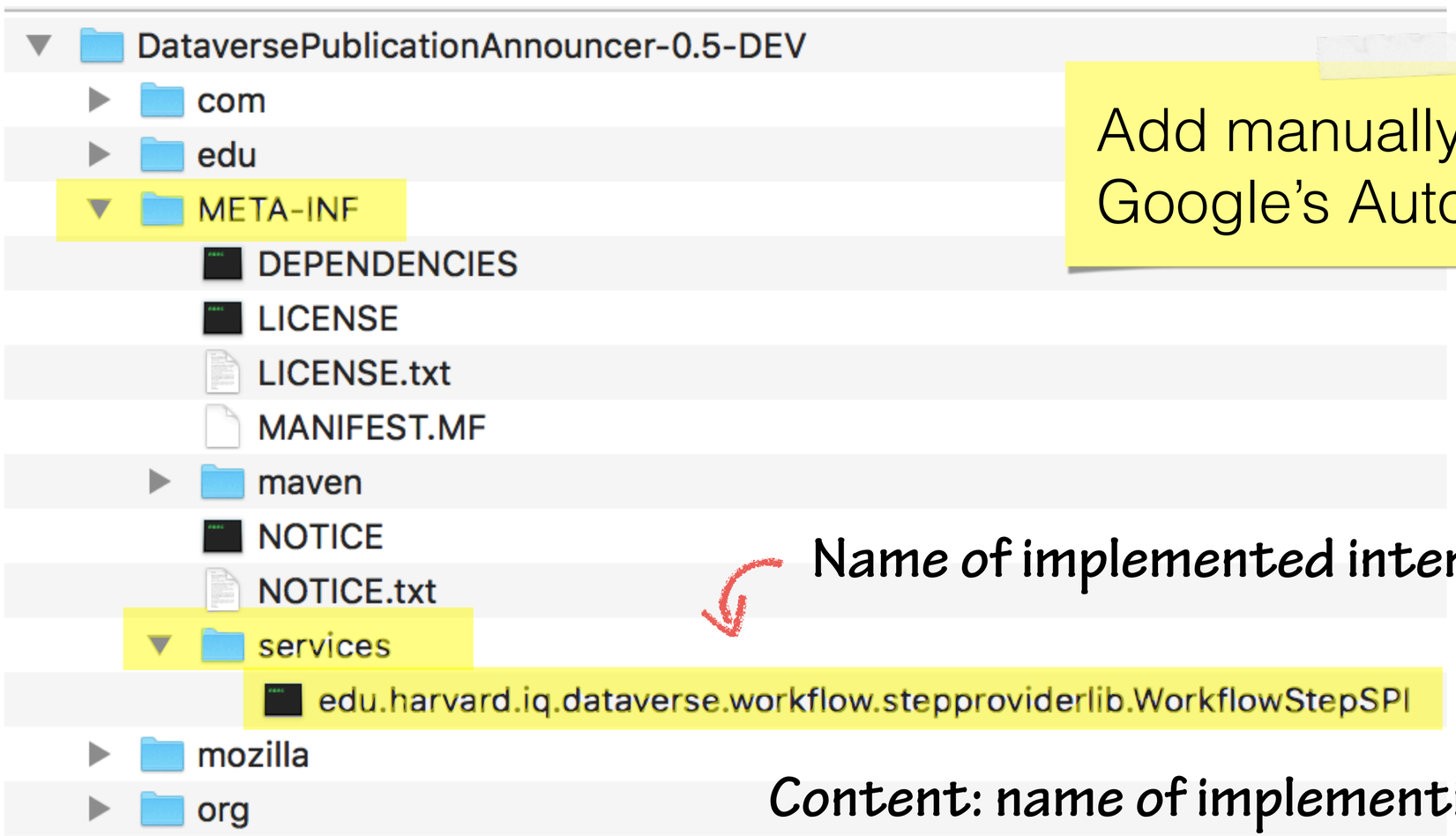


    - *Careful what you expose*

- Third parties implement providers in another .jar, and declare provided services

- Monolith loads plugins upon startup

Plugin .jar File Layout



Loading Plugins

1) Place the plugin .jar file in the lib folder inside the domain folder

2) Load like so:

```
public WorkflowServiceBean() {
    providers.put(":internal", new InternalWorkflowStepSP());

    logger.log(Level.INFO, "Searching for workflow step providers...");
    ServiceLoader<WorkflowStepSPI> loader = ServiceLoader.load(WorkflowStepSPI.class);
    try {
        for ( WorkflowStepSPI wss : loader ) {
            logger.log(Level.INFO, "Found WorkflowStepProvider: {0}", wss.getClass().getCanonicalName());
            providers.put( wss.getClass().getCanonicalName(), wss );
        }
        logger.log(Level.INFO, "Searching for Workflow Step Providers done.");
    } catch (NoClassDefFoundError ncdfe) {
        logger.log(Level.WARNING, "Class not found: " + ncdfe.getMessage(), ncdfe);
    } catch (ServiceConfigurationError serviceError) {
```

Loading Plugins

1) Place the plugin .jar file in the lib folder inside the domain folder

2) Load like so:

```
public WorkflowServiceBean() {
    providers.put(":internal", new InternalWorkflowStepSP());

    logger.log(Level.INFO, "Searching for workflow step providers...");
    ServiceLoader<WorkflowStepSPI> loader = ServiceLoader.load(WorkflowStepSPI.class);
    try {
        for ( WorkflowStepSPI wss : loader ) {
            logger.log(Level.INFO, "Found WorkflowStepProvider: {0}", wss.getClass().getCanonicalName());
            providers.put( wss.getClass().getCanonicalName(), wss );
        }
        logger.log(Level.INFO, "Searching for Workflow Step Providers done.");
    } catch (NoClassDefFoundError ncdfe) {
        logger.log(Level.WARNING, "Class not found: " + ncdfe.getMessage(), ncdfe);
    } catch (ServiceConfigurationError serviceError) {
```

Loading Plugins

1) Place the plugin .jar file in the lib folder inside the domain folder

2) Load like so:

```
public WorkflowServiceBean() {
    providers.put(":internal", new InternalWorkflowStepSP());

    logger.log(Level.INFO, "Searching for workflow step providers...");
    ServiceLoader<WorkflowStepSPI> loader = ServiceLoader.load(WorkflowStepSPI.class);
    try {
        for ( WorkflowStepSPI wss : loader ) {
            logger.log(Level.INFO, "Found WorkflowStepProvider: {0}", wss.getClass().getCanonicalName());
            providers.put( wss.getClass().getCanonicalName(), wss );
        }
        logger.log(Level.INFO, "Searching for Workflow Step Providers done.");
    } catch (NoClassDefFoundError ncdfe) {
        logger.log(Level.WARNING, "Class not found: " + ncdfe.getMessage(), ncdfe);
    } catch (ServiceConfigurationError serviceError) {
```

Loading Plugins

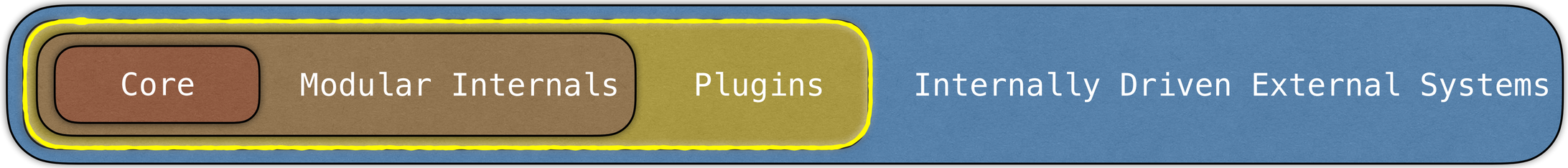
1) Place the plugin .jar file in the lib folder inside the domain folder

2) Load like so:

```
public WorkflowServiceBean() {
    providers.put(":internal", new InternalWorkflowStepSP());

    logger.log(Level.INFO, "Searching for workflow step providers...");
    ServiceLoader<WorkflowStepSPI> loader = ServiceLoader.load(WorkflowStepSPI.class);
    try {
        for ( WorkflowStepSPI wss : loader ) {
            logger.log(Level.INFO, "Found WorkflowStepProvider: {0}", wss.getClass().getCanonicalName());
            providers.put( wss.getClass().getCanonicalName(), wss );
        }
        logger.log(Level.INFO, "Searching for Workflow Step Providers done.");
    } catch (NoClassDefFoundError ncdfe) {
        logger.log(Level.WARNING, "Class not found: " + ncdfe.getMessage(), ncdfe);
    } catch (ServiceConfigurationError serviceError) {
```

Caution:
Classpath Issues Ahead

- Java SE class loading is standardized and intuitive
- Java EE class loading - not so much
  - Plugin dependencies may need to be added with the plugin, even if the application already depends on them (container dependent!)



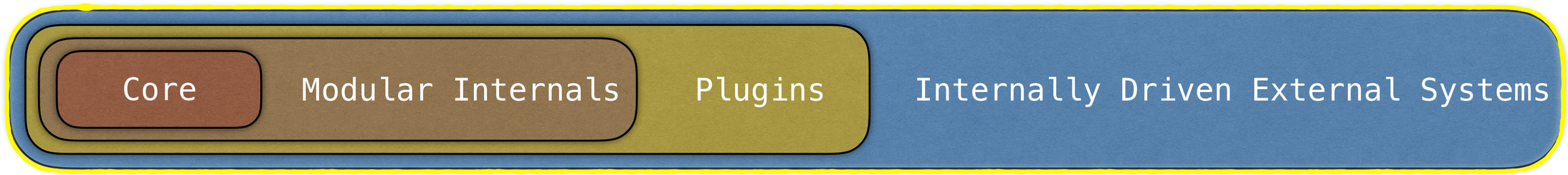
Plugins

Codebase

- ✔ Size can go down as functionality is moved out
- ✔ Extension via jar files

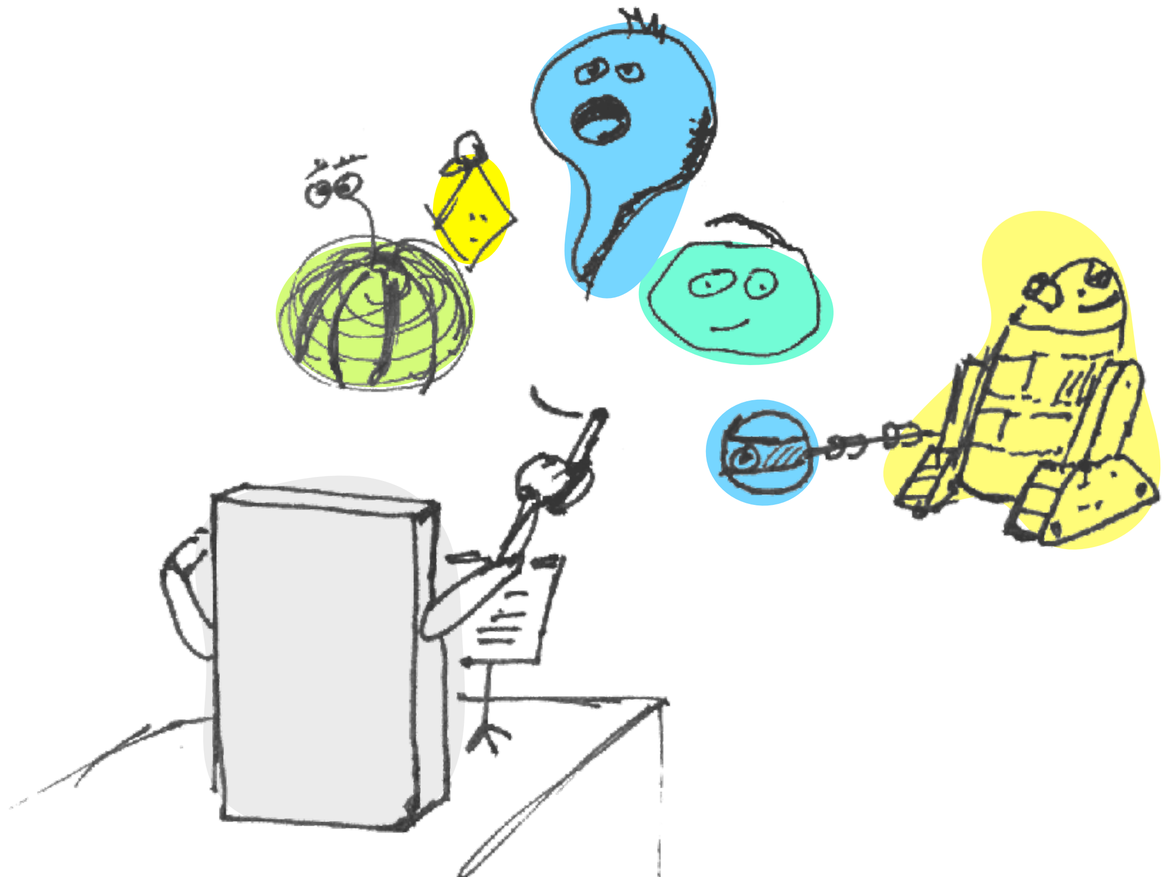
Headache

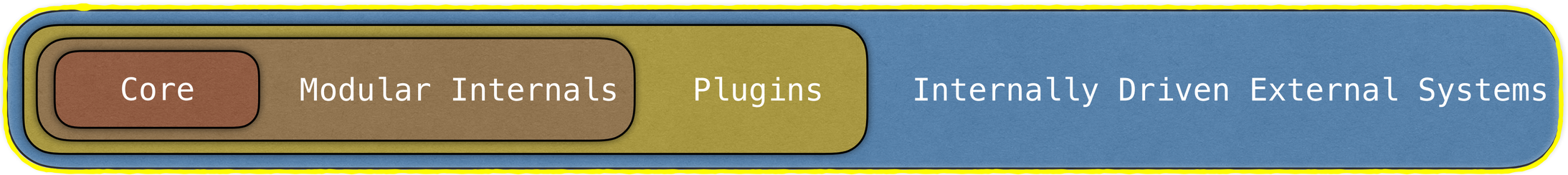
- ✔ Empowers the community to add functionality on its own
- ✘ A new public interface to maintain
- ✘ Security and stability may be an issue



Internally Driven External Systems (IDES)

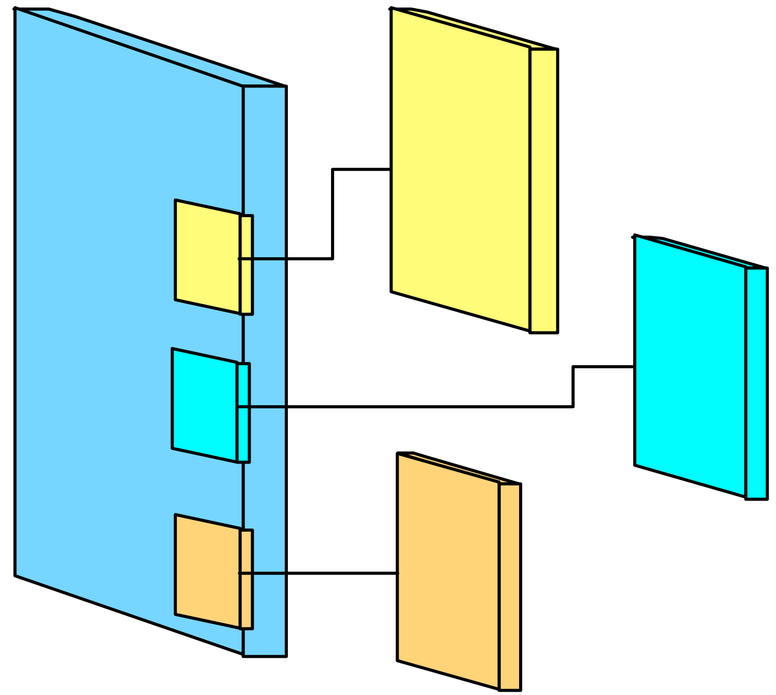
- Core application sends requests to external systems (and waits)
- External systems reports back
- ***Smaller monolith that orchestrates other systems***

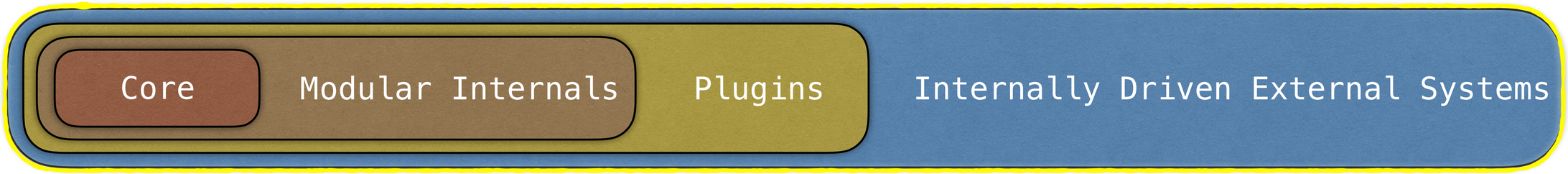




Remote-Aware IDES

- Monolith is aware of the external system, actually holding a specific piece of code for it
  - Essentially, a driver
- Not super modular, just broken into two.

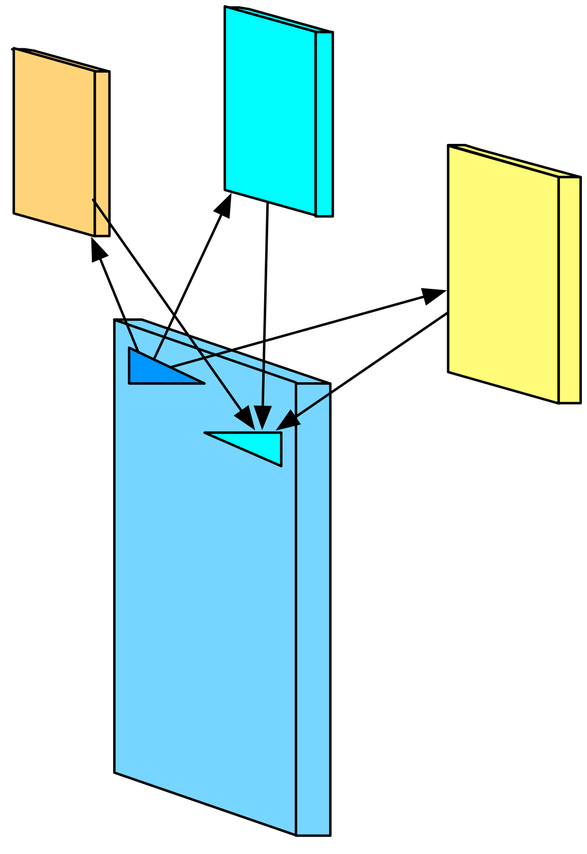


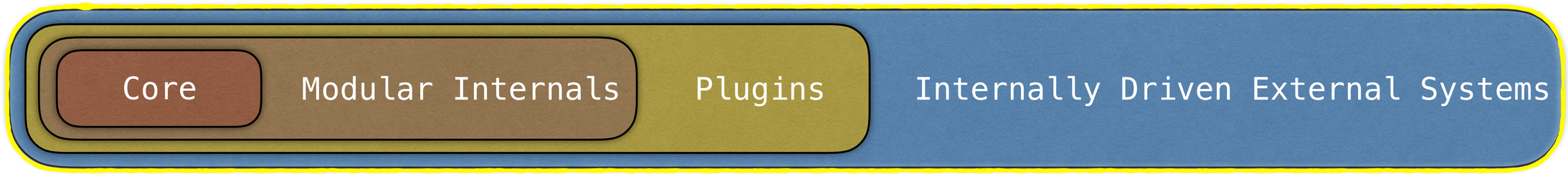


Standard Interface IDES

- Monolith provides a configurable way of sending messages and receiving responses

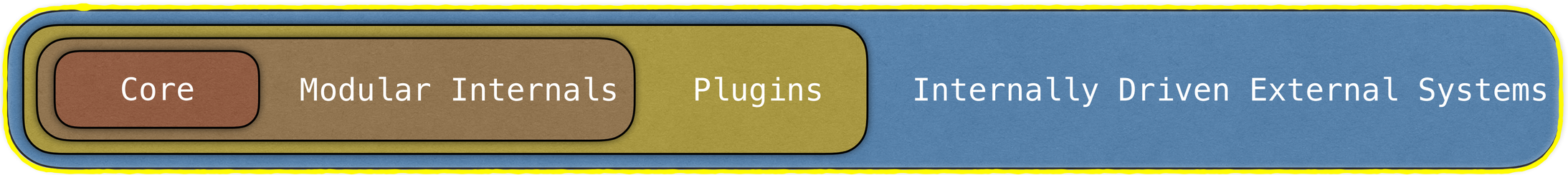
- Remote systems implement standard for listening and responding





Standard Interface IDES Example: **REST Client**

```
{
  "provider":":internal",
  "stepType":"http/sr",
  "parameters": {
      "url":"http://localhost:5050/dump/${invocationId}",
      "method":"POST",
      "contentType":"text/plain",
      "body":"START RELEASE ${dataset.id} as ${dataset.displayName}",
      "expectedResponse":"OK.*",
      "rollbackUrl":"http://localhost:5050/dump/${invocationId}",
      "rollbackMethod":"DELETE ${dataset.id}"
  }
}
```

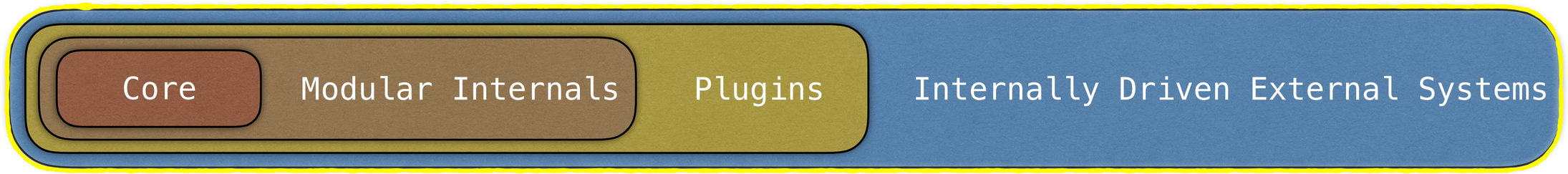


Standard Interface IDES Example: **REST Client**

```
{
  "provider":":internal",
  "stepType":"http/sr",
  "parameters": {
      "url":"http://localhost:5050/dump/${invocationId}",
      "method":"POST",
      "contentType":"text/plain",
      "body":"START RELEASE ${dataset.id} as ${dataset.displayName}",
      "expectedResponse":"OK.*",
      "rollbackUrl":"http://localhost:5050/dump/${invocationId}",
      "rollbackMethod":"DELETE ${dataset.id}"
  }
}
```

Available variables are:

- `invocationId`
- `dataset.id`
- `dataset.identifier`
- `dataset.globalId`
- `dataset.displayName`
- `dataset.citation`
- `minorVersion`
- `majorVersion`
- `releaseStatus`



Standard Interface IDES

Codebase

- ✔ Size can go down as functionality is moved out
- ✔ Extension via configuration

Headache

- ✔ Empowers the community to add functionality on its own
- ✔ Core team not aware of remote systems at all

Case Study: ***Publication Workflows***

or

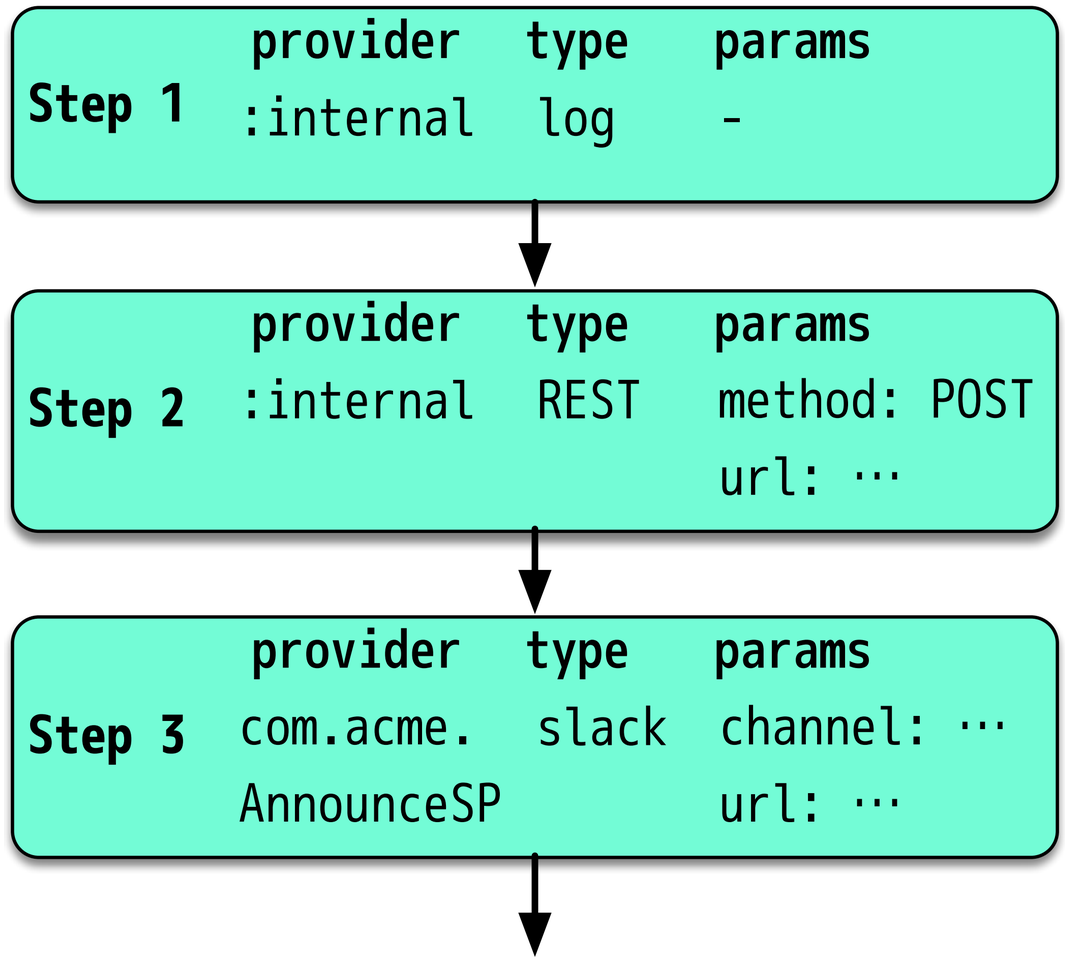
Mix everything and add users

Publication Workflows

- Releasing a dataset may require actions by external, 3rd party systems
- Different domains use different systems
- Actions may fail
- Actions may take an arbitrarily amount of time
  - e.g. human approval or transferring terabytes

Publication Workflows: Design

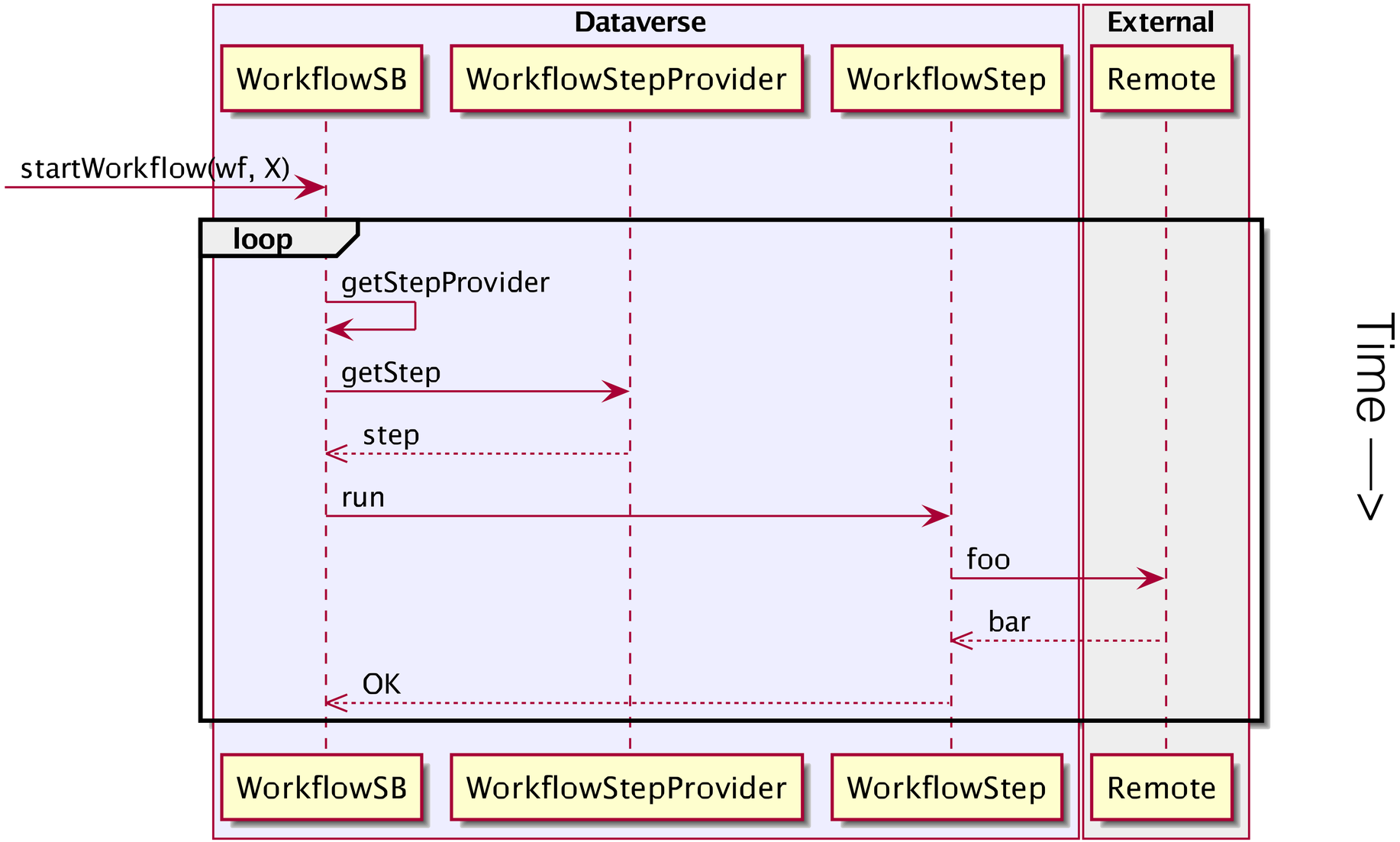
- A Workflow is a series of ***WorkflowSteps*** Stored in DB
- WorkflowSteps define methods to ***run()***, ***resume()***, and ***rollback()***
- run() can return a ***"pending" result***, which causes the workflow runtime to ***store the workflow state*** in the database
- Workflow states are ***restored*** when remote system responds
- Failed step causes the runtime to ***rollback all previous successful*** steps



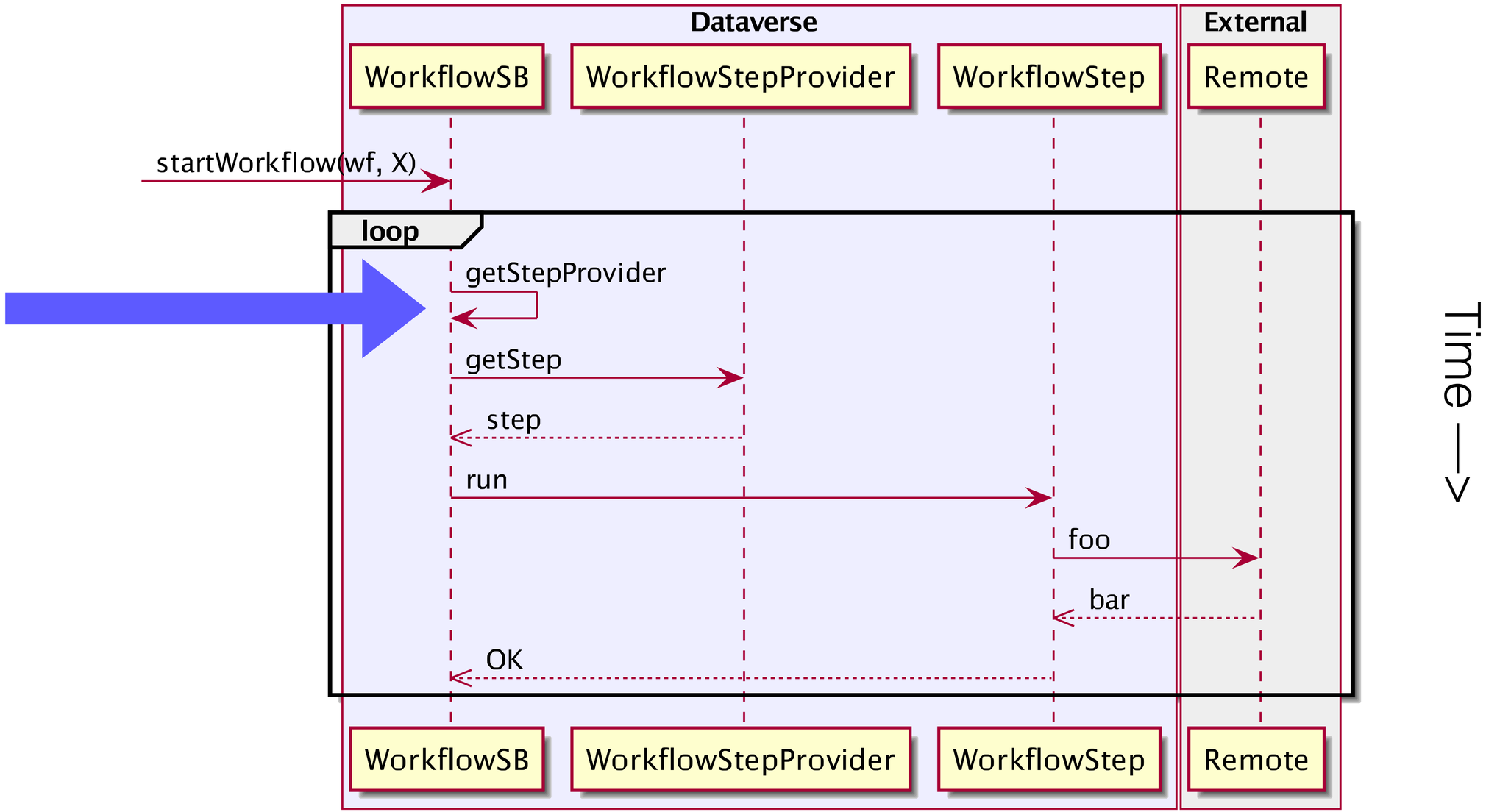
Sample WorkflowStepSPI Implementation (Plugin)

```java
@AutoService(WorkflowStepSPI.class)
public class AnnouncerWorkflowStepSPI implements WorkflowStepSPI {

    @Override
    public WorkflowStep getStep(String stepType, Map<String, String> stepParameters) {
        switch (stepType) {
            case "slack":
                return new SlackAnnouncer(stepParameters.get("username"),
                                          stepParameters.get("channel"),
                                          stepParameters.get("url"));
            case "say":
                return new SayAnnouncer();

            default:
                throw new RuntimeException("Unknown step type '" + stepType + "'");

        }
    }

}
```

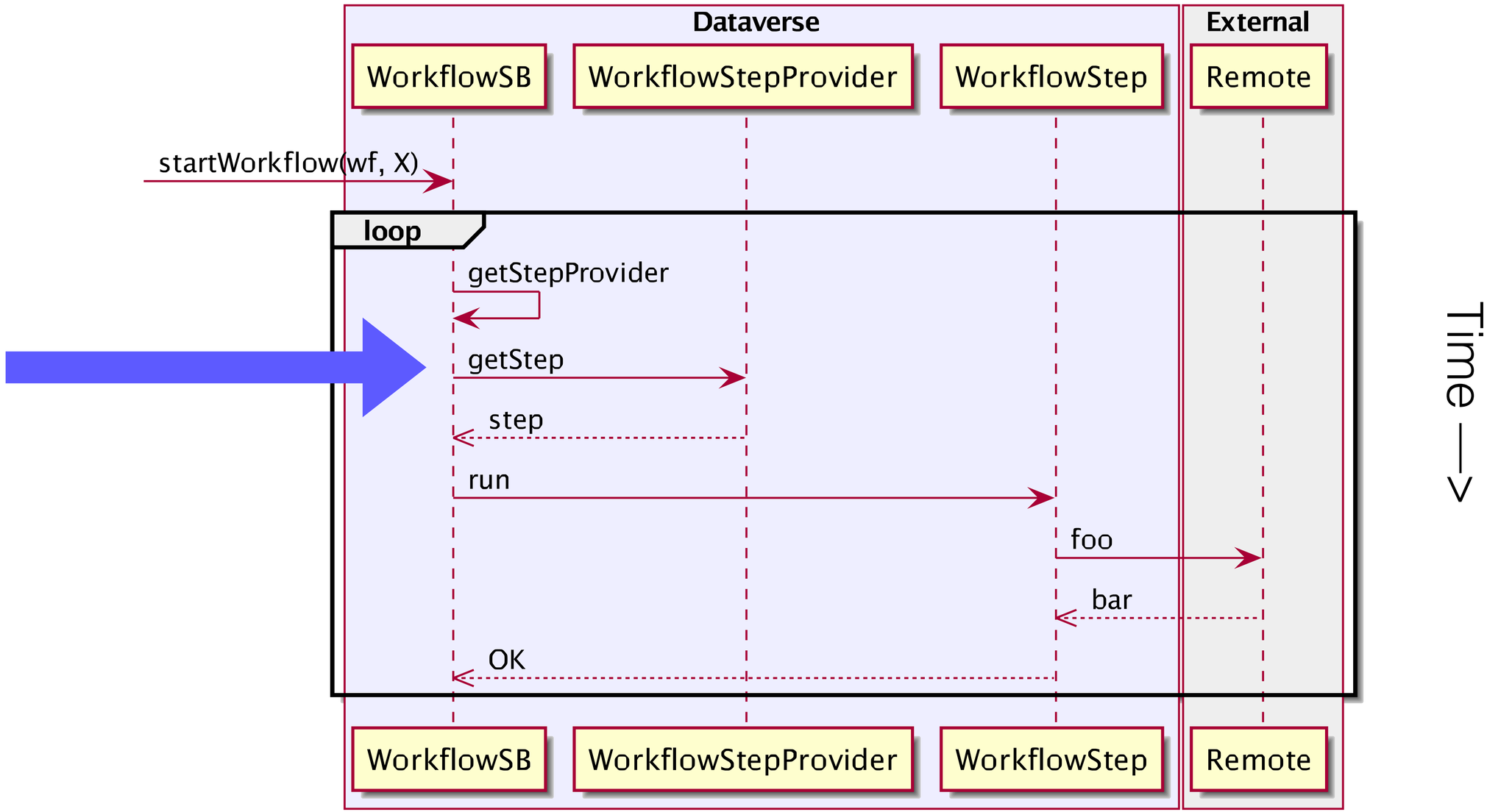
Workflow Sequence



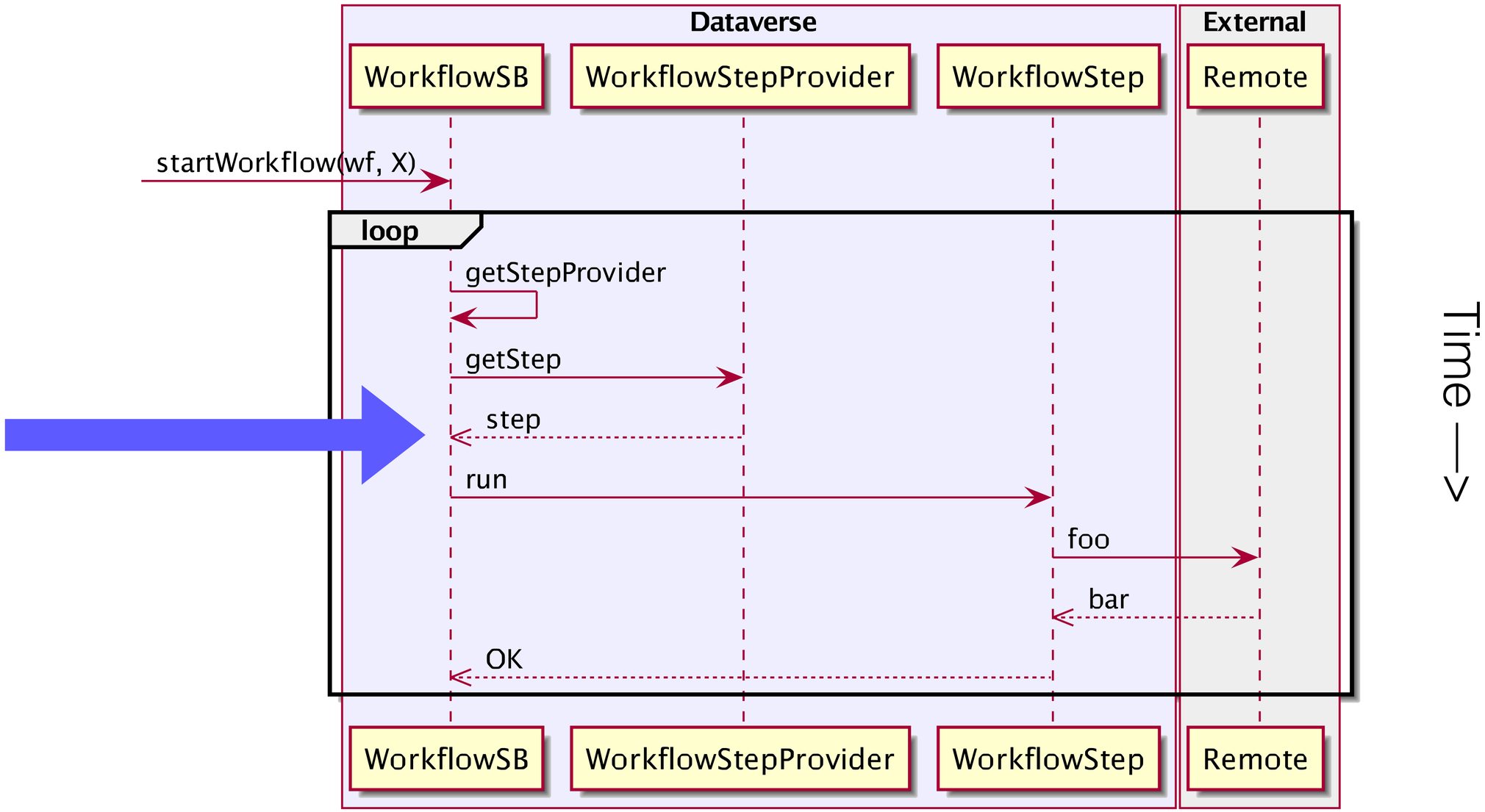
Workflow Sequence



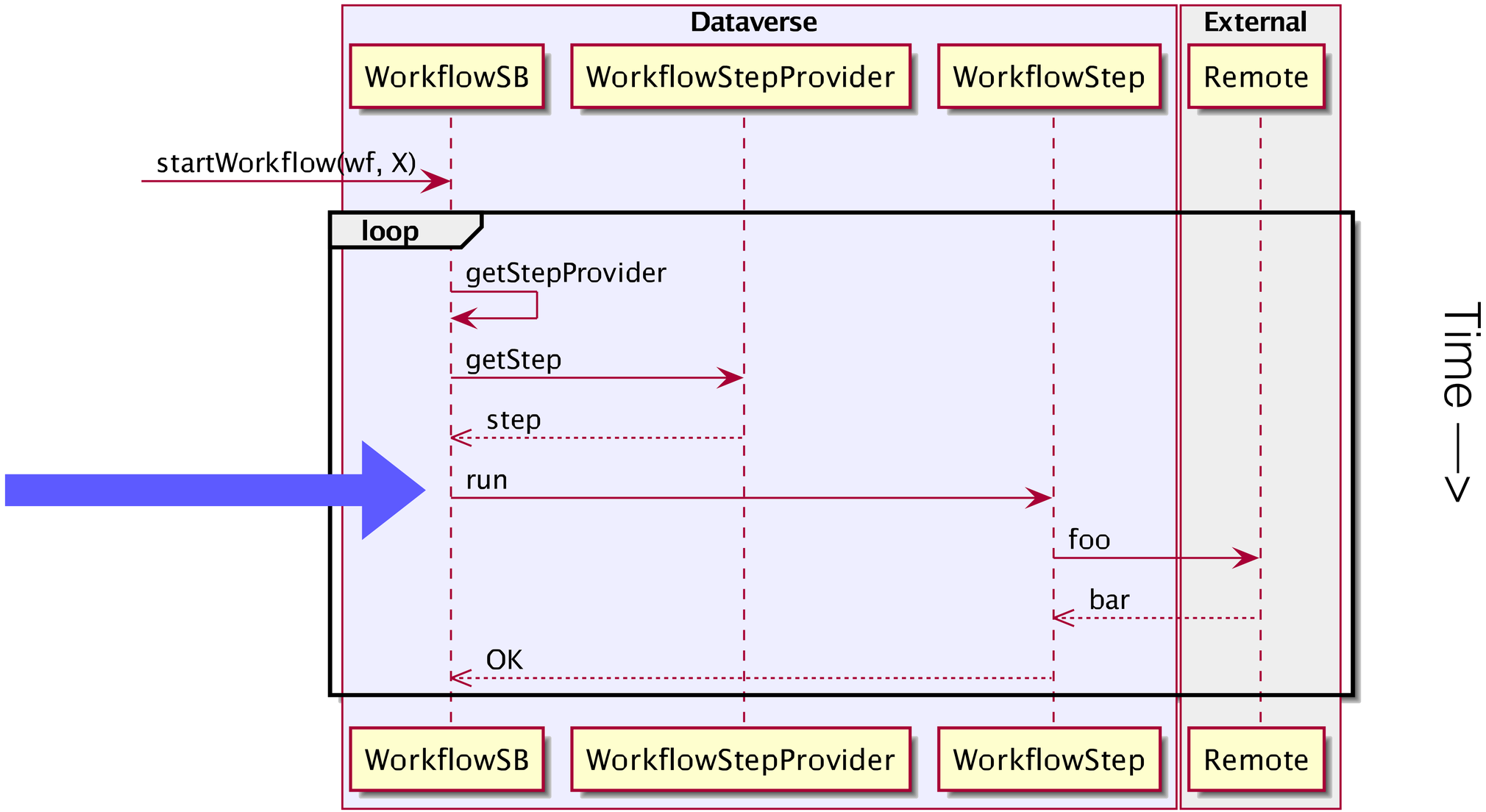
Workflow Sequence



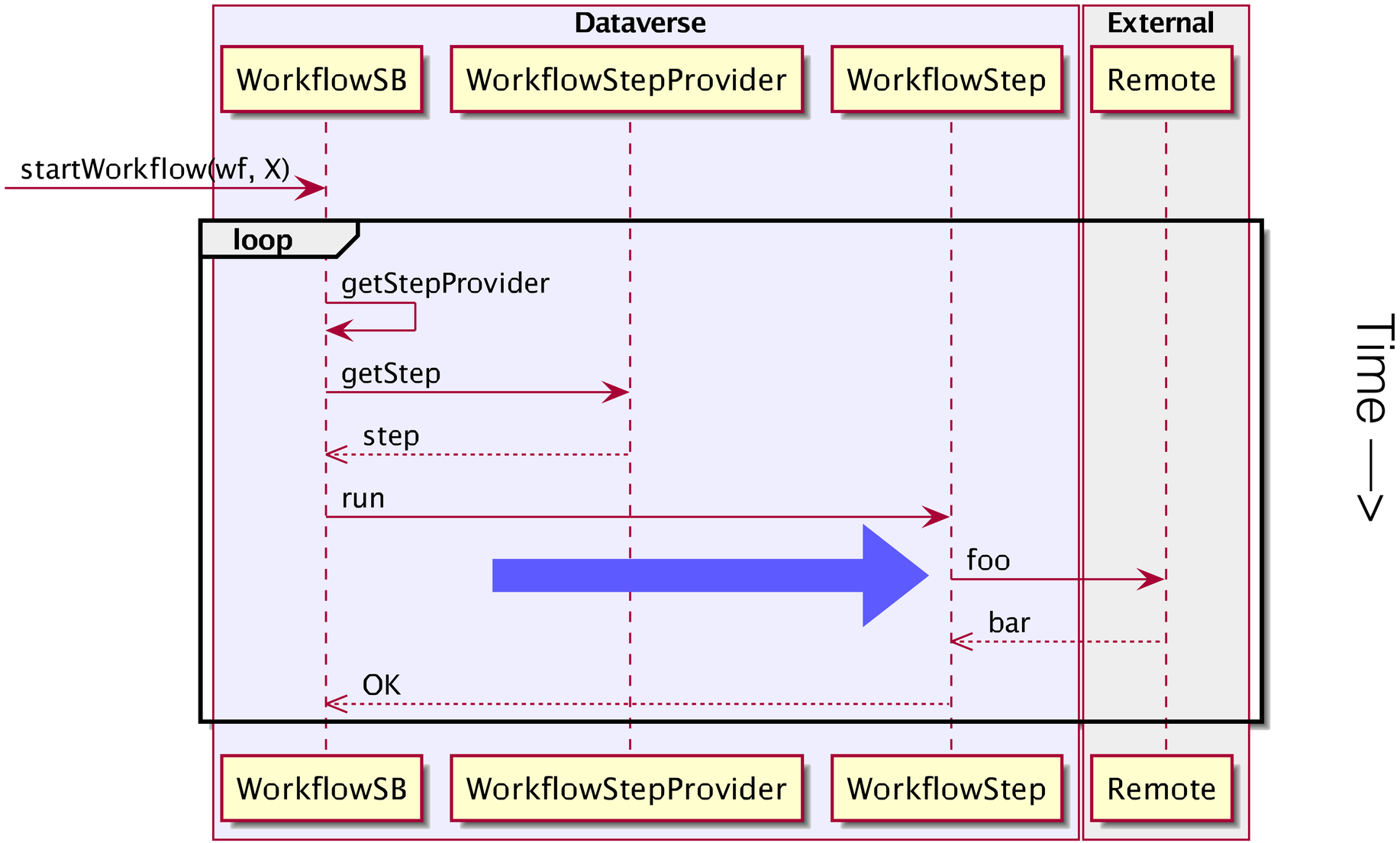
Workflow Sequence



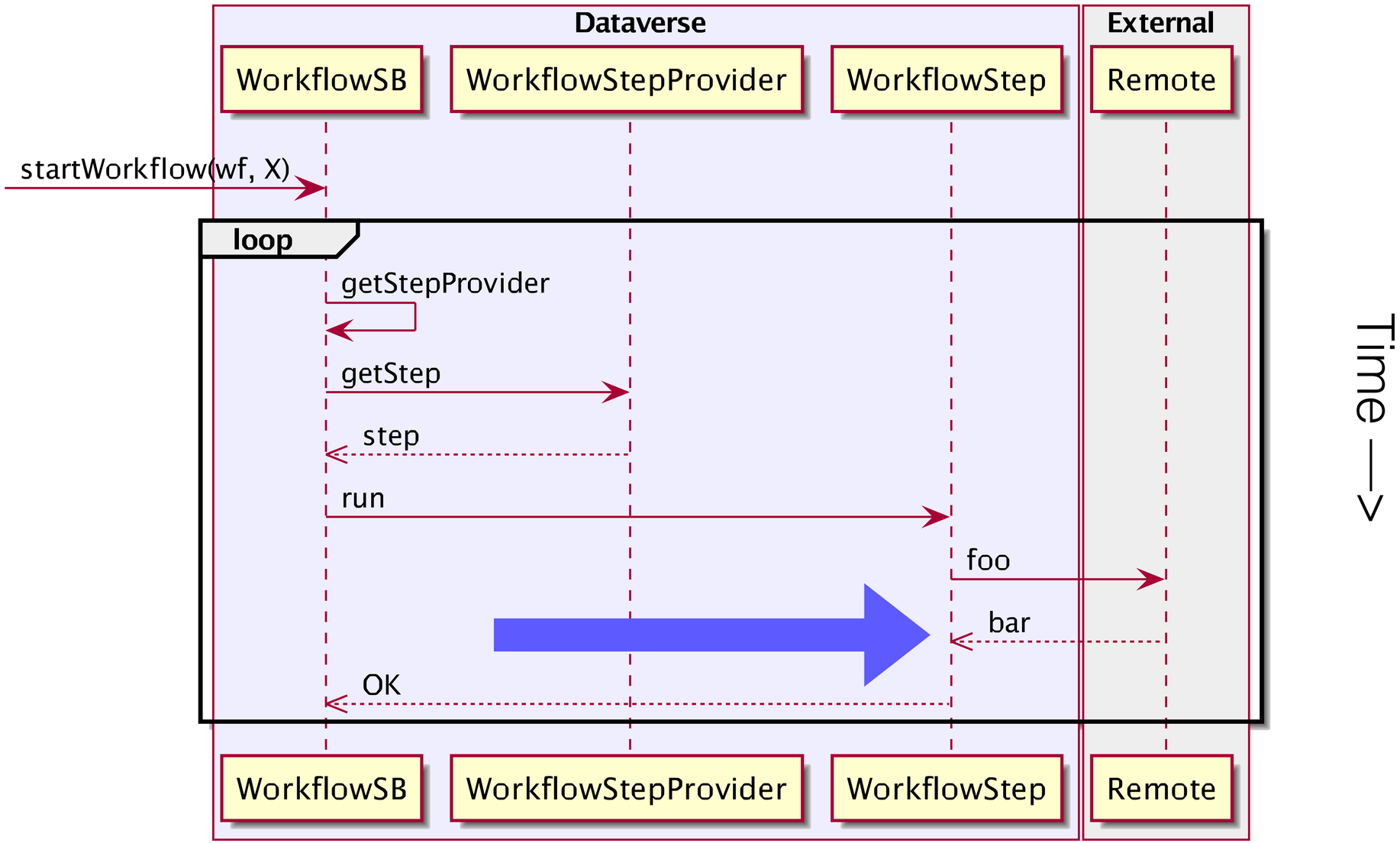
Workflow Sequence



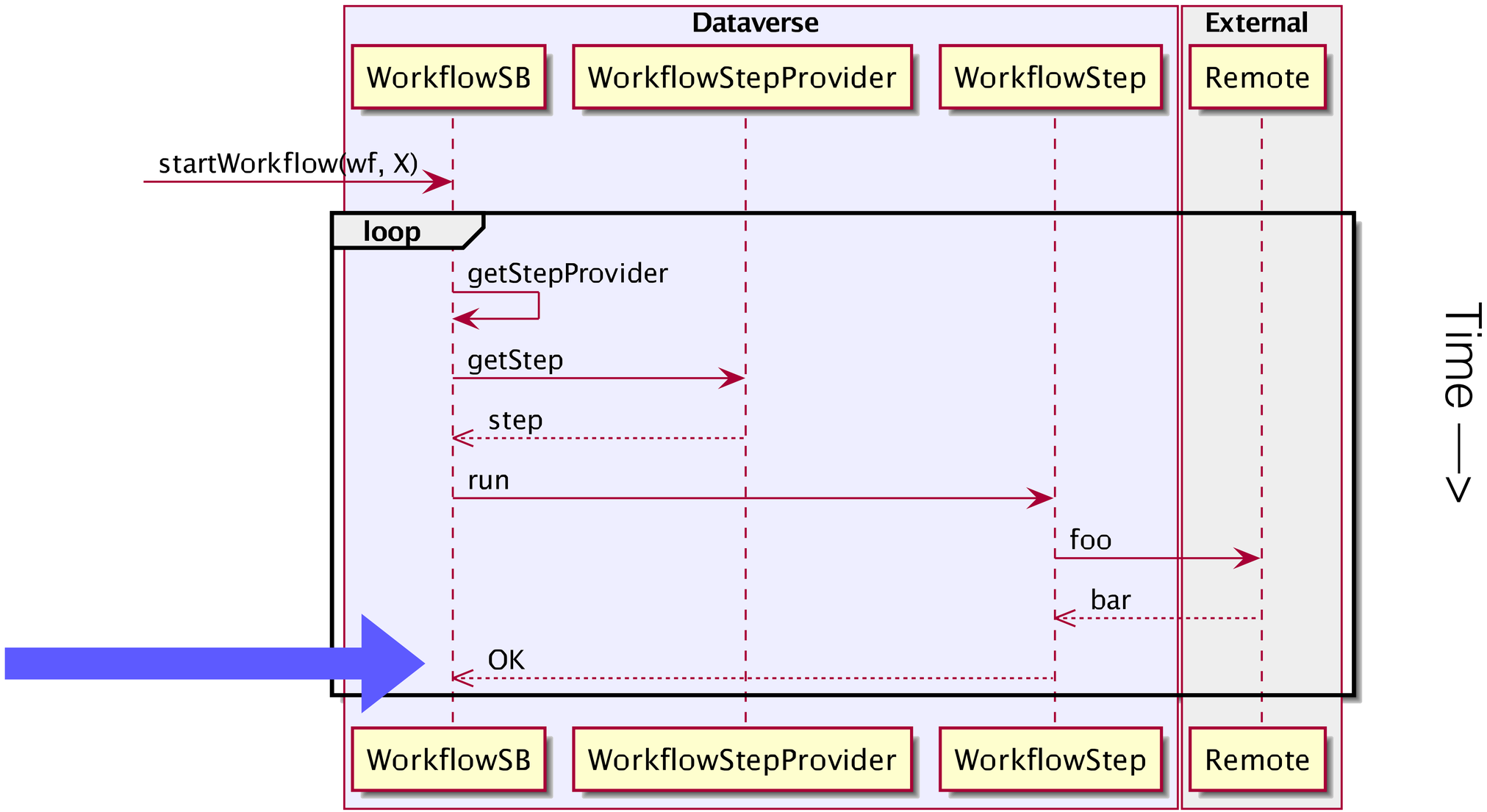
Workflow Sequence



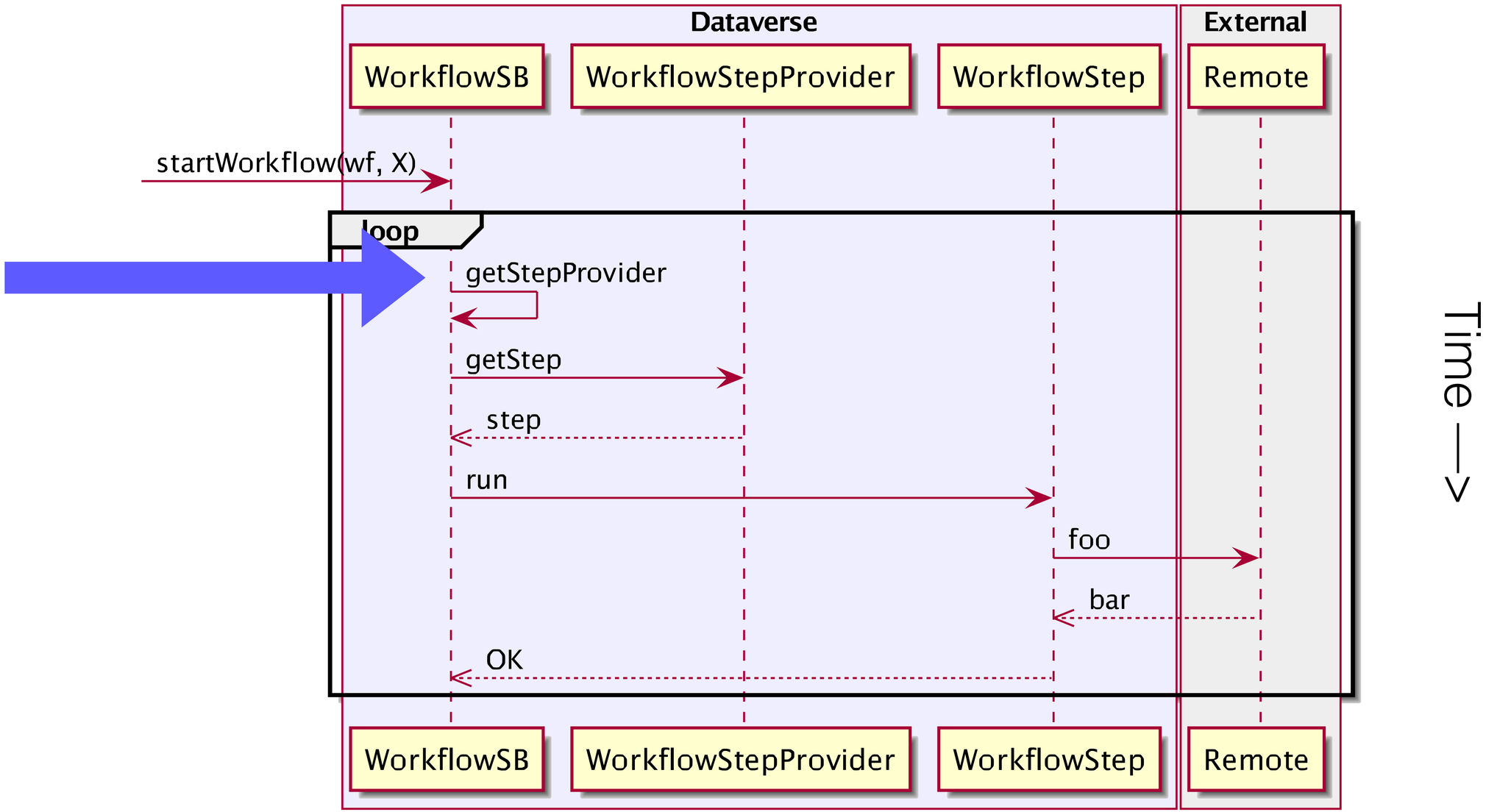
Workflow Sequence



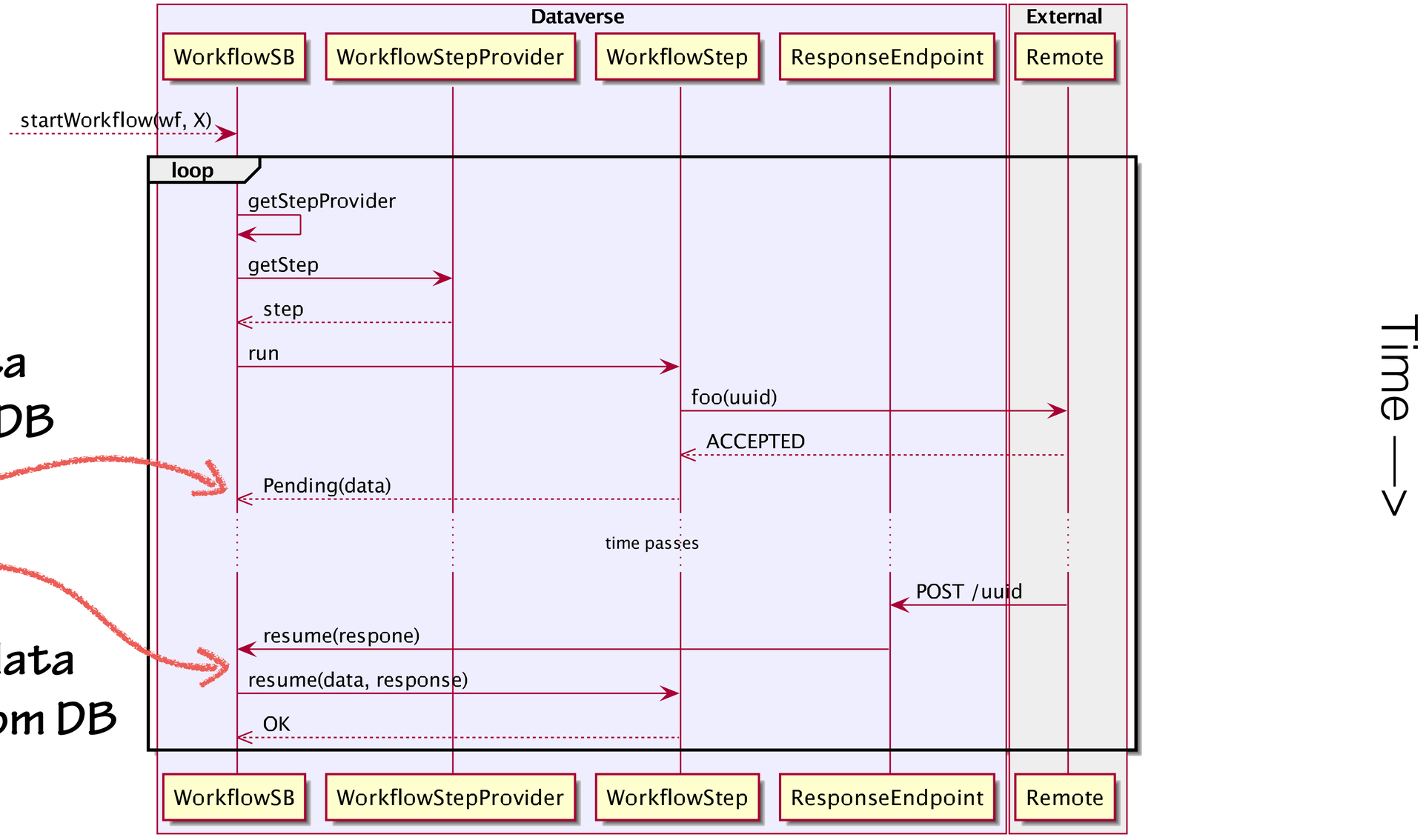
Workflow Sequence



Workflow Sequence



Workflow Sequence

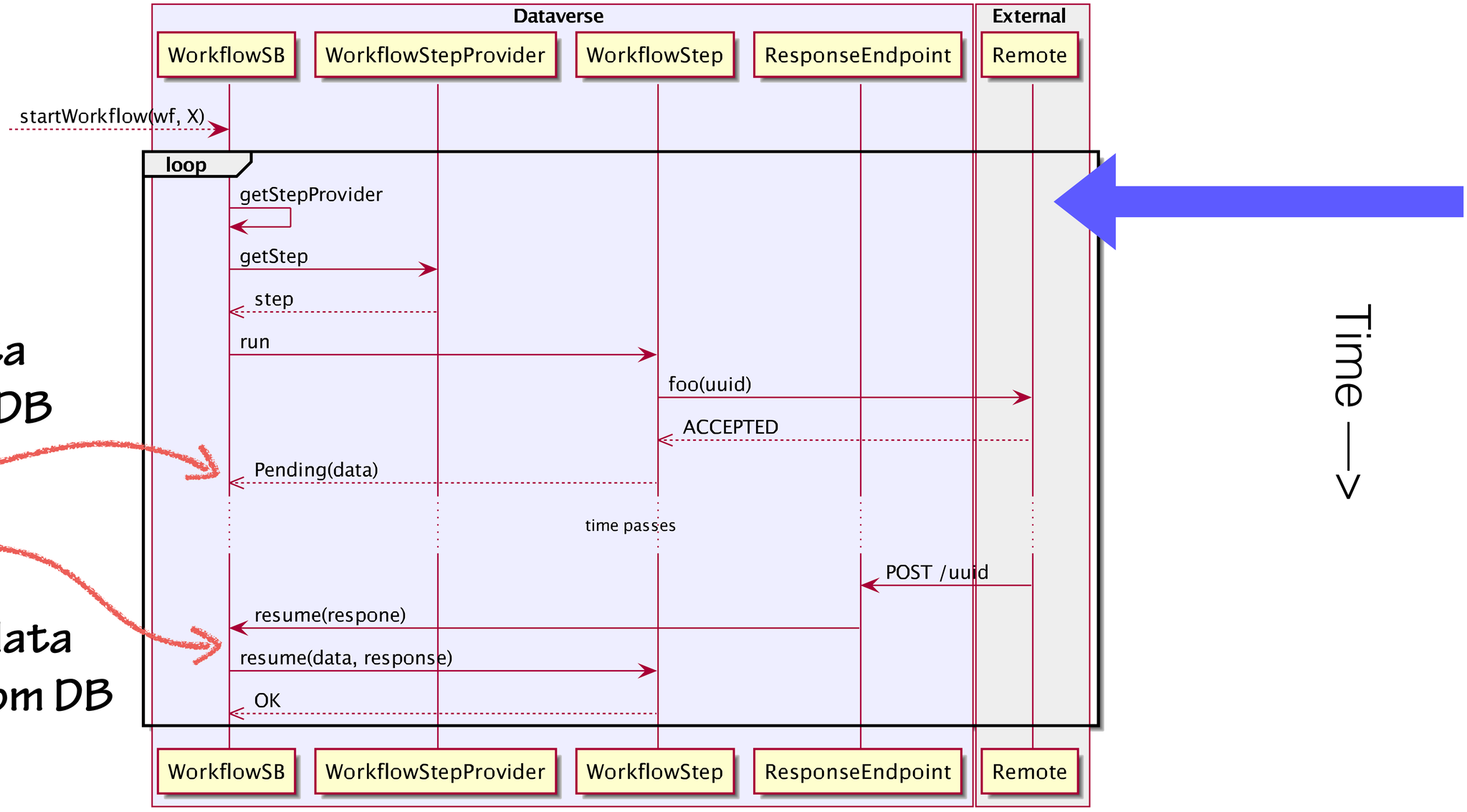


Store step data and context in DB

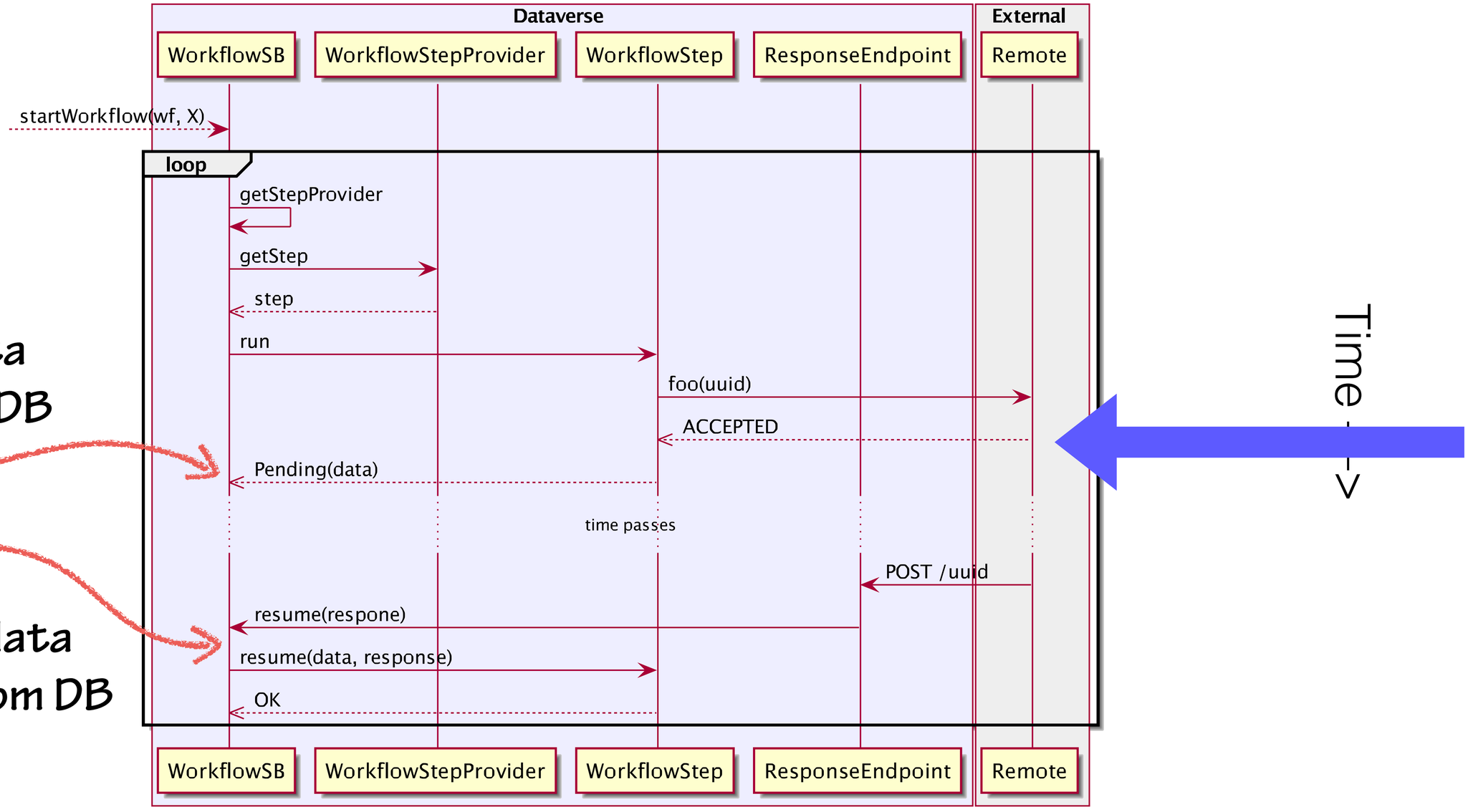
Restore step data and context from DB

Time —>

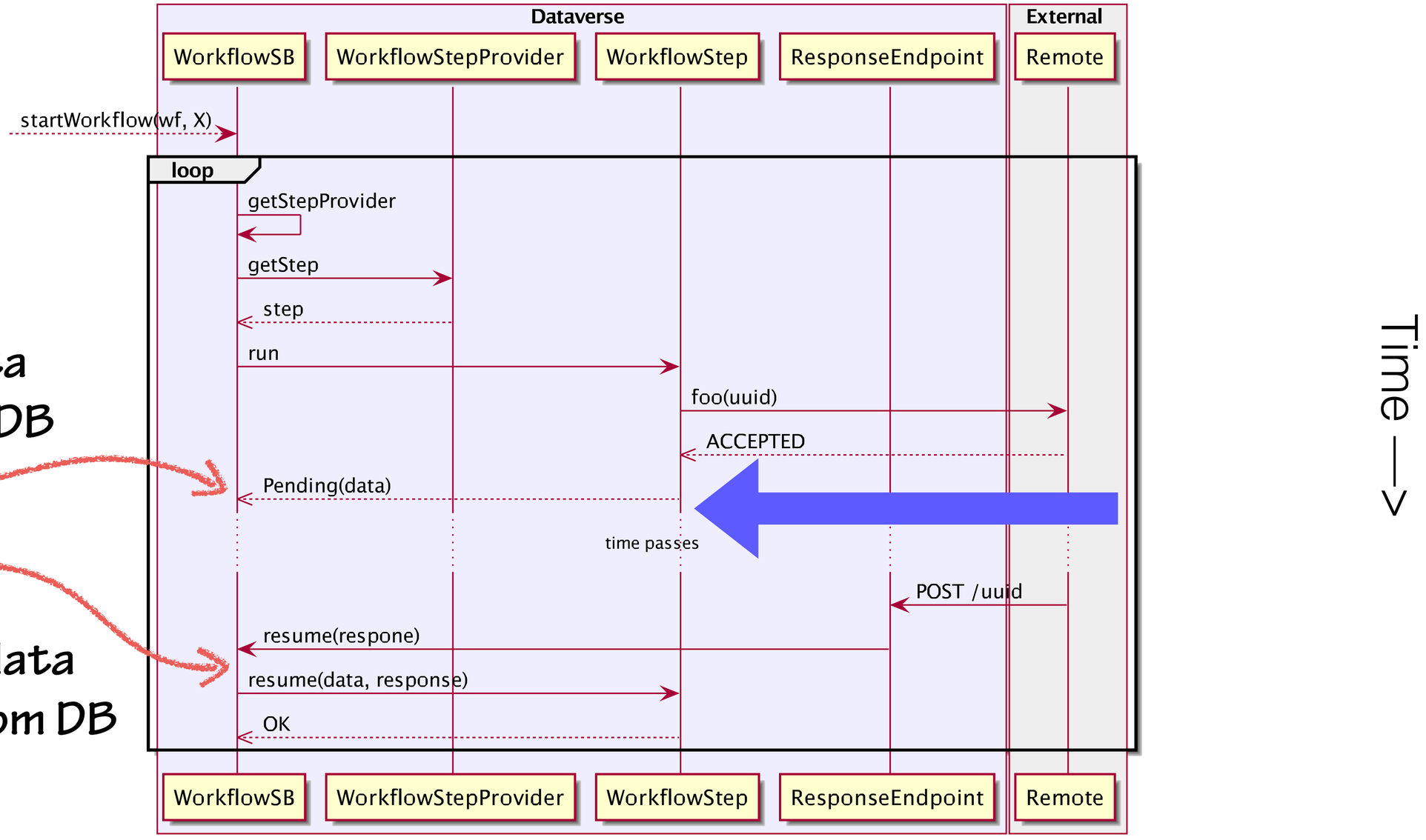
Workflow Sequence



Workflow Sequence



Workflow Sequence

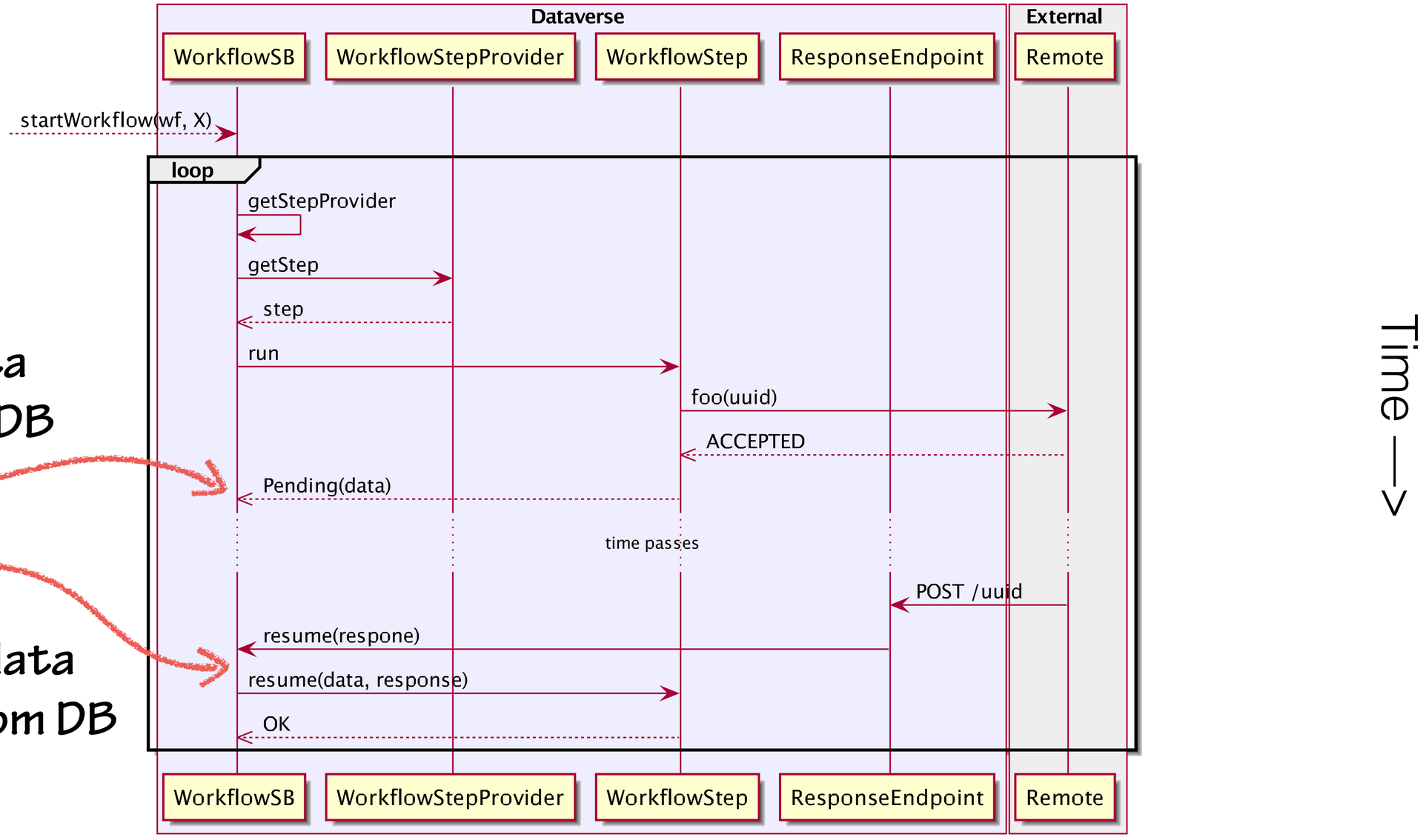


Store step data and context in DB

Restore step data and context from DB

Time —>

Workflow Sequence

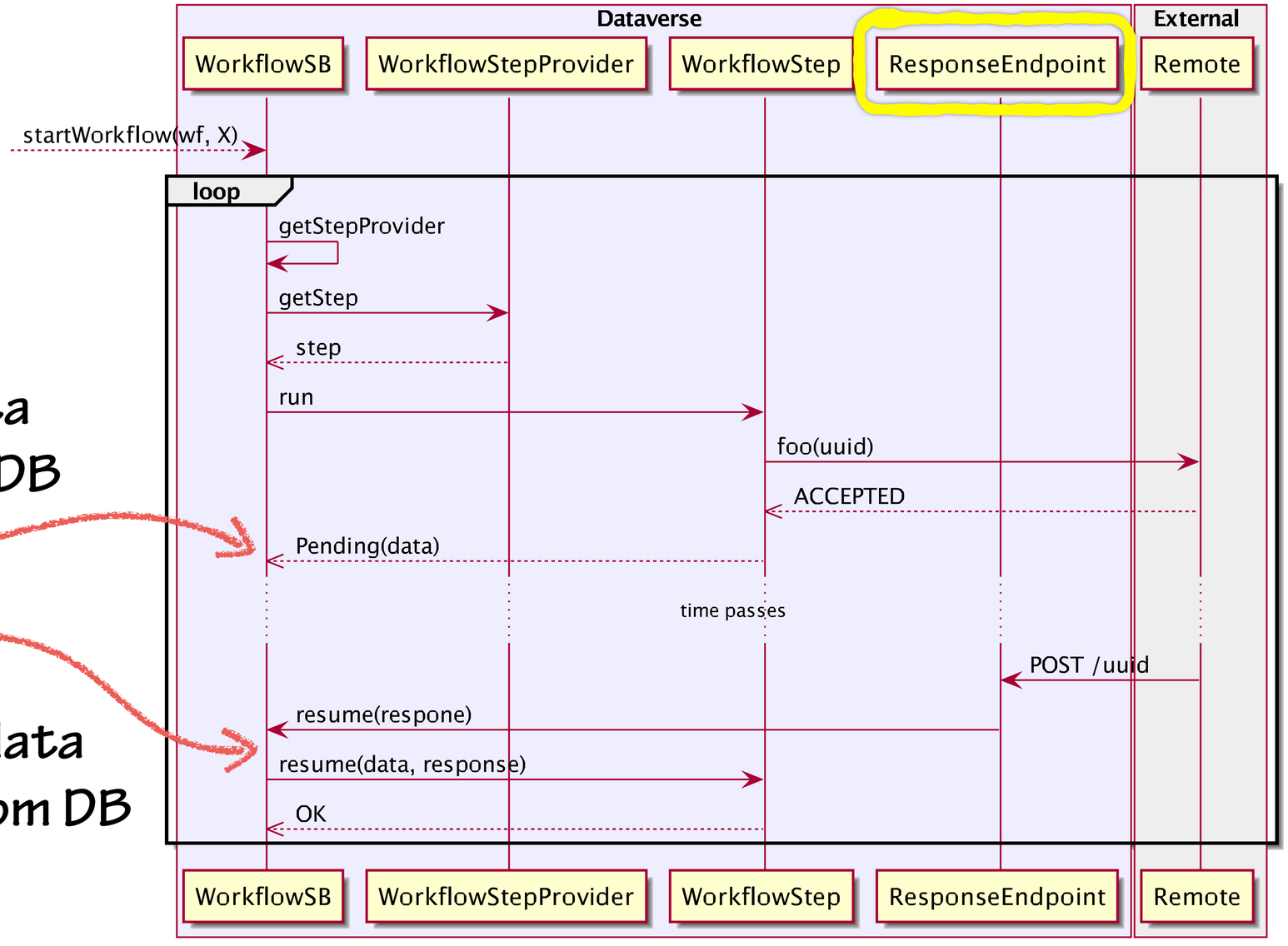


Store step data and context in DB

Restore step data and context from DB

Time —>

Workflow Sequence

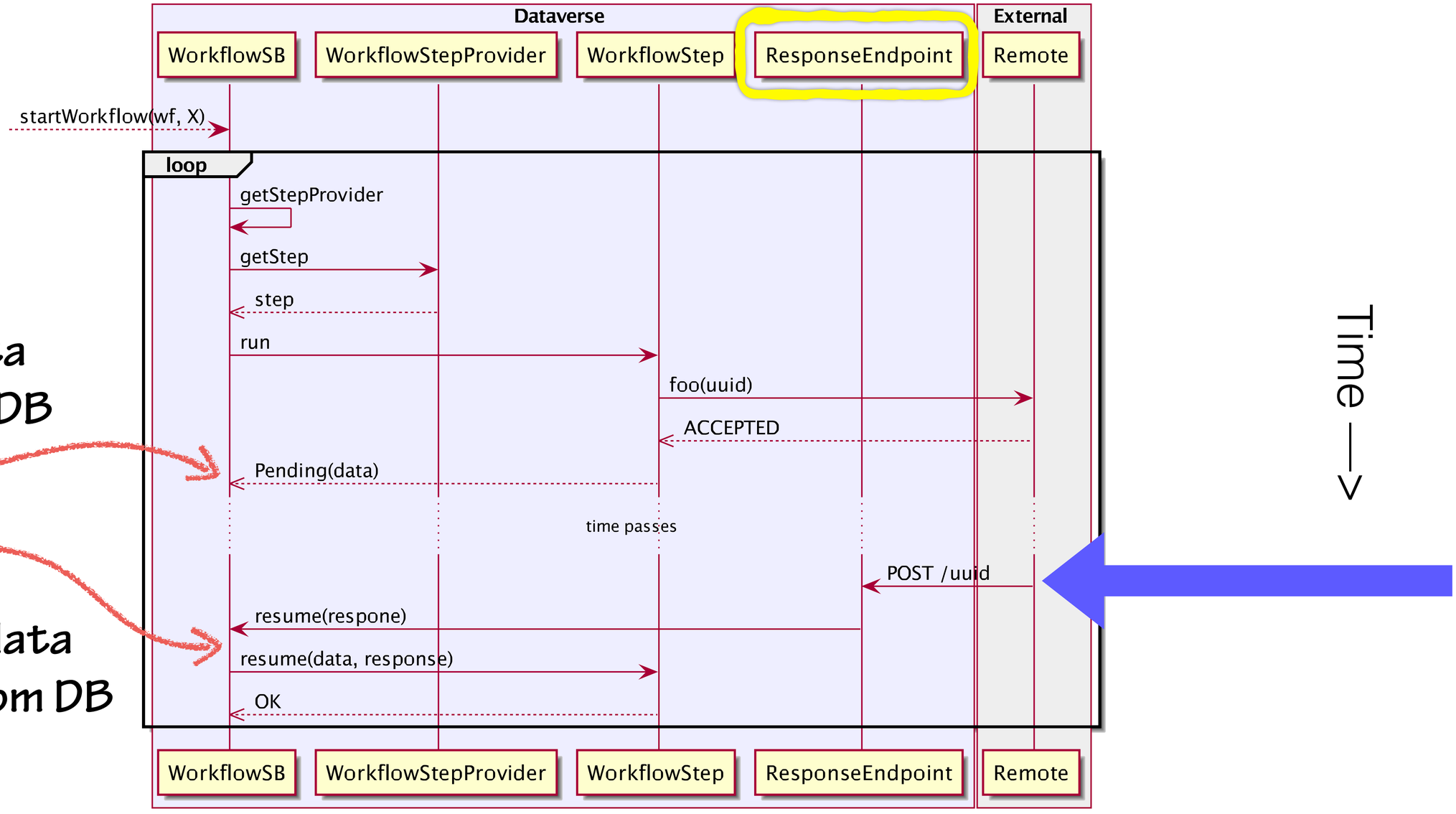


Store step data and context in DB

Restore step data and context from DB

Time —>

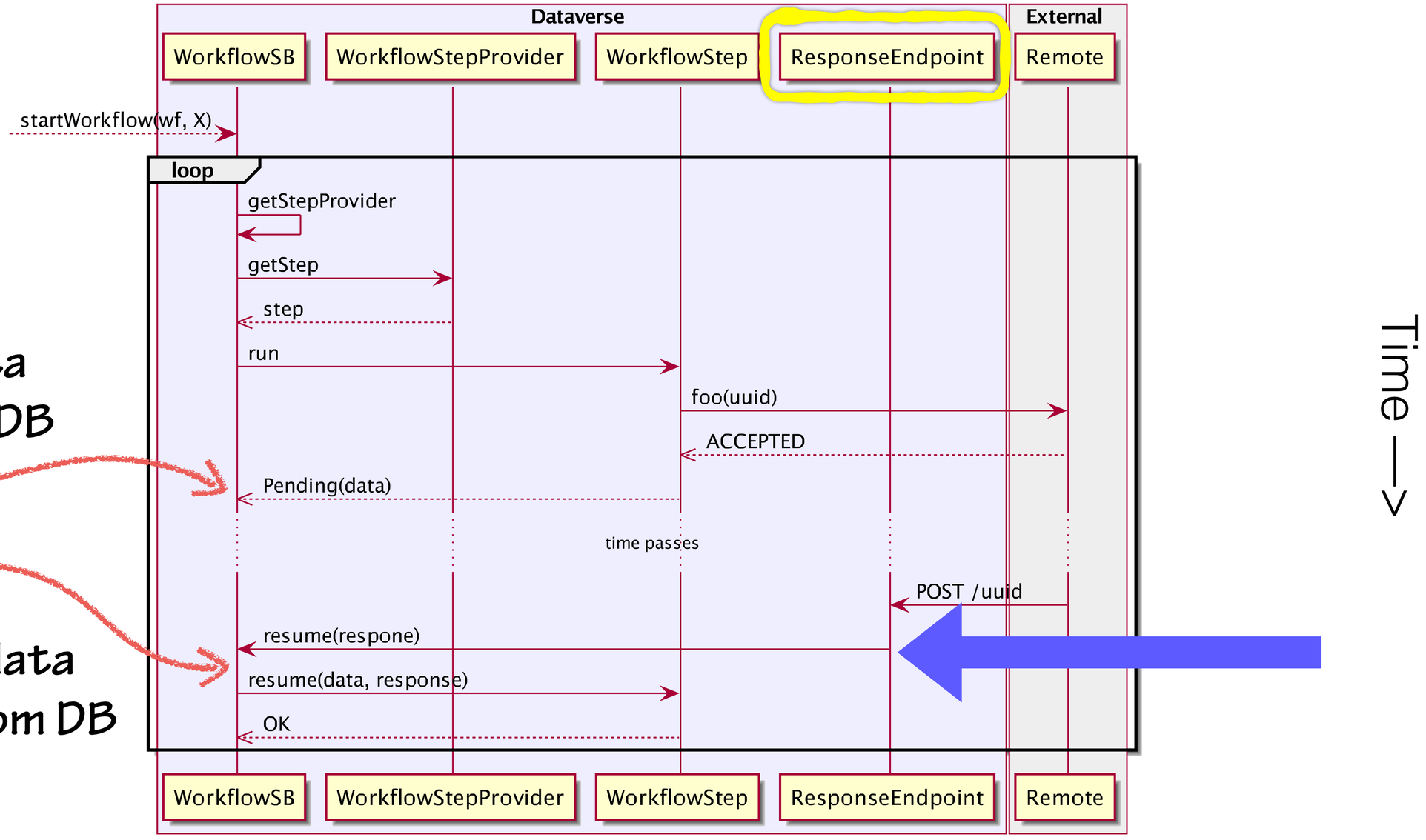
Workflow Sequence



Store step data and context in DB

Restore step data and context from DB

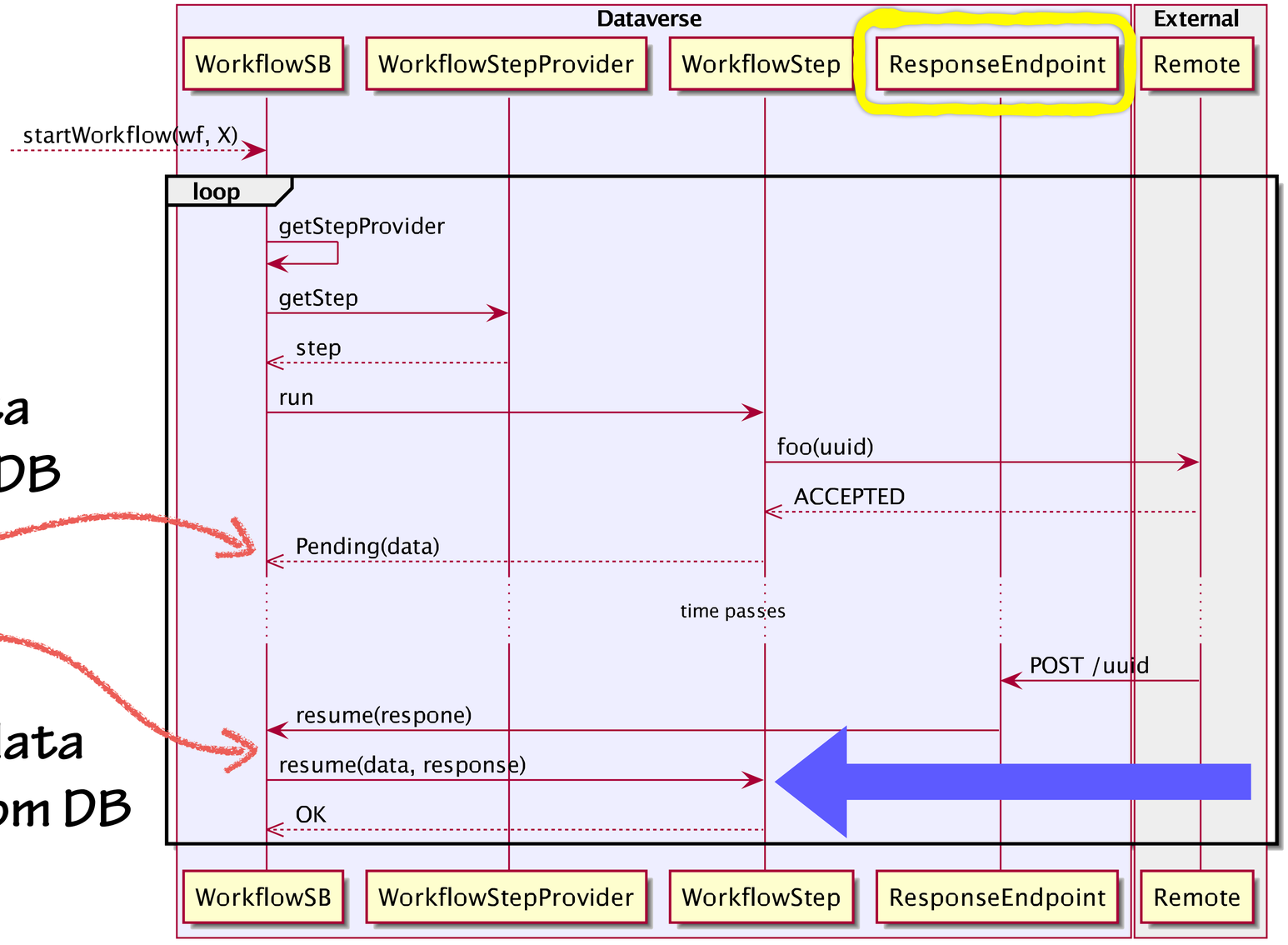
Workflow Sequence



Store step data and context in DB

Restore step data and context from DB

Workflow Sequence

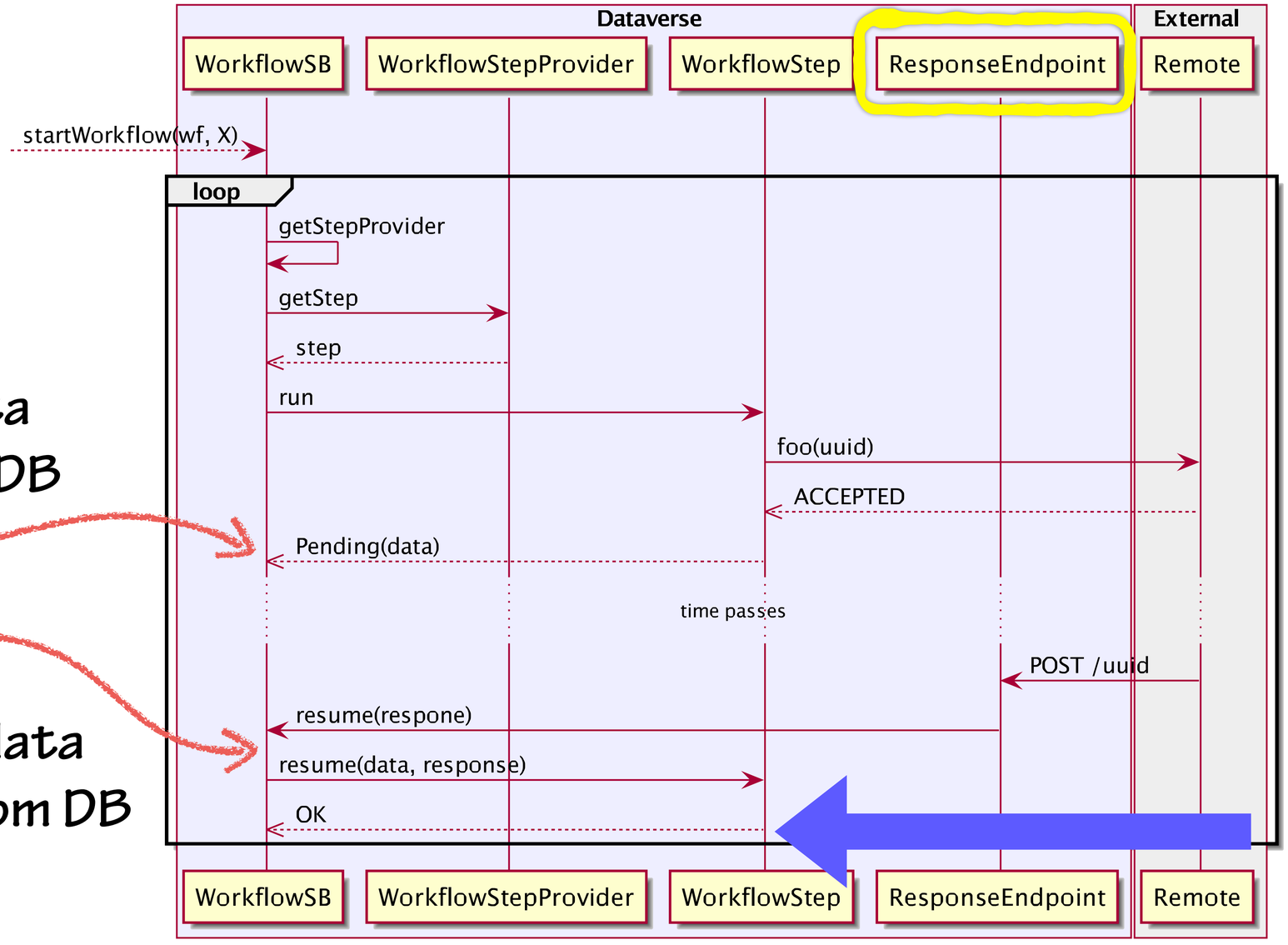


Store step data and context in DB

Restore step data and context from DB

Time —>

Workflow Sequence

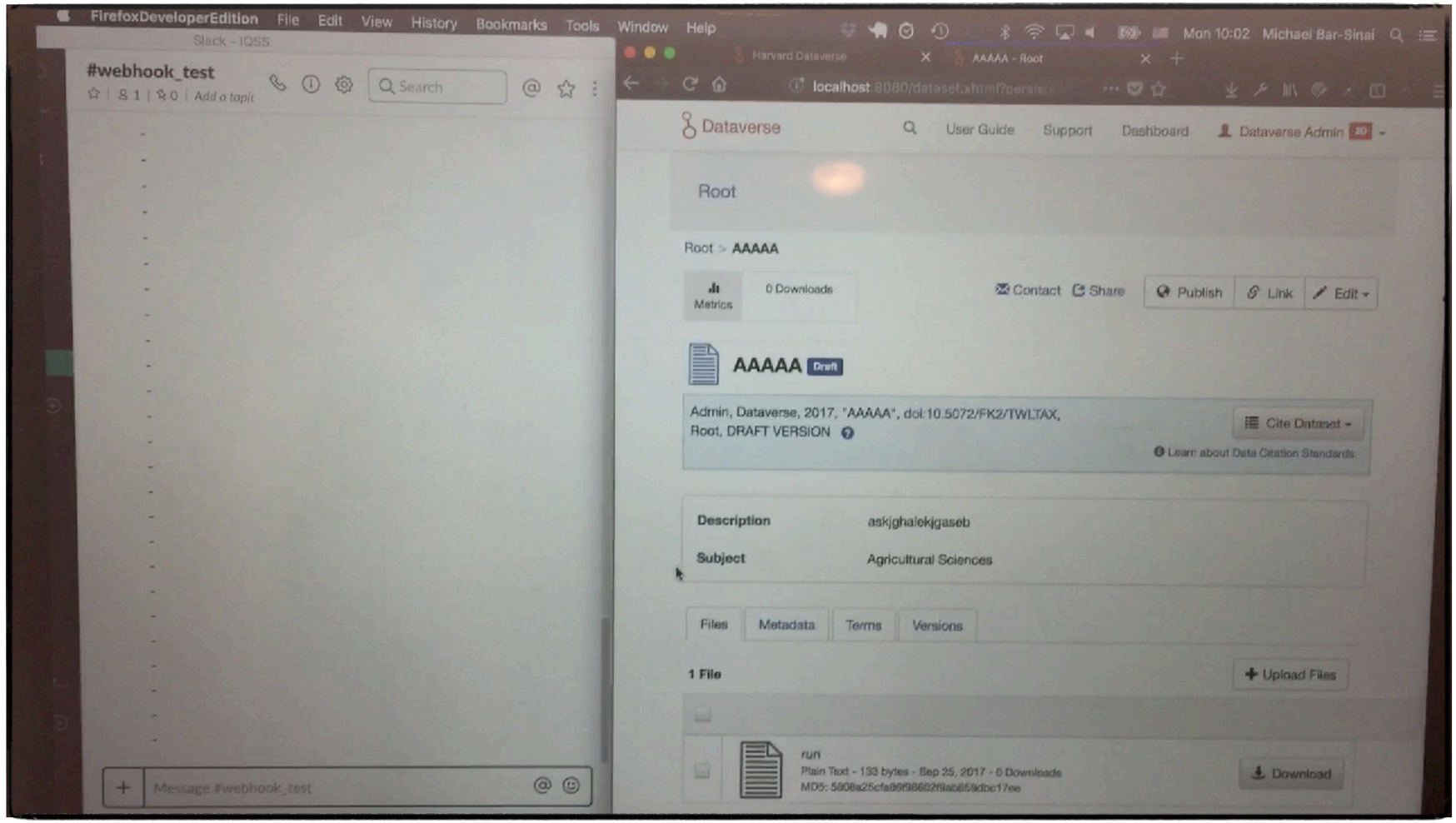


Store step data and context in DB

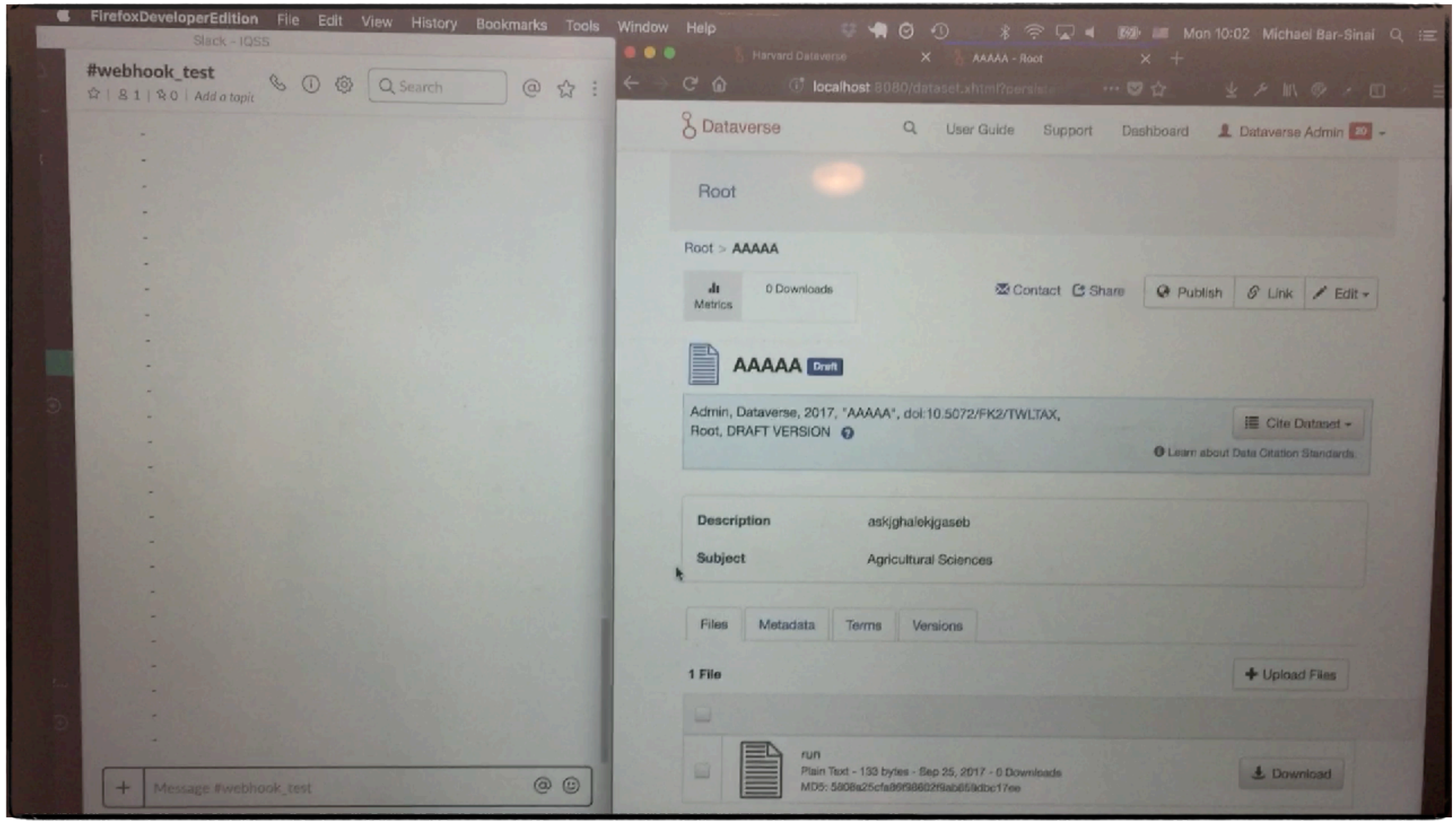
Restore step data and context from DB

Time —>

…But Does It Work?



…But Does It Work?



Workflow Mix

plugin + config + external standards

Codebase

- ✔ Size can go down as functionality is moved out
- ✔ Extension via configuration
- ✘ Need to add Infrastructure

Headache

- ✔ Empowers the community to add functionality on its own
- ✔ Core team not aware of remote systems at all
- ✘ A new public interface to maintain
- ✘ Security and stability may be an issue

Honorary mention

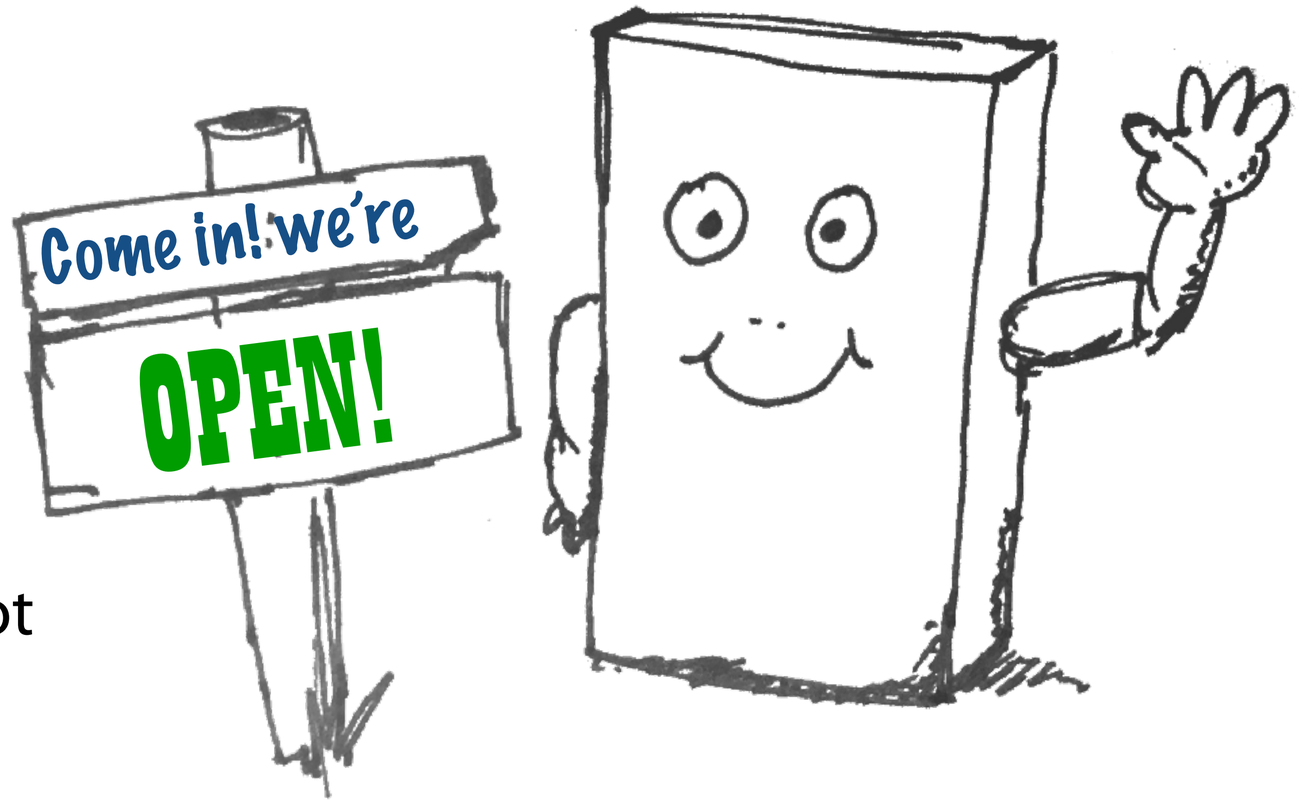
Nashorn and running user scripts

Summing Up

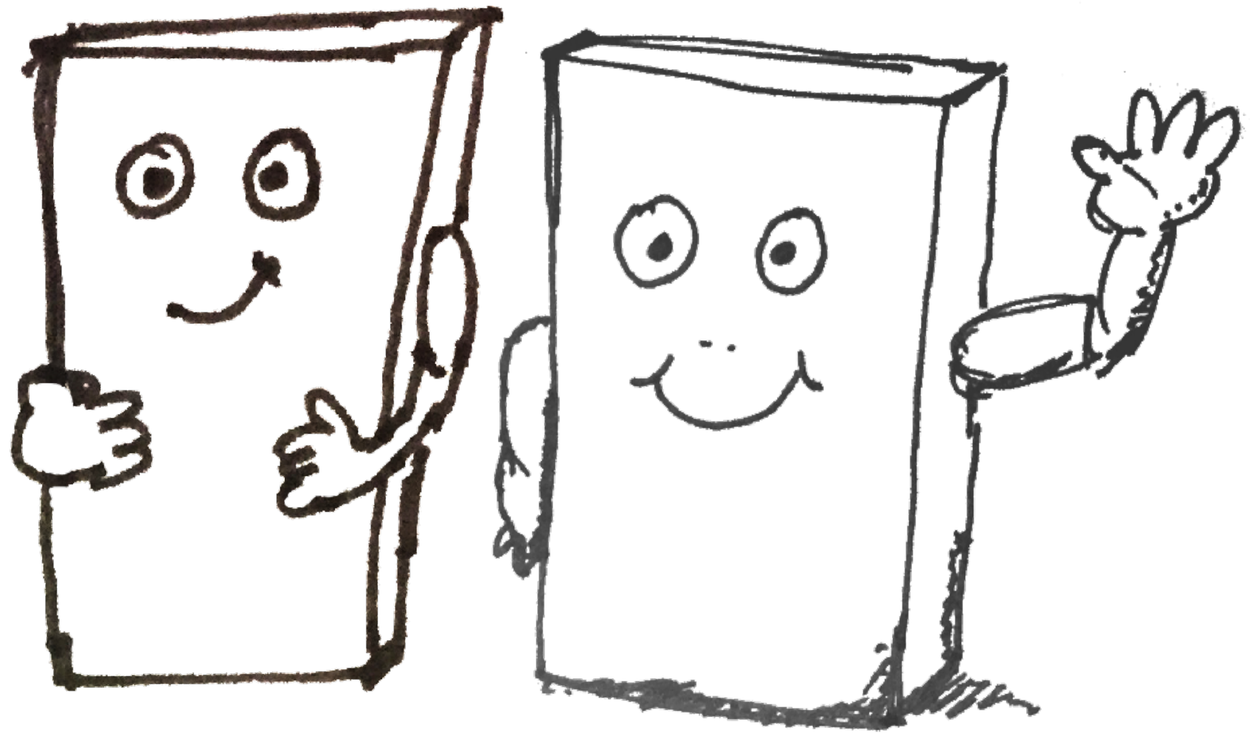
- Fun side effect: Modularization makes you think about your application and what parts of it are the core.

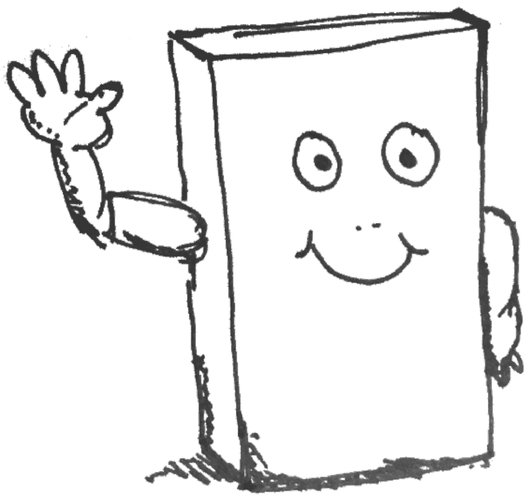
- There is a middle ground between Monolithic and MicroServices

- Each project can find its own sweet spot



Questions?

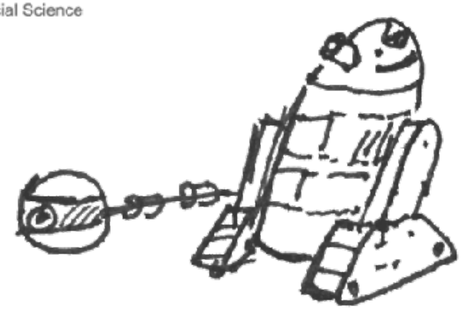




The Dataverse® Project

Thanks!

IQSS
The Institute for Quantitative Social Science



More IQSS Sessions at JavaOne 2017:

- BOF1594: **Introduction to Spark Streaming for Real Time Data Analysis**
  - Bob Treacy and Ellen Kraffmiller, Moscone West - Today, Room 2022 6:30-7:15pm
- BOF3429: **Herding Cats: Harvard Dataverse's Approach to Technical Community Management**
  - Gustavo Durand and Danny Brooke, Moscone West - Tomorrow, Room 2022 6:30-7:15pm
- BOF2805: **How to Run a Successful Open Source Java EE Project**
  - Philip Durbin and Stephen Kraffmiller, Moscone West - Tomorrow, Room 2022 Date: 7:30-8:15pm