a Distributed Testing UI

Andreea Babiuc
Chris Dillon
James Elford
Simon Evans
Andy Gurden

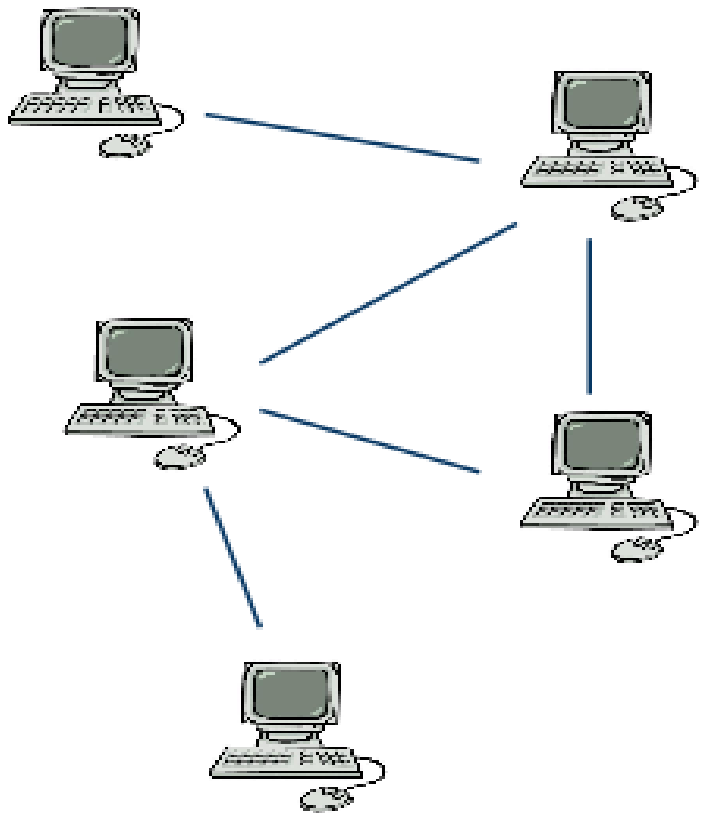
Overview of Talk

1. Motivation
   - How to, and why, test distributed systems?
   - What is Reef?
2. Reef Design Overview
   - Vazels overview
   - Client/server structure
3. Running Reef
4. Implementation & Architecture of Reef
   - Low-level Reef overview
   - Integration with Vazels
5. Evaluation/Conclusion
   - Technical/collaboration challenges
   - Successes/shortcomings

Distributed Systems Testing

Distributed Systems

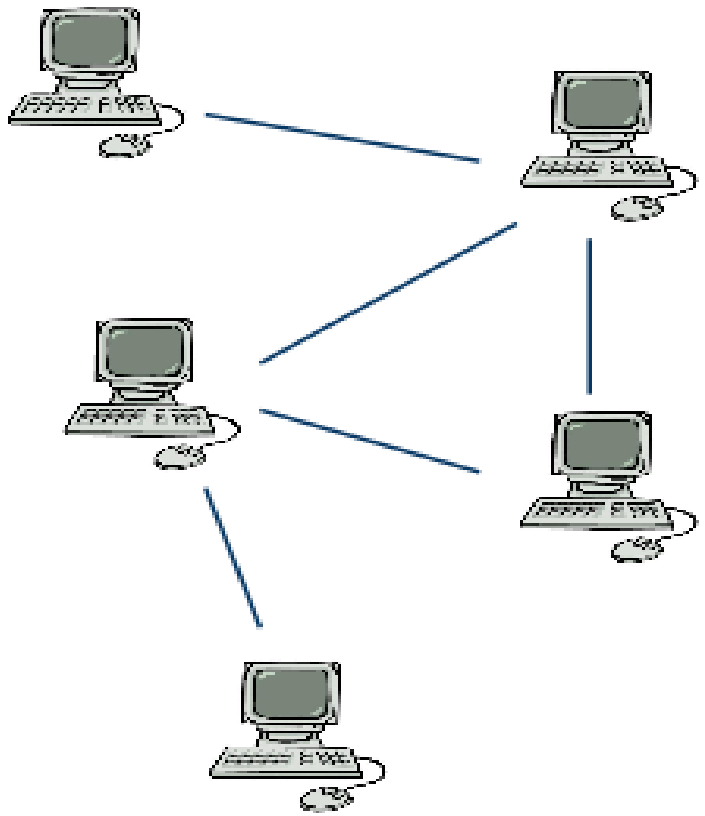
- Examples of uses - modelling, most major websites (e.g. Amazon), BitTorrent
- Hard to test:
  - Test where?
  - How to conduct the test?



Distributed Systems Testing

Distributed Systems

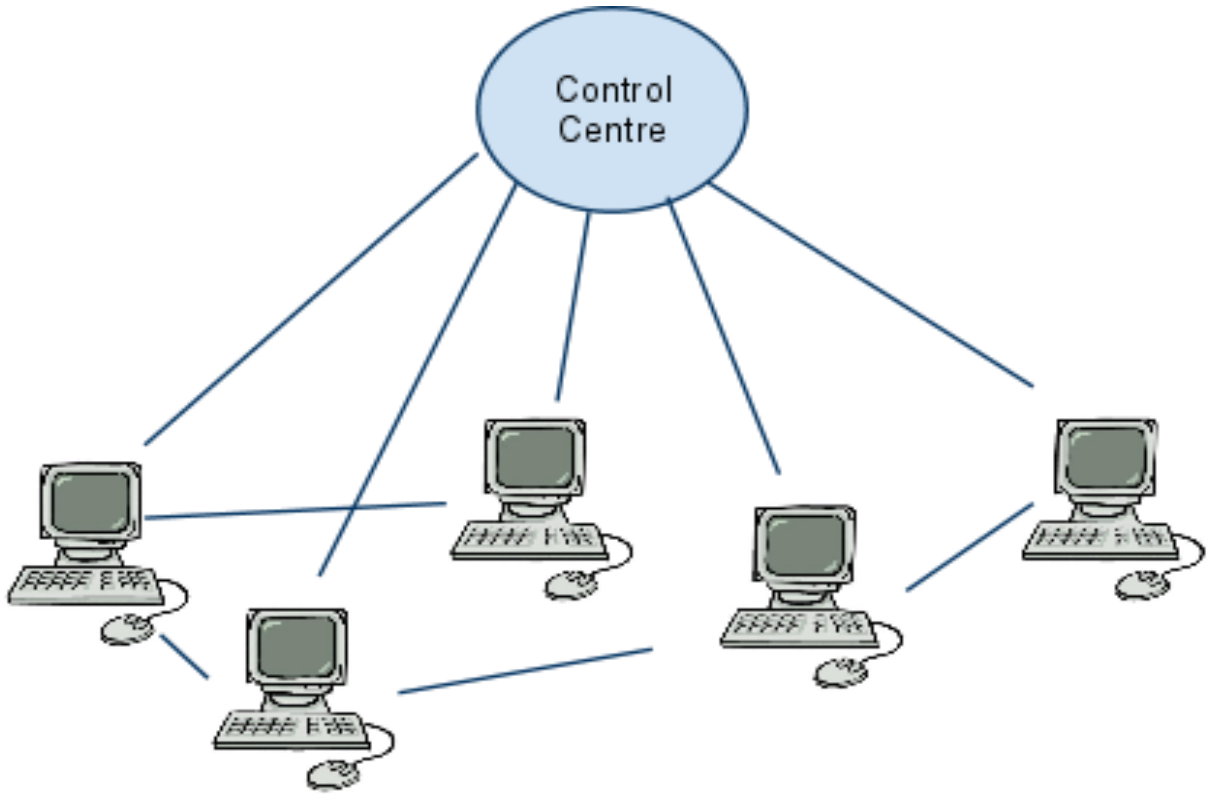
- Examples of uses - modelling, most major websites (e.g. Amazon), BitTorrent
- Hard to test:
  - Test where?
  - How to conduct the test?



To make testing easier over a distributed system, there are testbeds, e.g. PlanetLab

Distributed Systems Testing

To help test on these testbeds, there are tools to aid with automating common tasks, e.g. deployment, network topology, collection of results.



Reef

Reef is one of these tools for the automated running of tests on distributed systems.

The command-line tool Vazels is used as the engine for Reef. Vazels runs the experiment, given the configuration by Reef.

Our tool, Reef, is designed to help run tests on a testbed, such as PlanetLab. We will call the system being tested the **System Under Experimentation (SUE)**, and a group of tests upon the SUE will form an **experiment**.

Reef

Capabilities:

"**Vazels** is suitable for [a] wide range of use-cases – functional testing, performance analysis, analysis on the resilience to failures, analysis on the resilience to DDoS attacks, and execution of distributed batch jobs." (*www.vazels.org*)

Reef

Reef uses a web interface - why?

- Usable across platforms
- Lightweight - minimal software configuration - web client application particularly accessible

In a sentence:

Reef provides an easy-to-use graphical front-end for users on a remote system, through a thin web-client and a Python server back-end.

Alternatives:

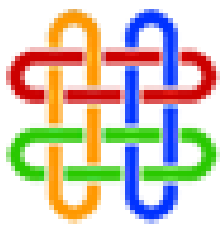
App server side (Java Swing), issues with X-forwarding
App client side (Java RMI), issues with hostnames

Technological Overview

- Reef built upon many technologies including:
  - Vazels
  - GWT
  - restlite
  - WSGI
  - AJAX
- Vazels has the most influence on our design.

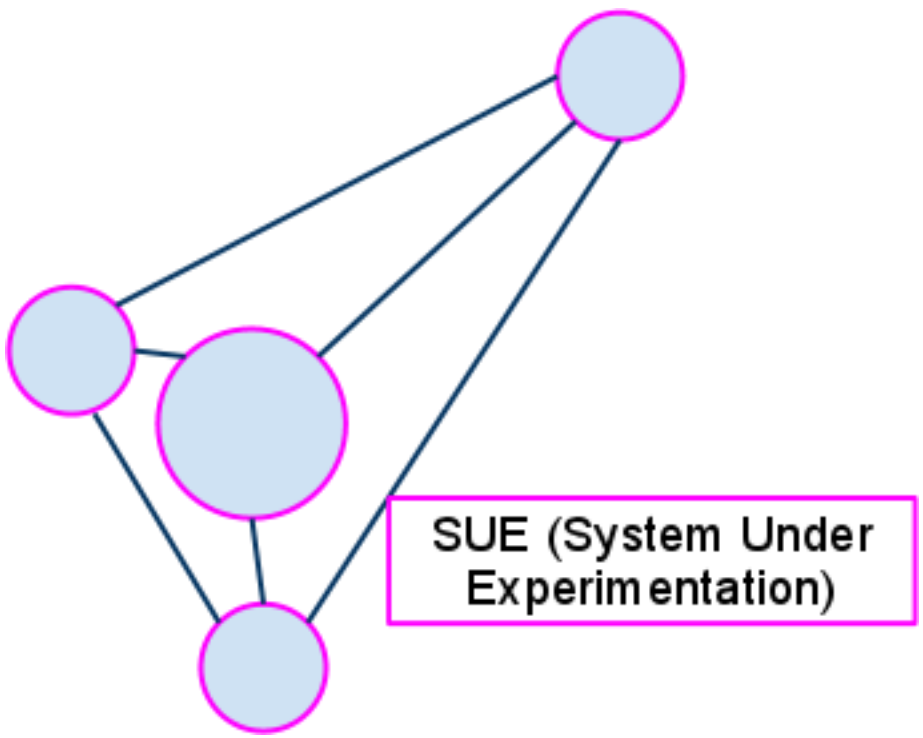
What is Vazels?

- Distributed testing engine
- Designed for command-line user input - no API
- How does it work?
  - Let's see...

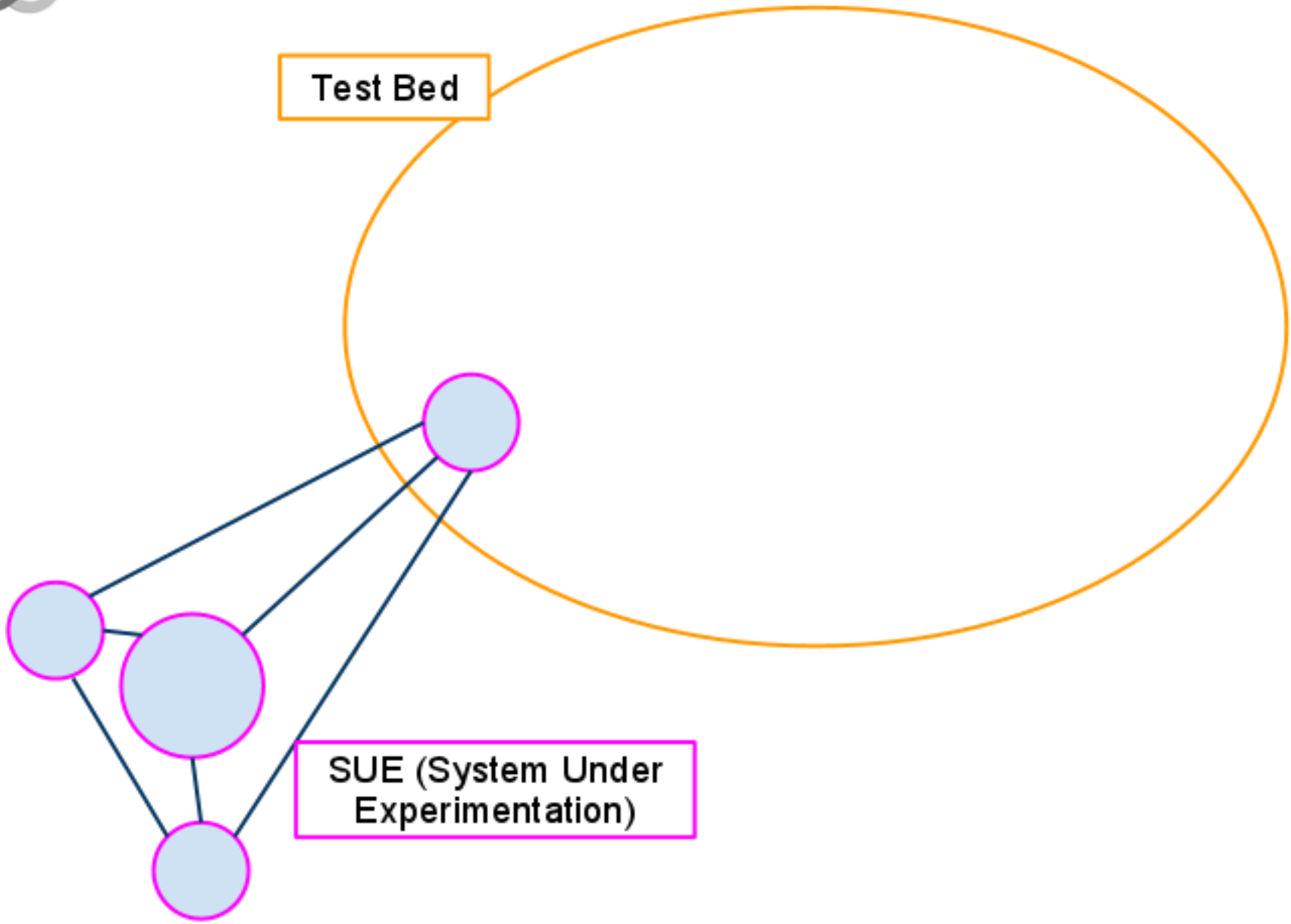


VAZELS

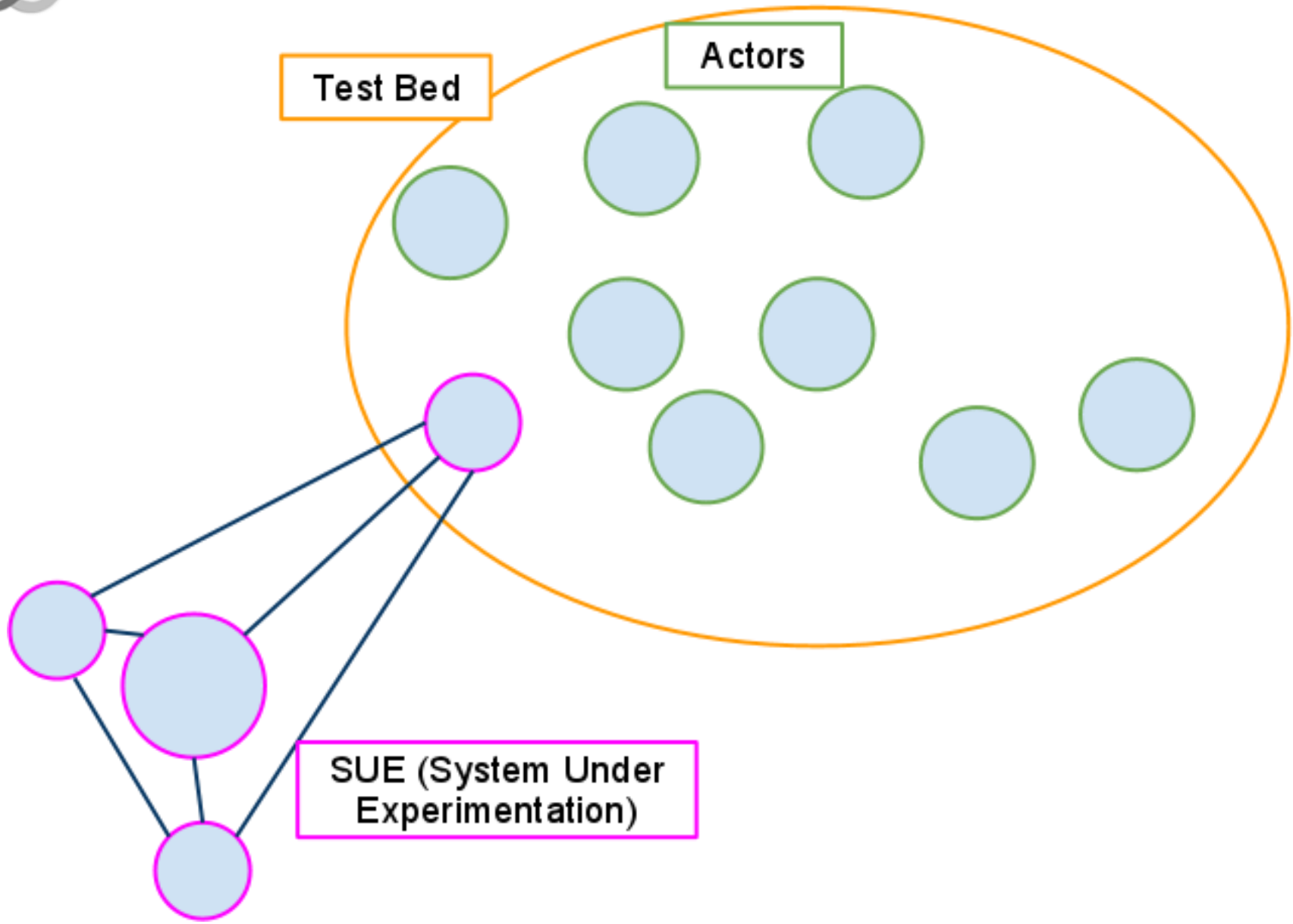
How does it work?



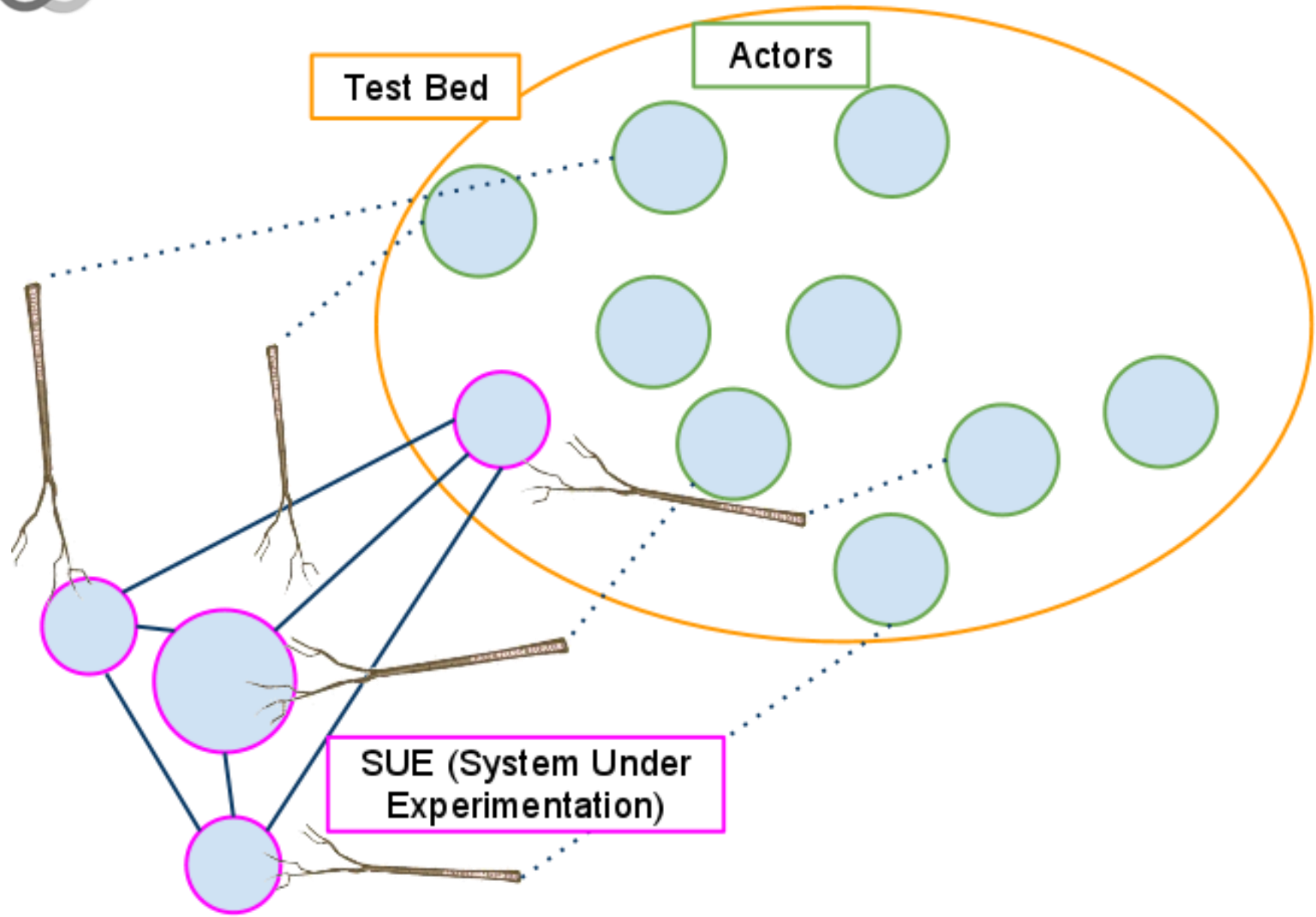
How does it work?



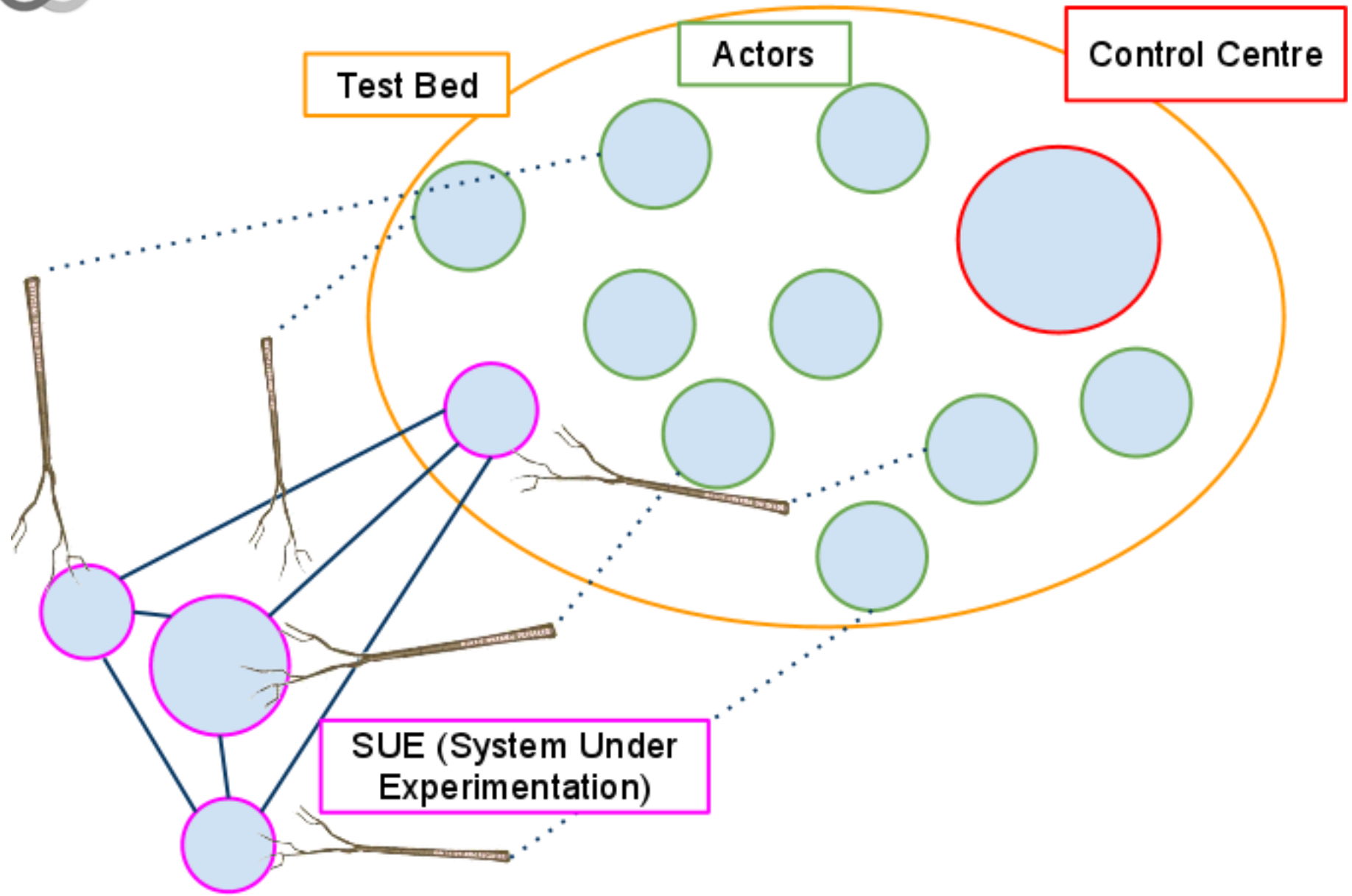
How does it work?



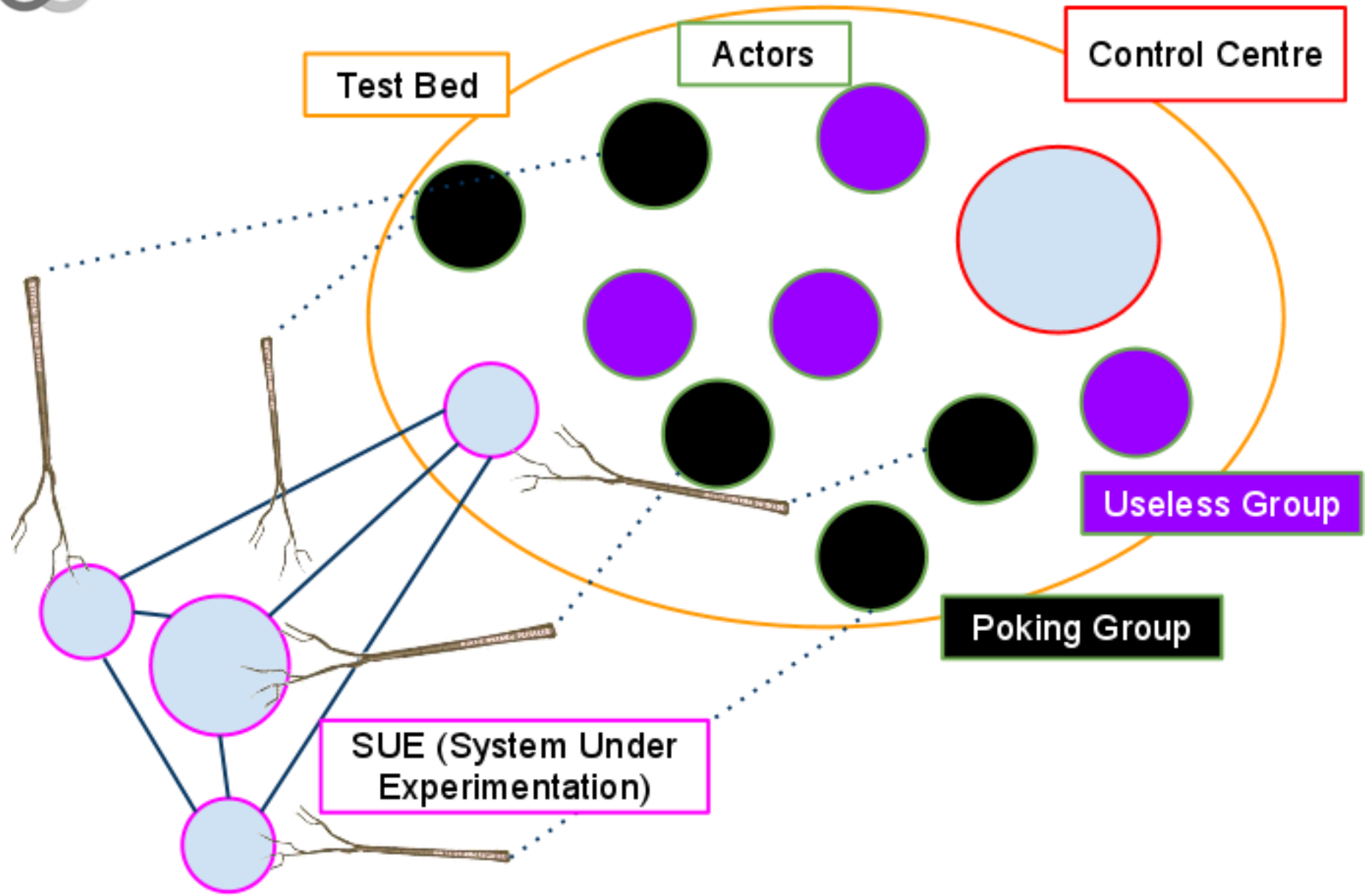
How does it work?



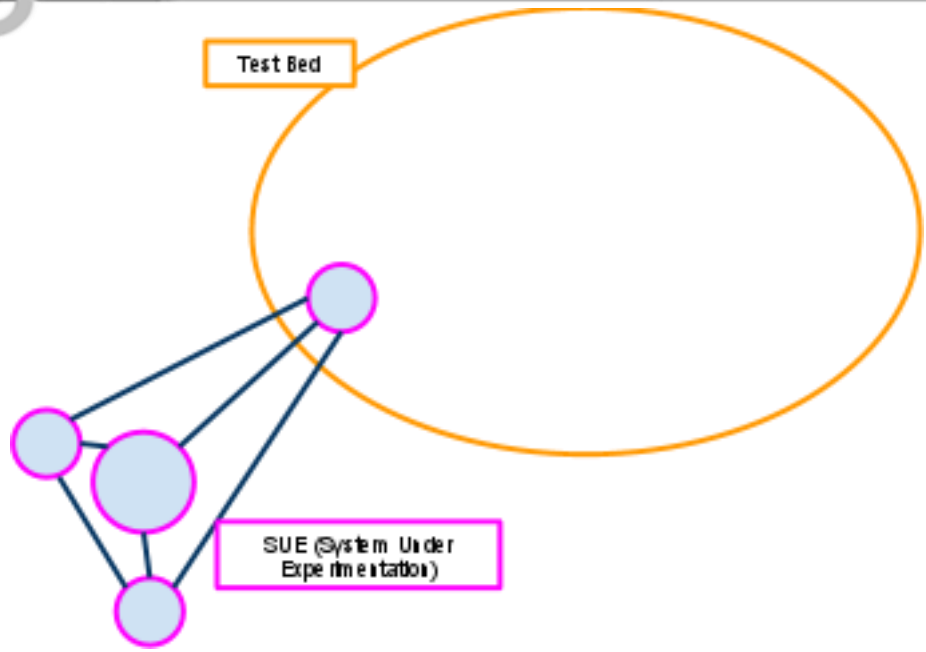
How does it work?



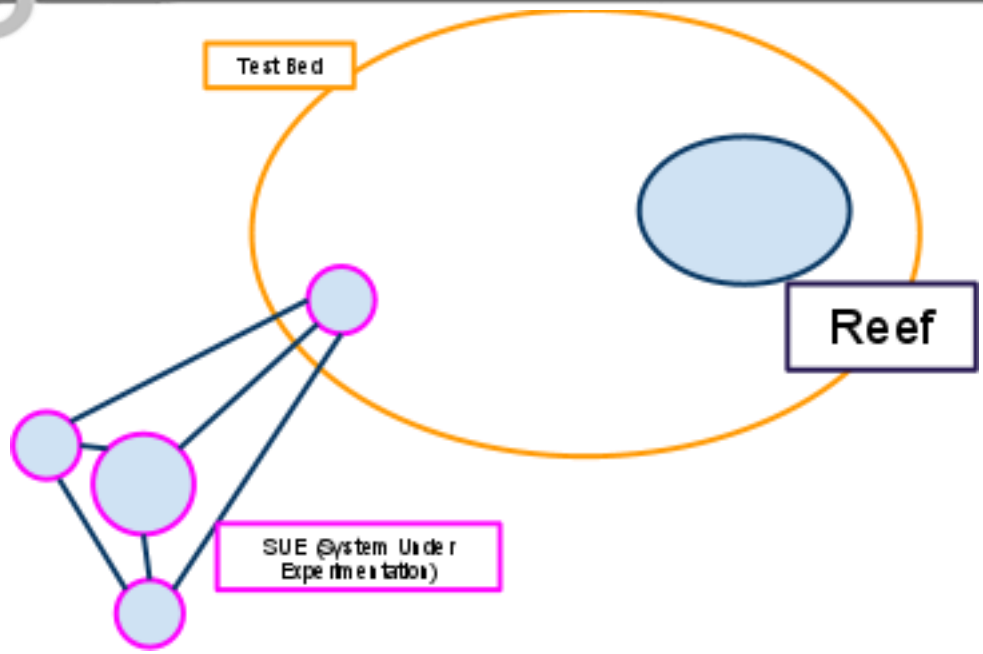
How does it work?



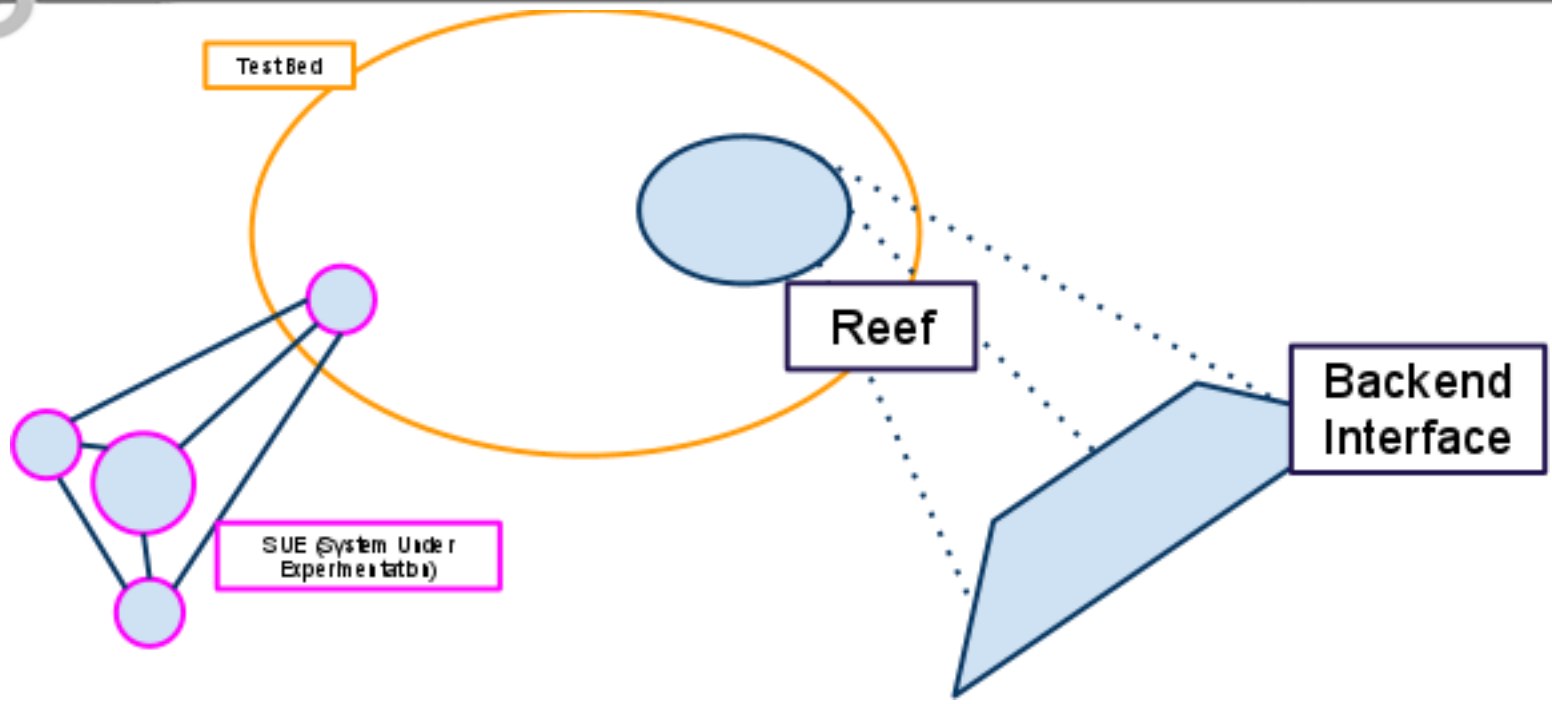
Interacting with Reef



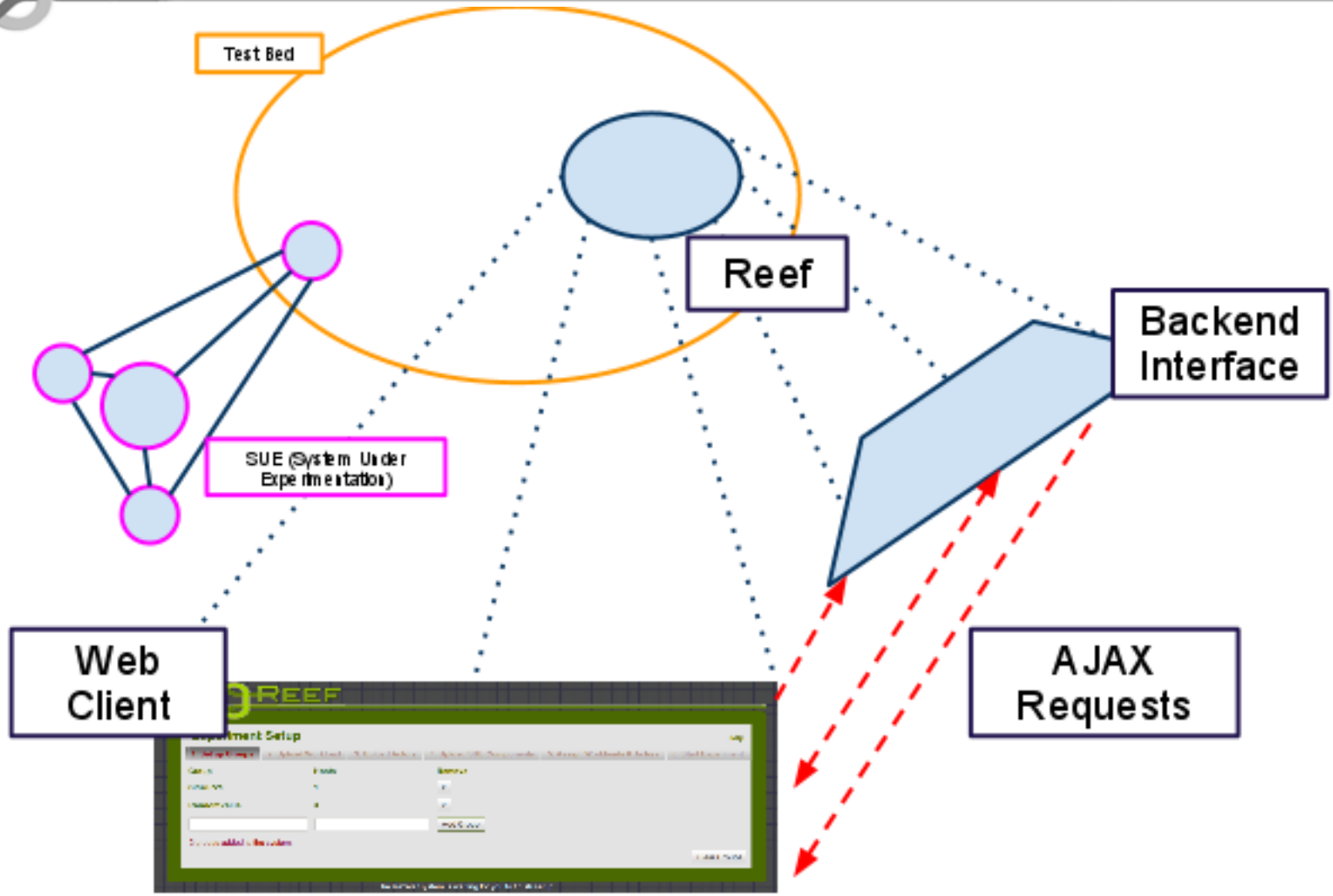
Interacting with Reef



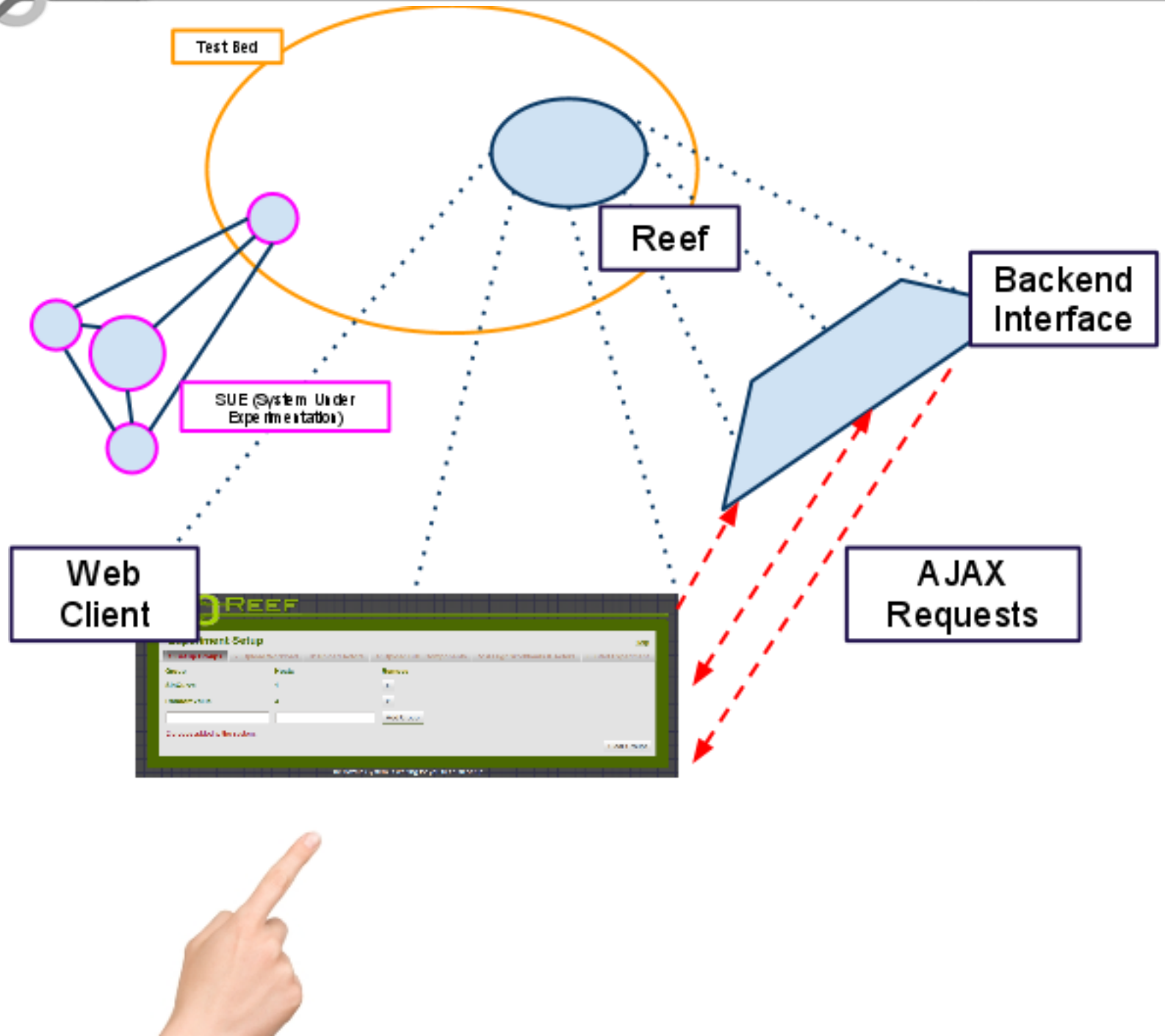
Interacting with Reef



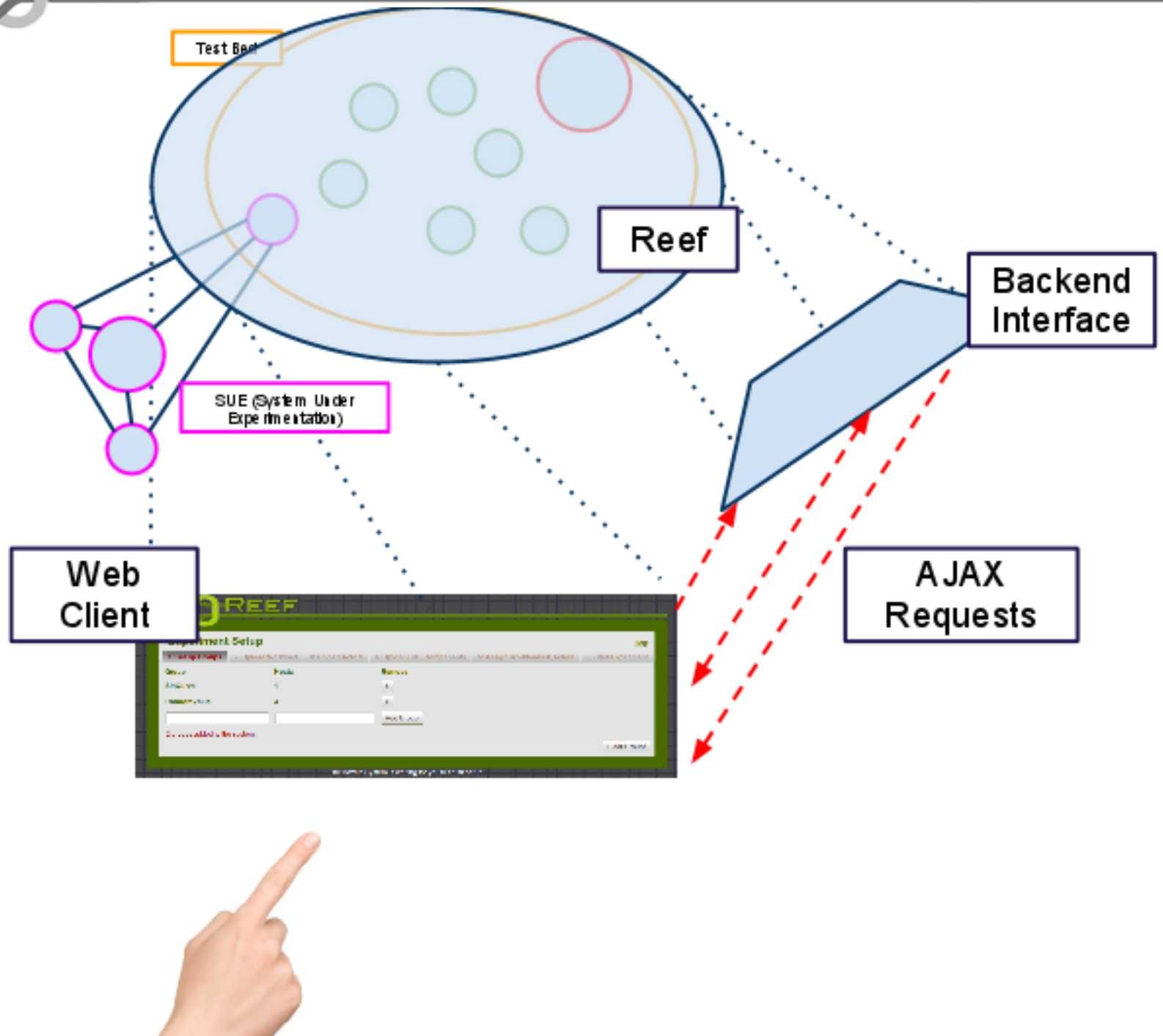
Interacting with Reef



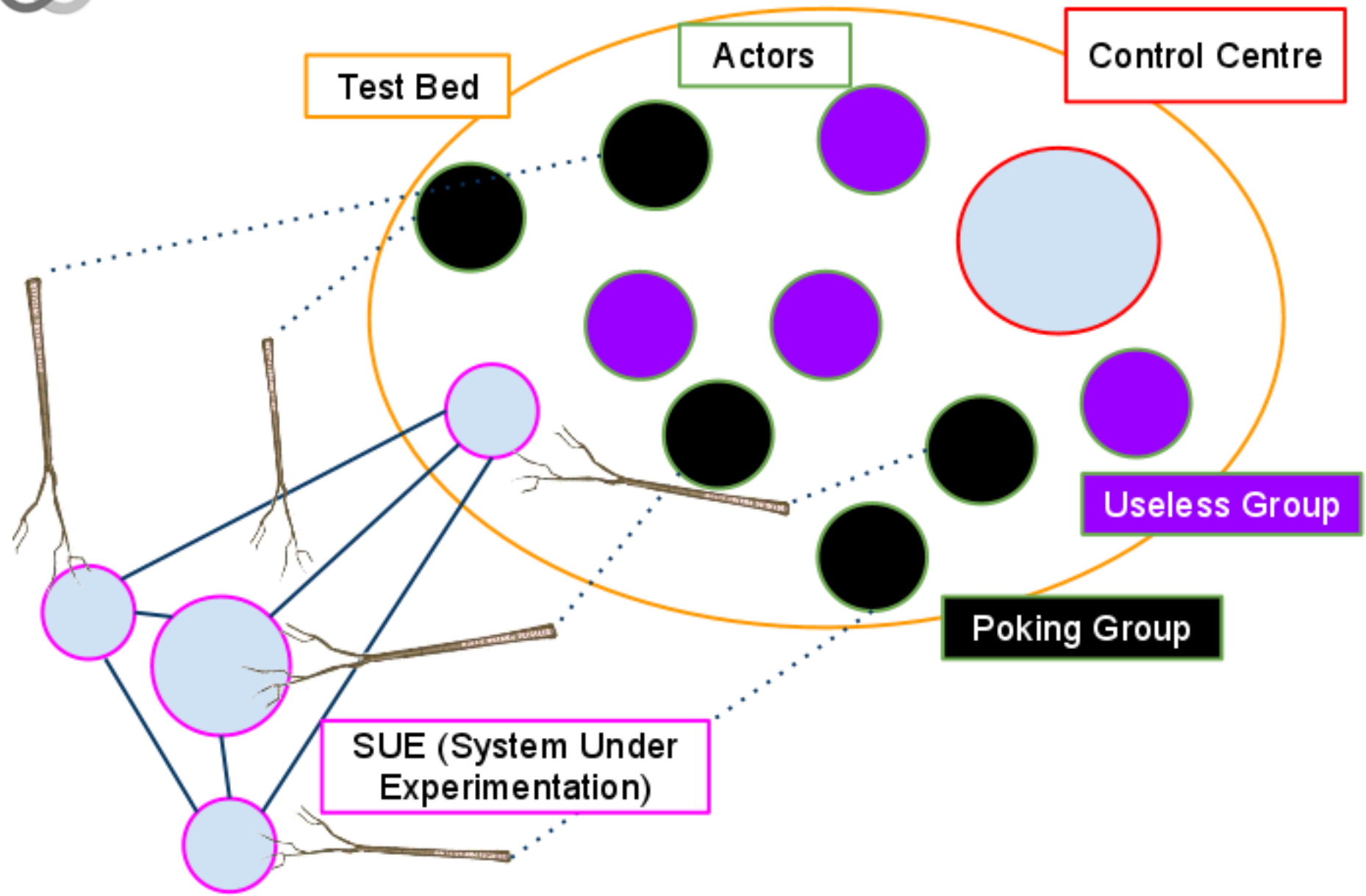
Interacting with Reef



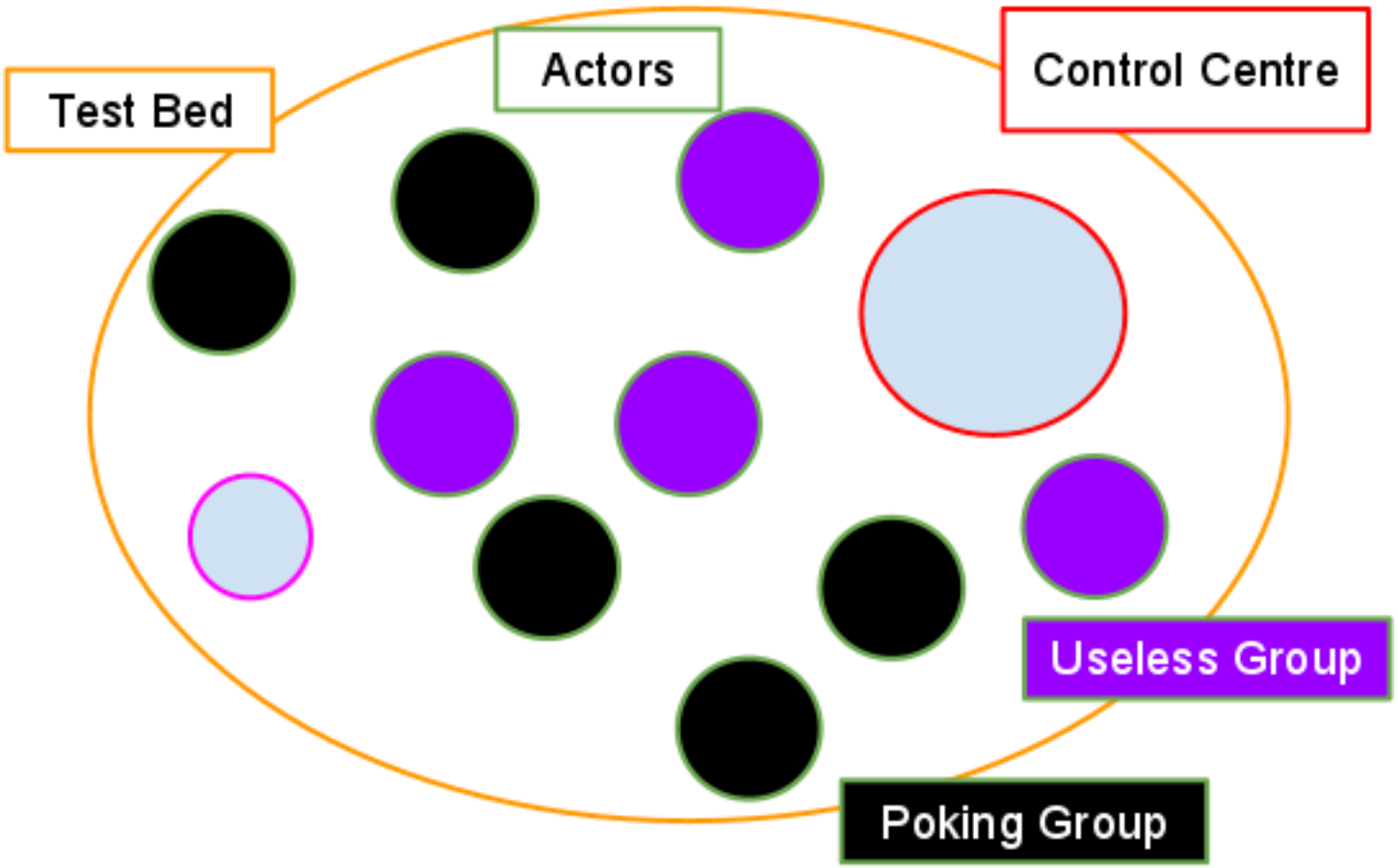
Interacting with Reef



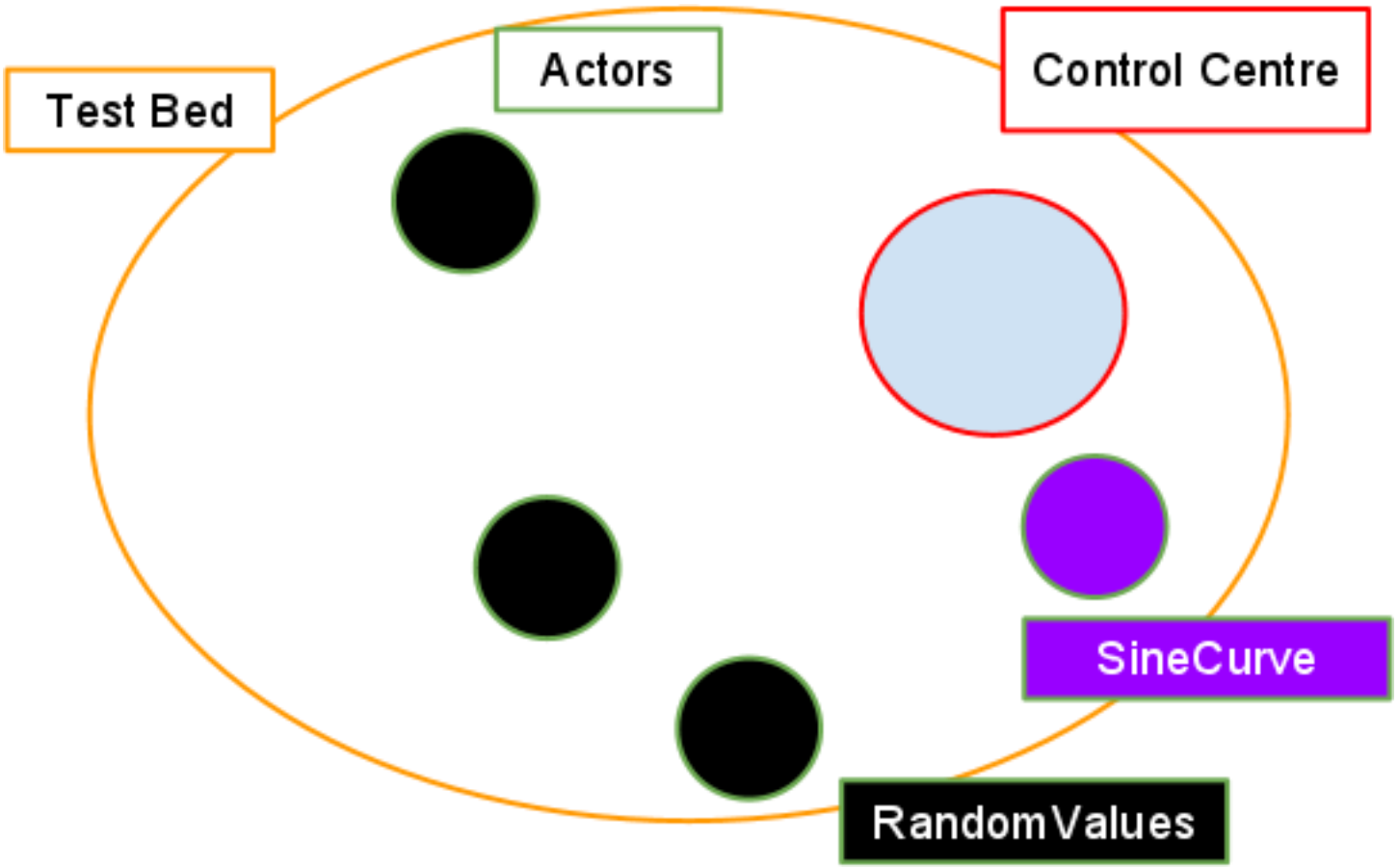
Demo Experiment



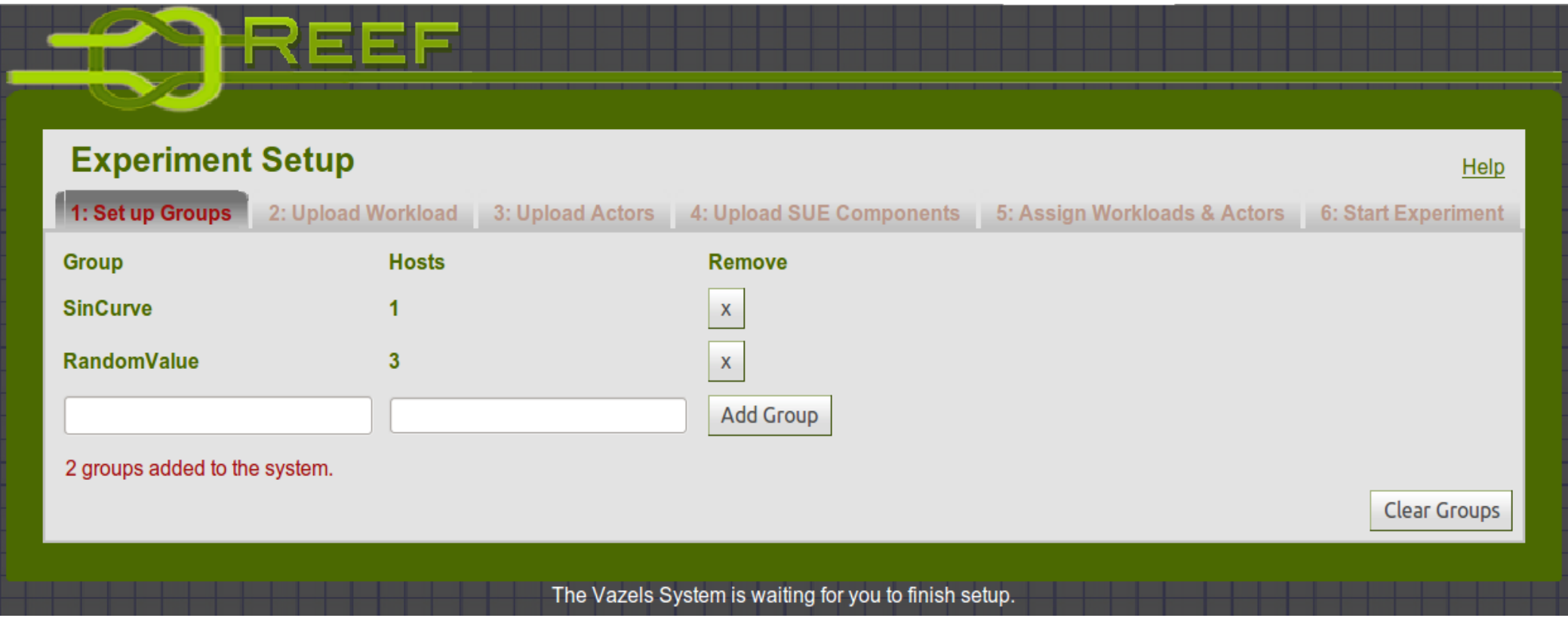
Demo Experiment



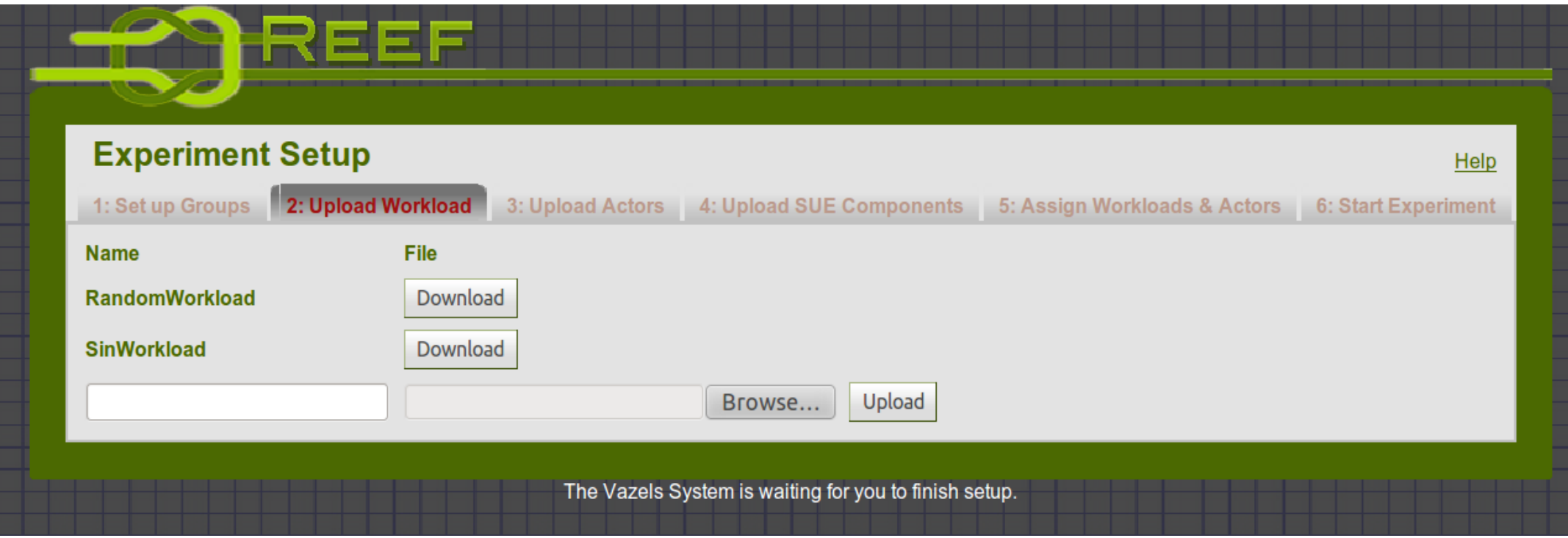
Demo Experiment



Add Groups


REEF
Experiment Setup
Help
1: Set up Groups
2: Upload Workload
3: Upload Actors
4: Upload SUE Components
5: Assign Workloads & Actors
6: Start Experiment
Group
Hosts
Remove
SinCurve
1
x
RandomValue
3
x
Add Group
2 groups added to the system.
Clear Groups
The Vazels System is waiting for you to finish setup.

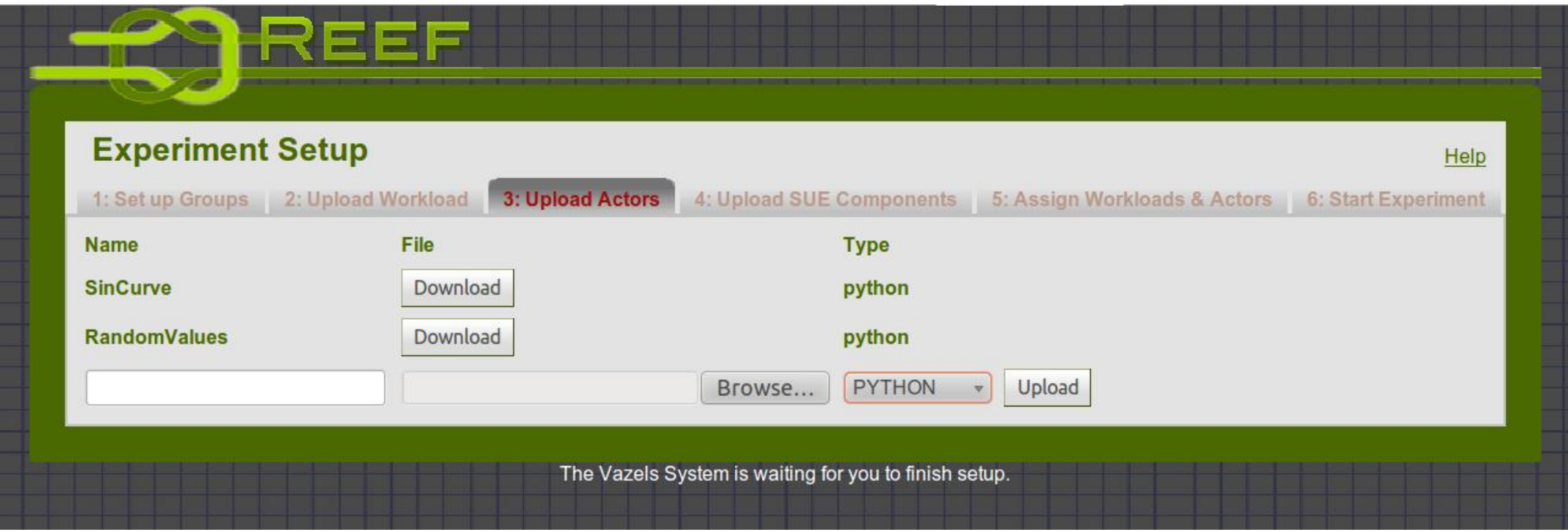
Upload Workloads



Sample Workload

```
actor {
    referenceName: "sin"
    type: PYTHON
    actorClass: "sin_actor.sin_curve"
}

invocationSequence {
    invocation {
        actorReference: "sin"
    }
    timePatternTrigger: "0 * 0 0 0"
}
```

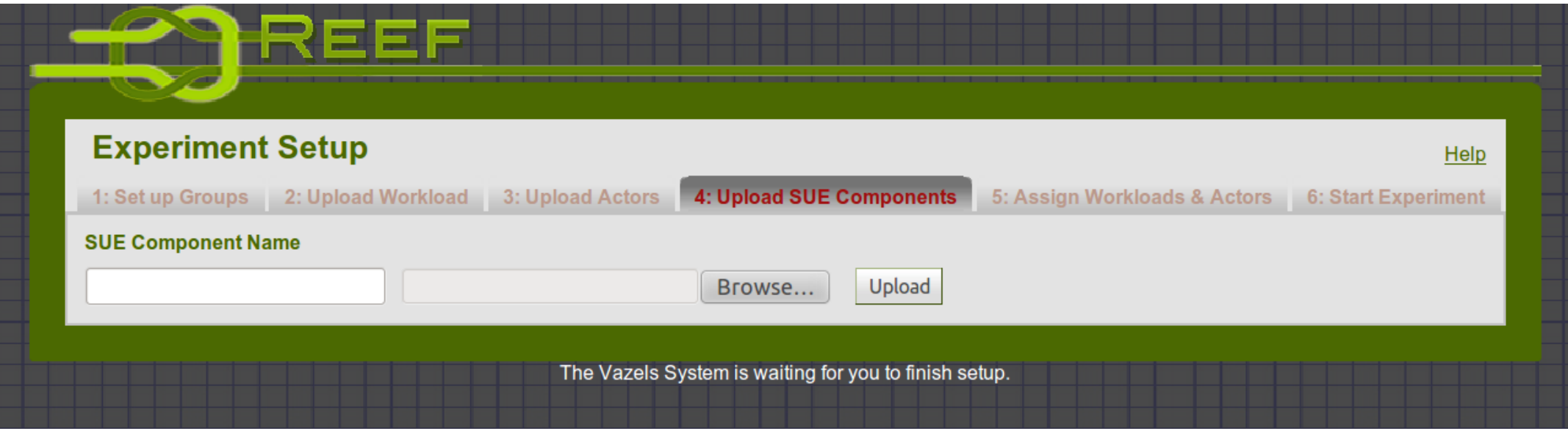
Upload Actors



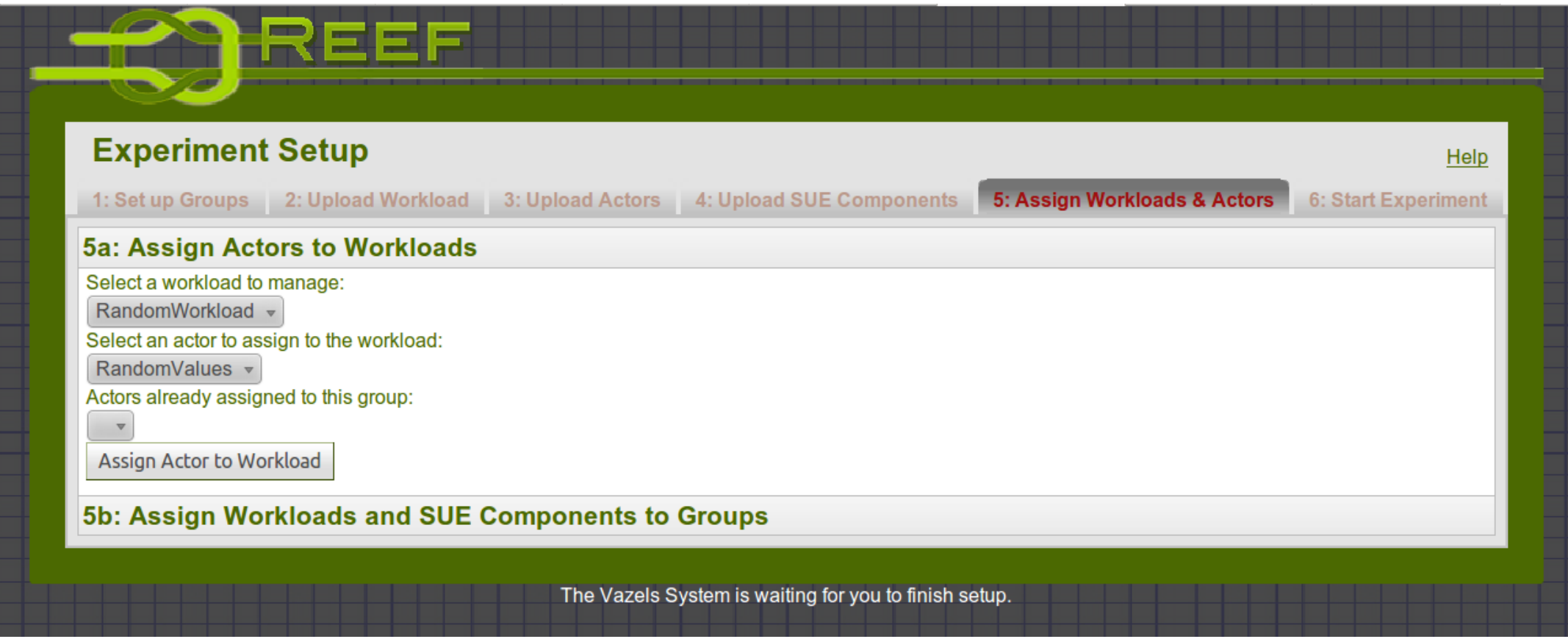
Sample Actor

```
class sin_curve:
    def invoke(self, snapshotsWriter, session):

     value = actor_pb2.Value()
     value.doubleValue = sin(time())
     snapshotsWriter.write_snapshot(
         "sin_curve", value)

     return True
```

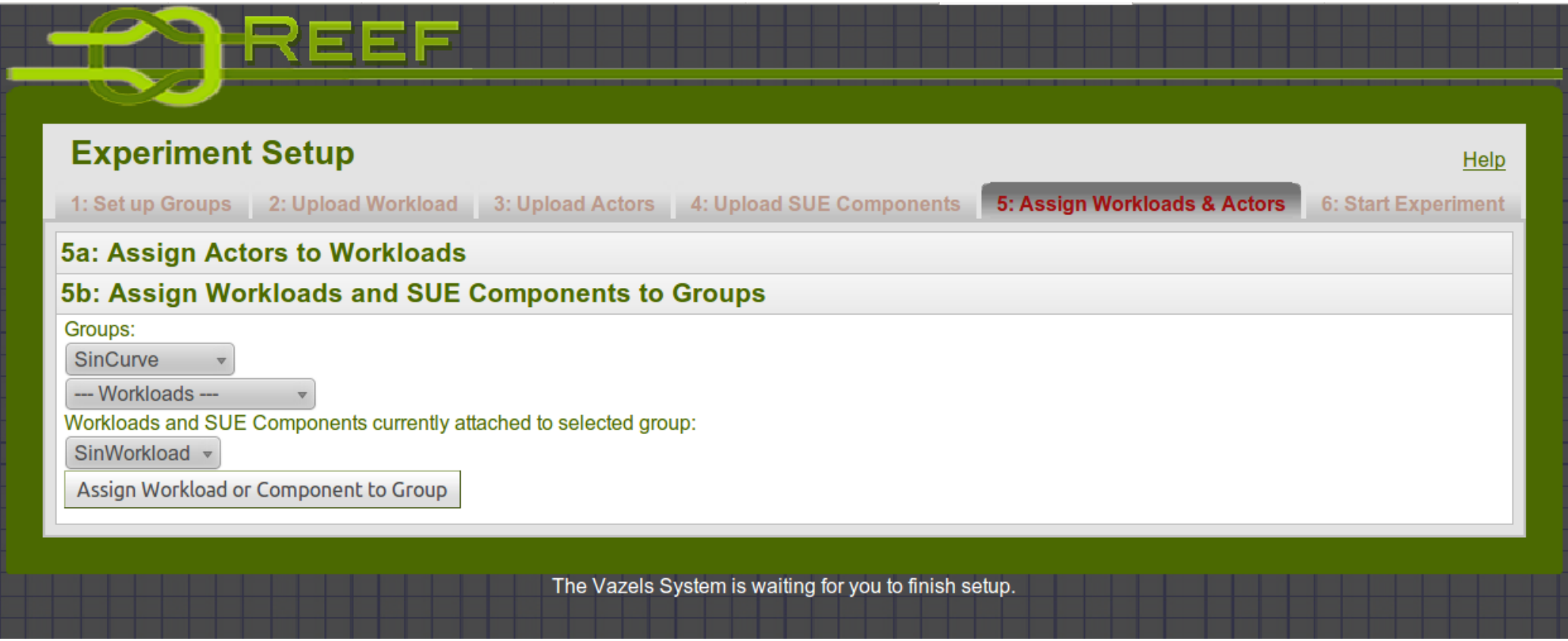
Upload SUE Components



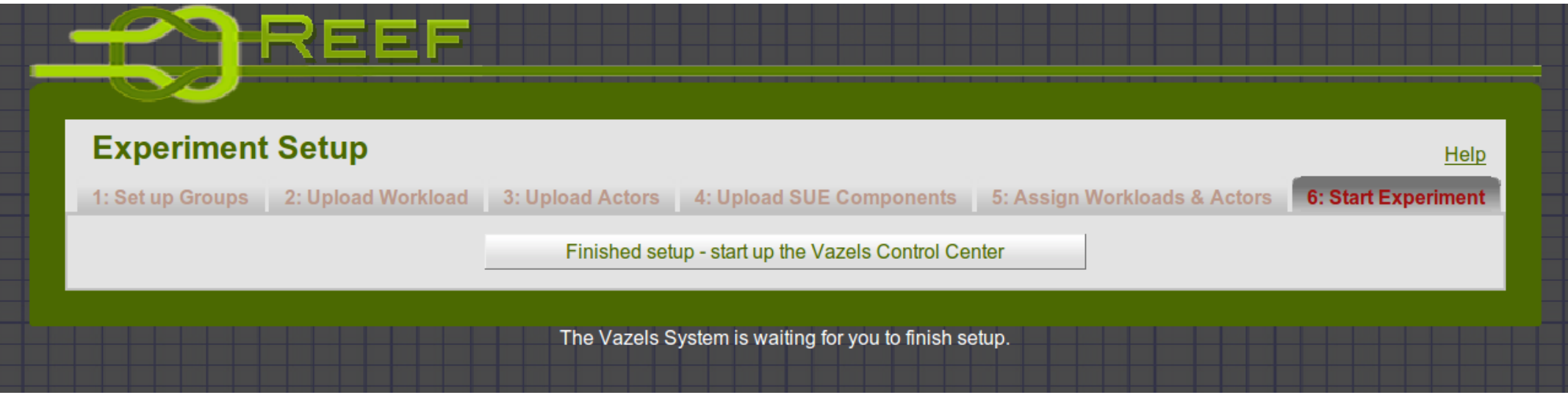
Assign Actors to Workloads



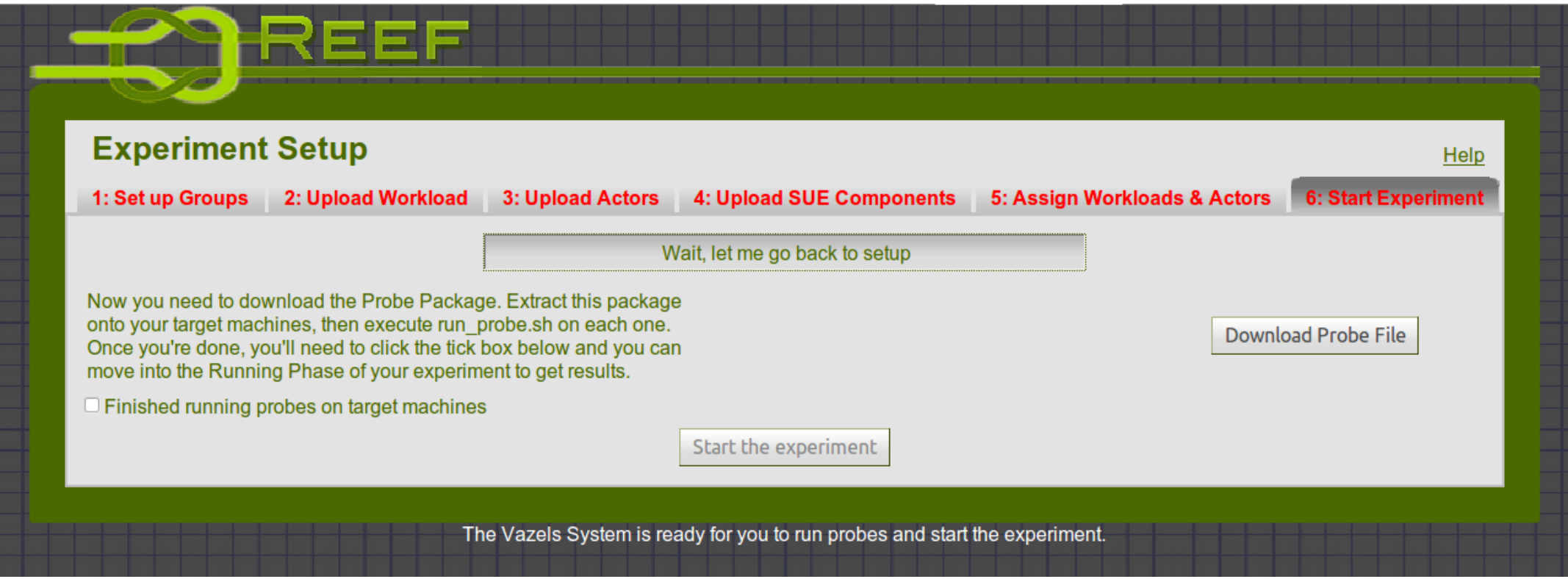
Assign Workloads to Groups



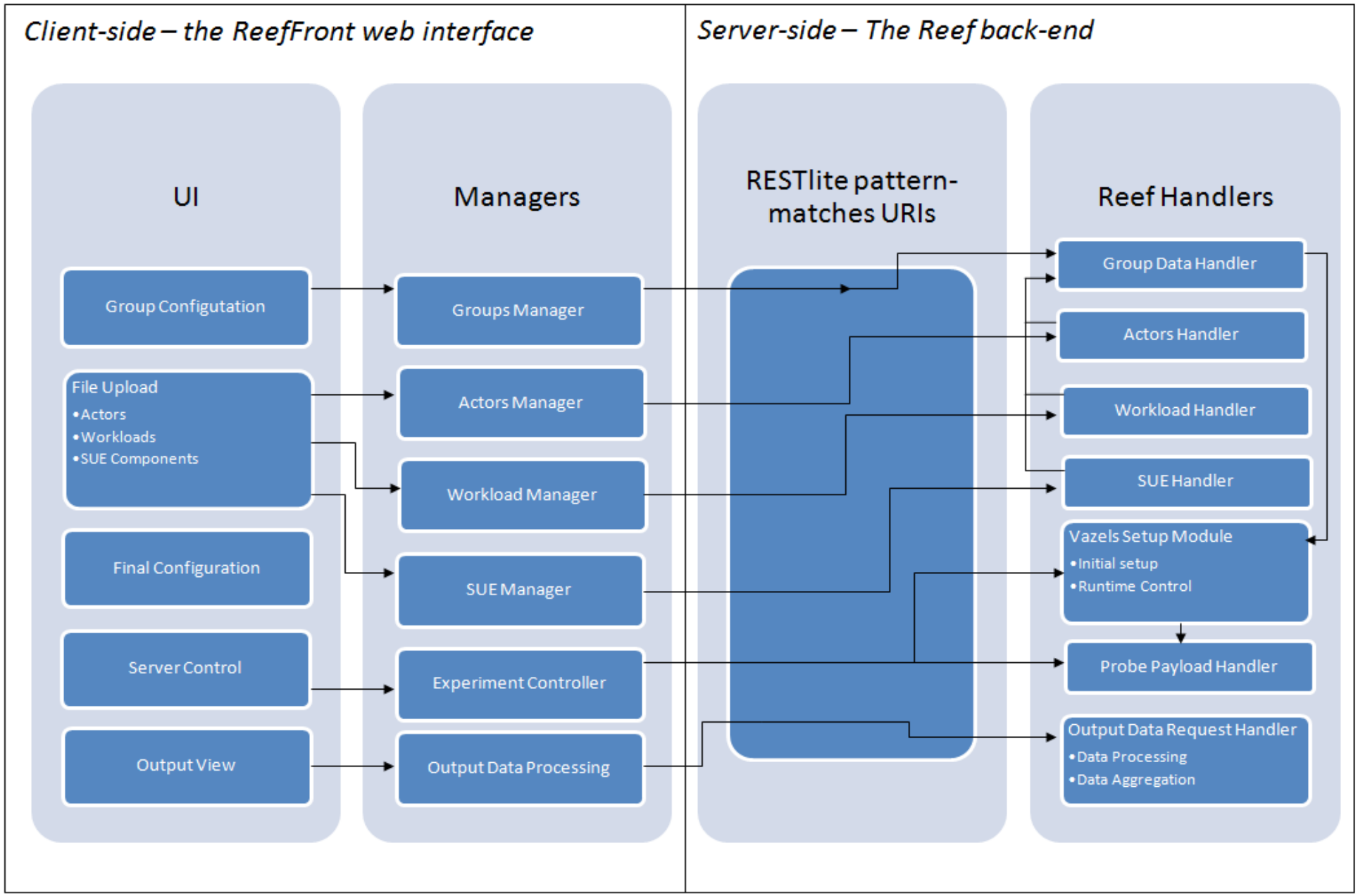
Confirming User has Finished



Start the Experiment

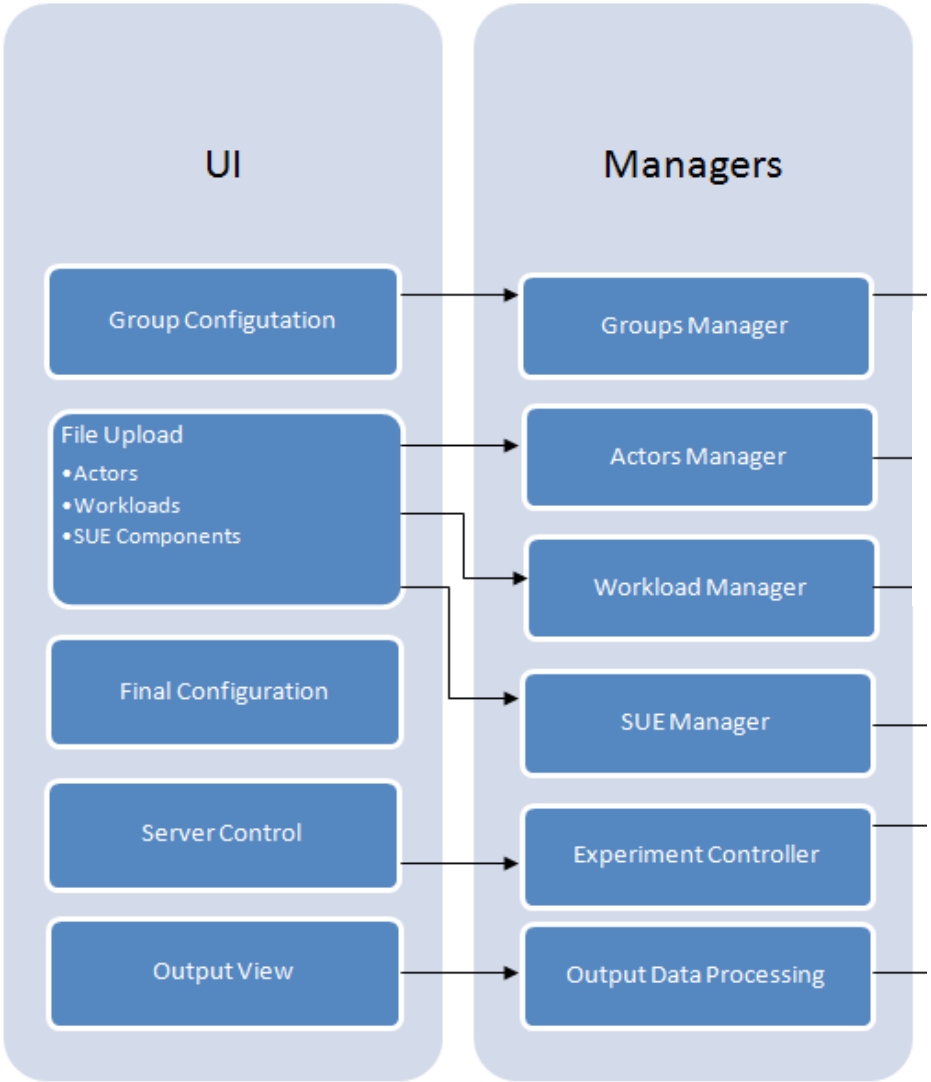


The Architecture

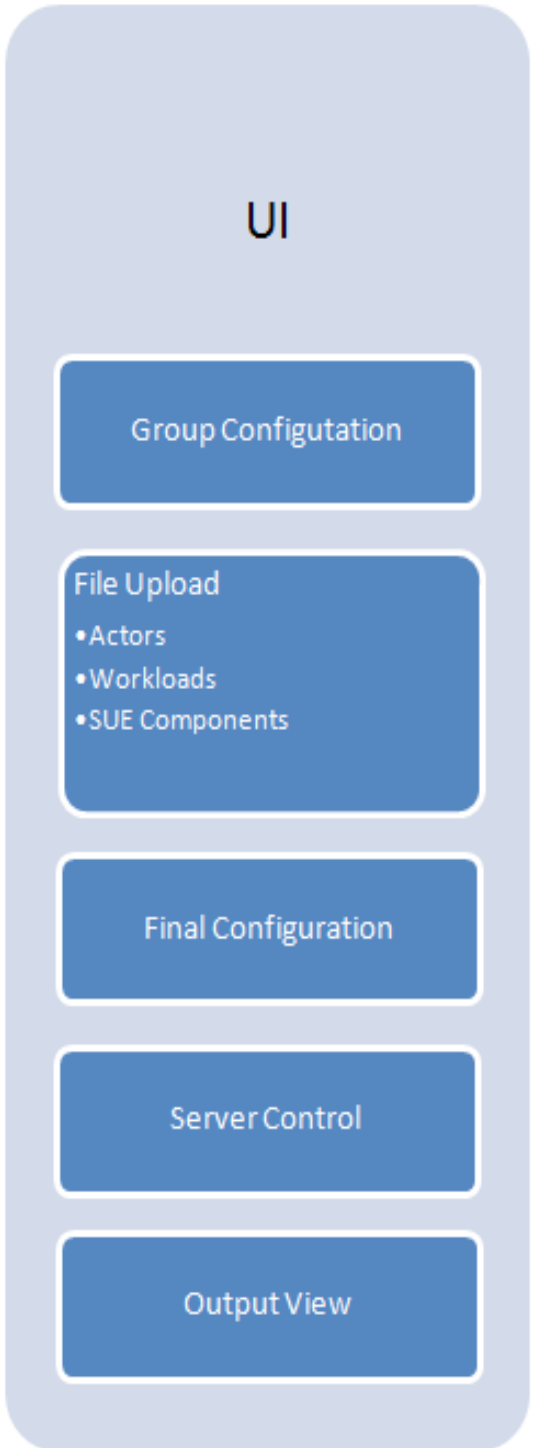


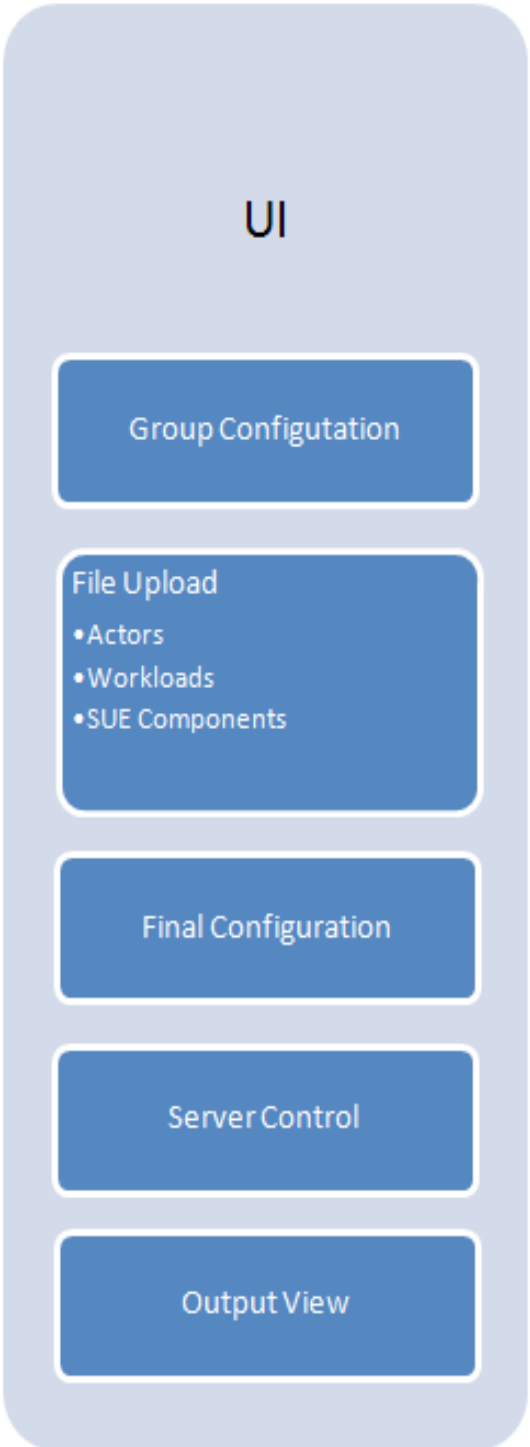
The Architecture

Client-side – the ReefFront web interface

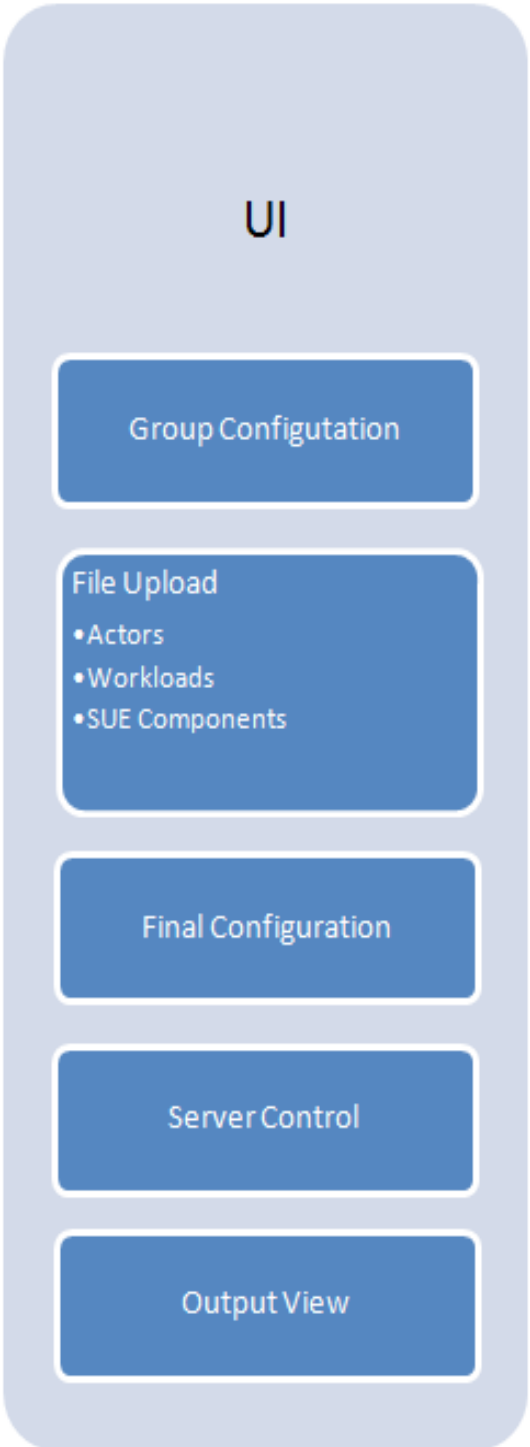


- Written entirely in Java
- Translated by GWT
- Closely follows MVC

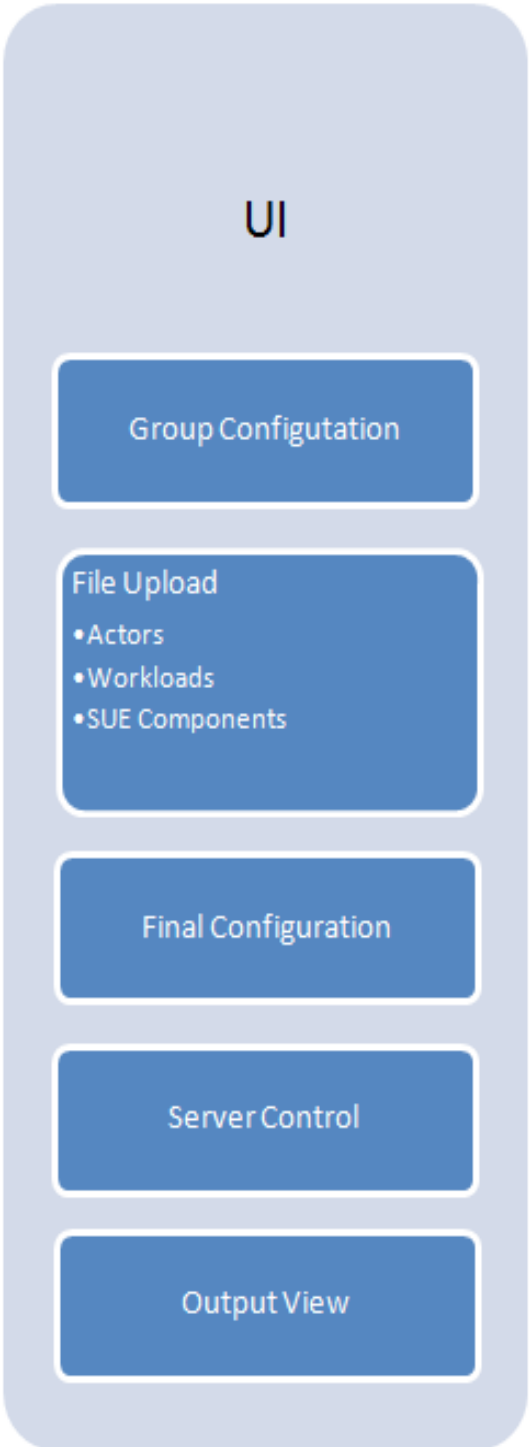

UI
Group Configutation
File Upload
•Actors
•Workloads
•SUE Components
Final Configuration
Server Control
Output View



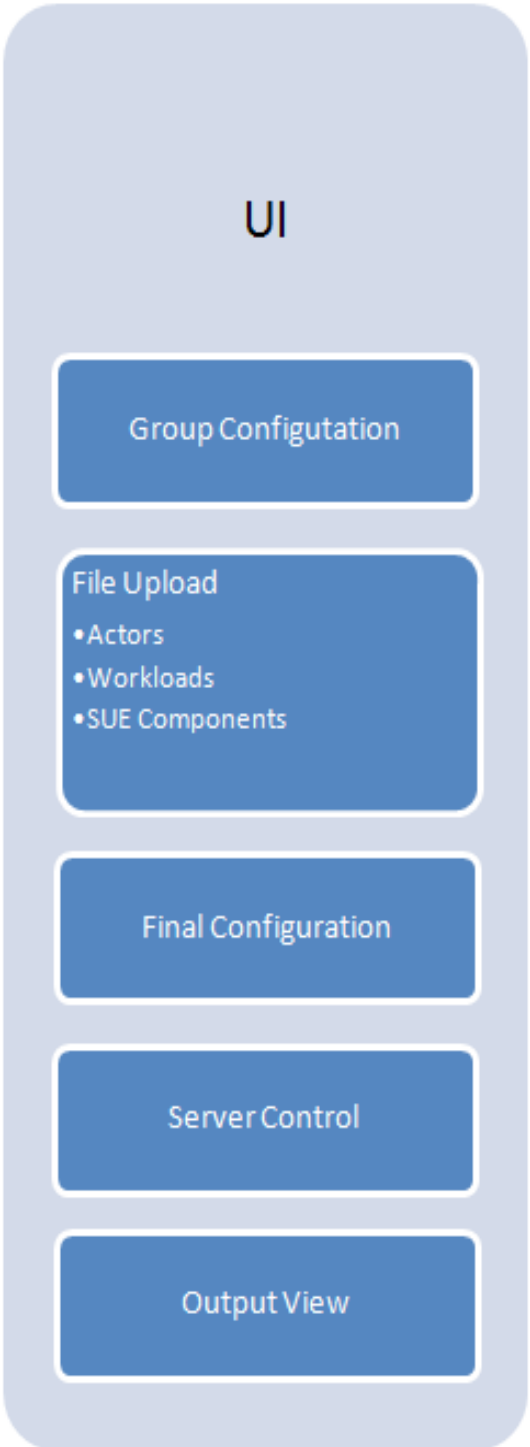
- Specify how many groups are required, their names, and their sizes



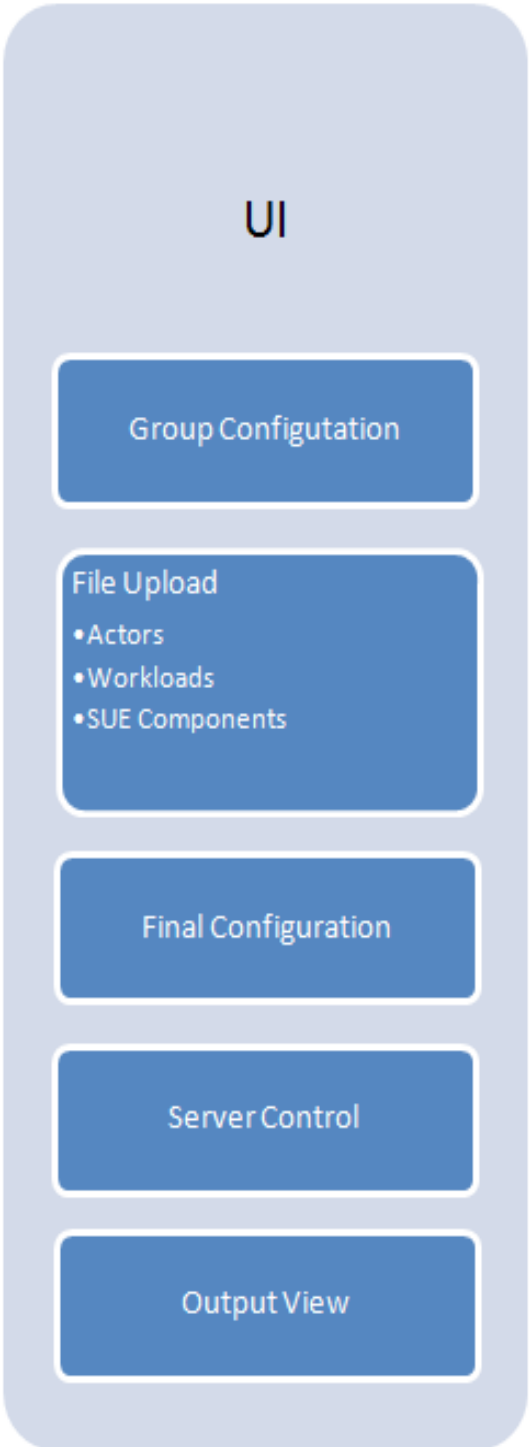
- Specify how many groups are required, their names, and their sizes
- Upload required files to the server



- Specify how many groups are required, their names, and their sizes
- Upload required files to the server
- Now that the server has all the files it needs, tell it how to set them up



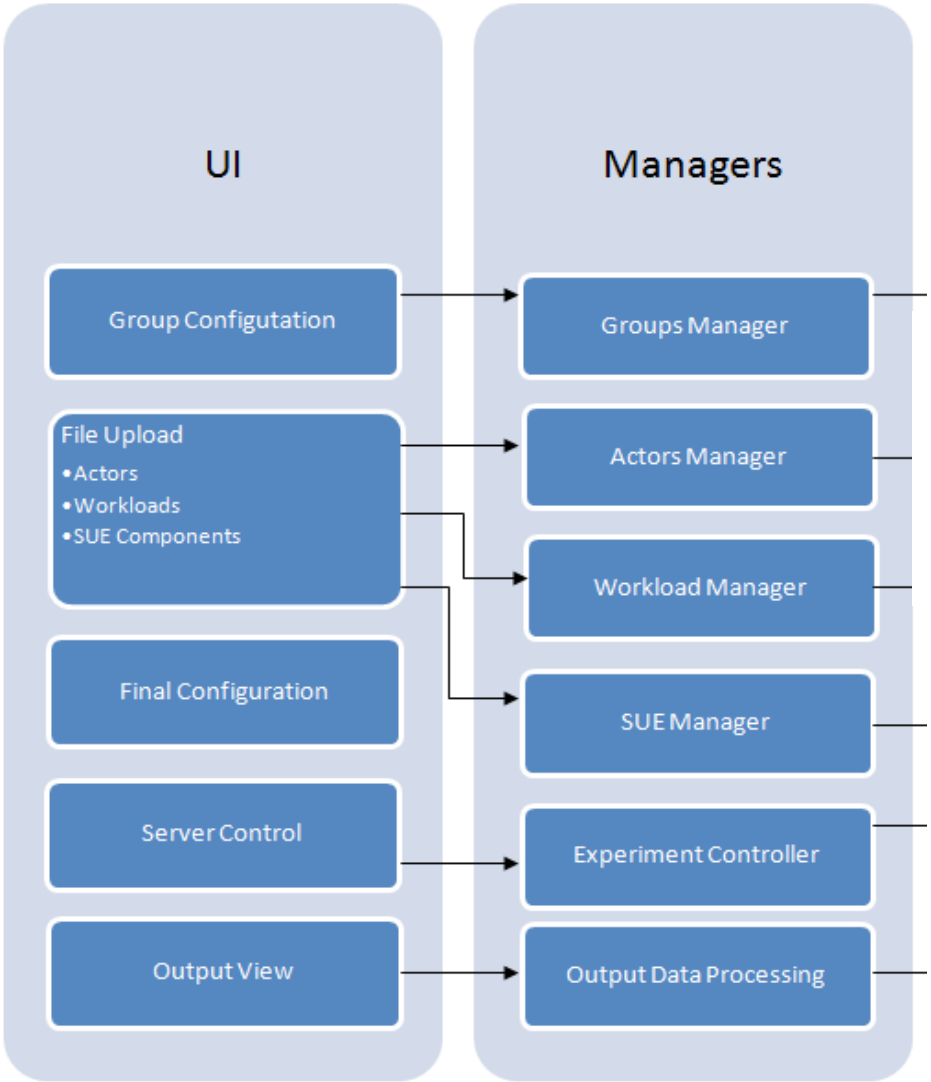
- Specify how many groups are required, their names, and their sizes
- Upload required files to the server
- Now that the server has all the files it needs, tell it how to set them up
- Load the Vazels Control Centre and begin the experiment



- Specify how many groups are required, their names, and their sizes
- Upload required files to the server
- Now that the server has all the files it needs, tell it how to set them up
- Load the Vazels Control Centre and begin the experiment
- Get results

The Architecture

Client-side – the ReefFront web interface



- Written entirely in Java
- Translated by GWT
- Closely follows MVC

The Groups Manager

- The most complex manager
- Groups configuration includes constructs of the Vazels system
- Keeps the system in sync, even after a refresh
- Abstracts the raw data on three levels:
  - SingleGroupManager
  - Group
  - Group Overlay

Abstracting Groups - SingleGroupManager

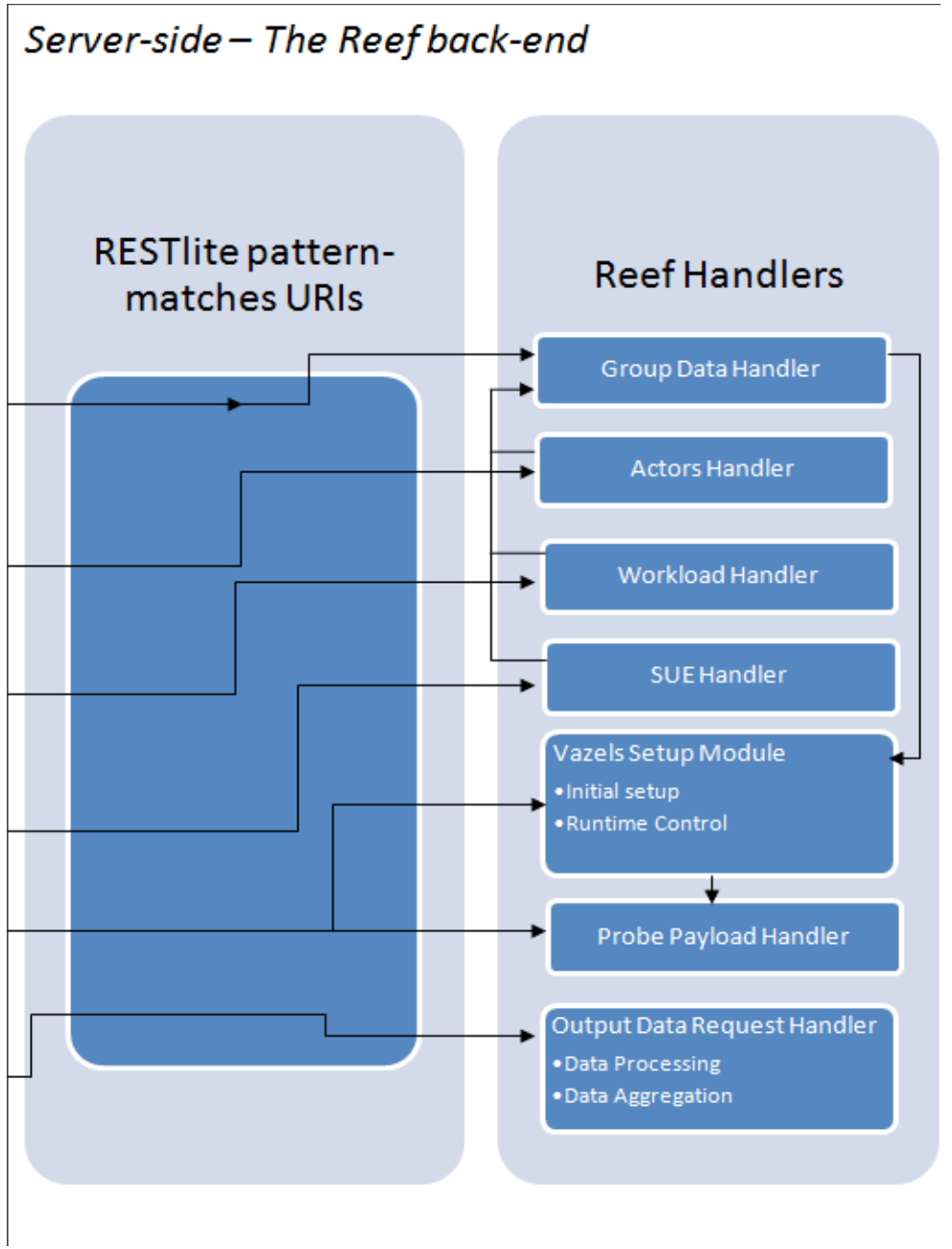
- As the GroupManager is to all groups, a SingleGroupManager is to a single group
- Wraps the Group data itself
- The only class that knows how to talk directly to the server about a specific group
- Data encapsulation

Abstracting Groups - The Group Class

- Contains the data of the Group
- Actually a wrapper around the GroupOverlay class
- Stores *all* the data on a group
- Contains no handling logic; relies on the SingleGroupManager to keep it up to date

The Architecture

- Python backend
- Lots of tiny WSGI apps
- And integration with Vazels



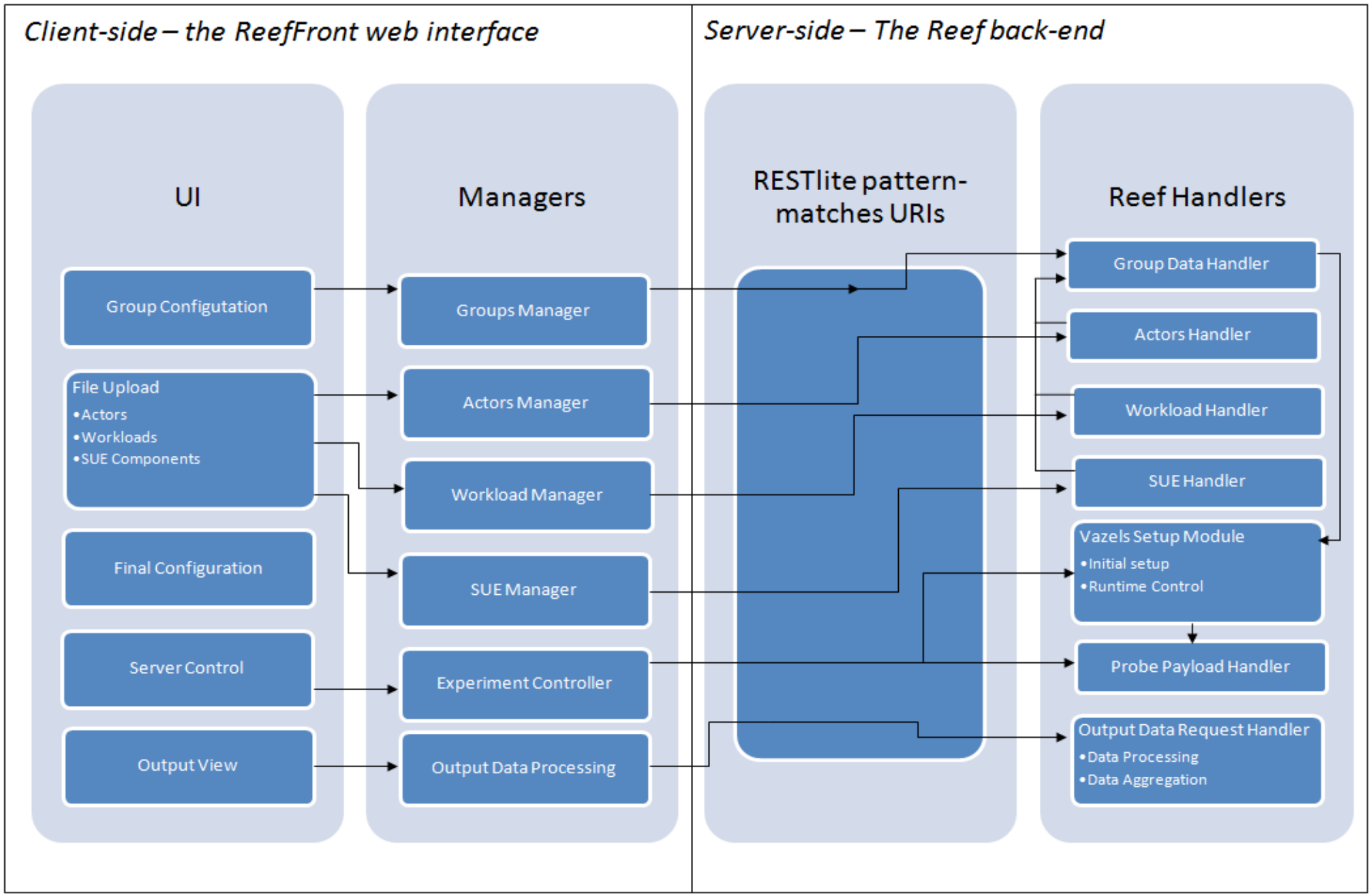
Getting the Vazels System Running

- Our system talks to the Vazels system on the command line
- We collect all the information together, and then issue the same commands as the user would normally need to
- No magic

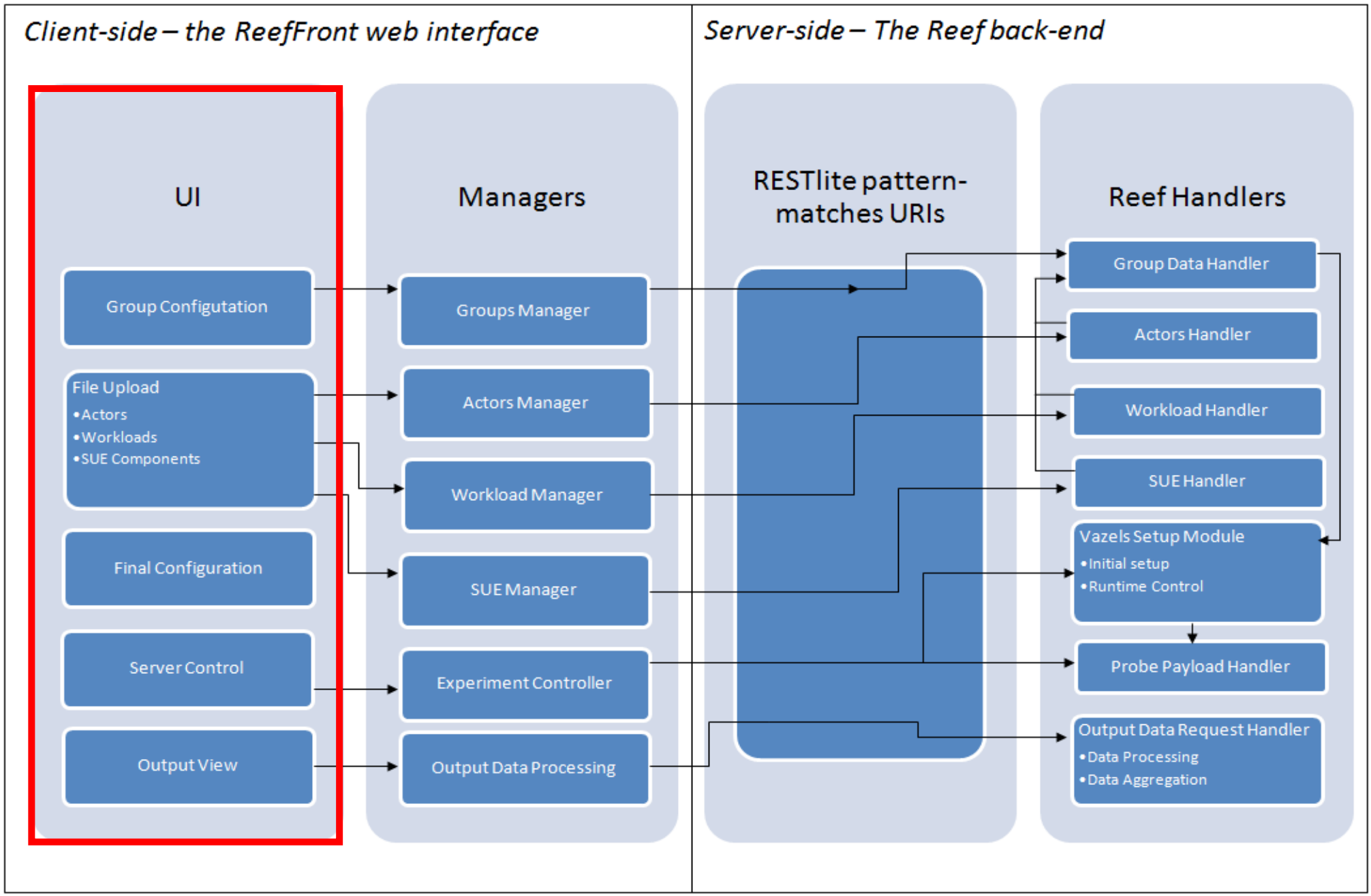
Getting the Vazels System Running

- Our system talks to the Vazels system on the command line
- We collect all the information together, and then issue the same commands as the user would normally need to
- No magic - but the command line is a brittle interface!

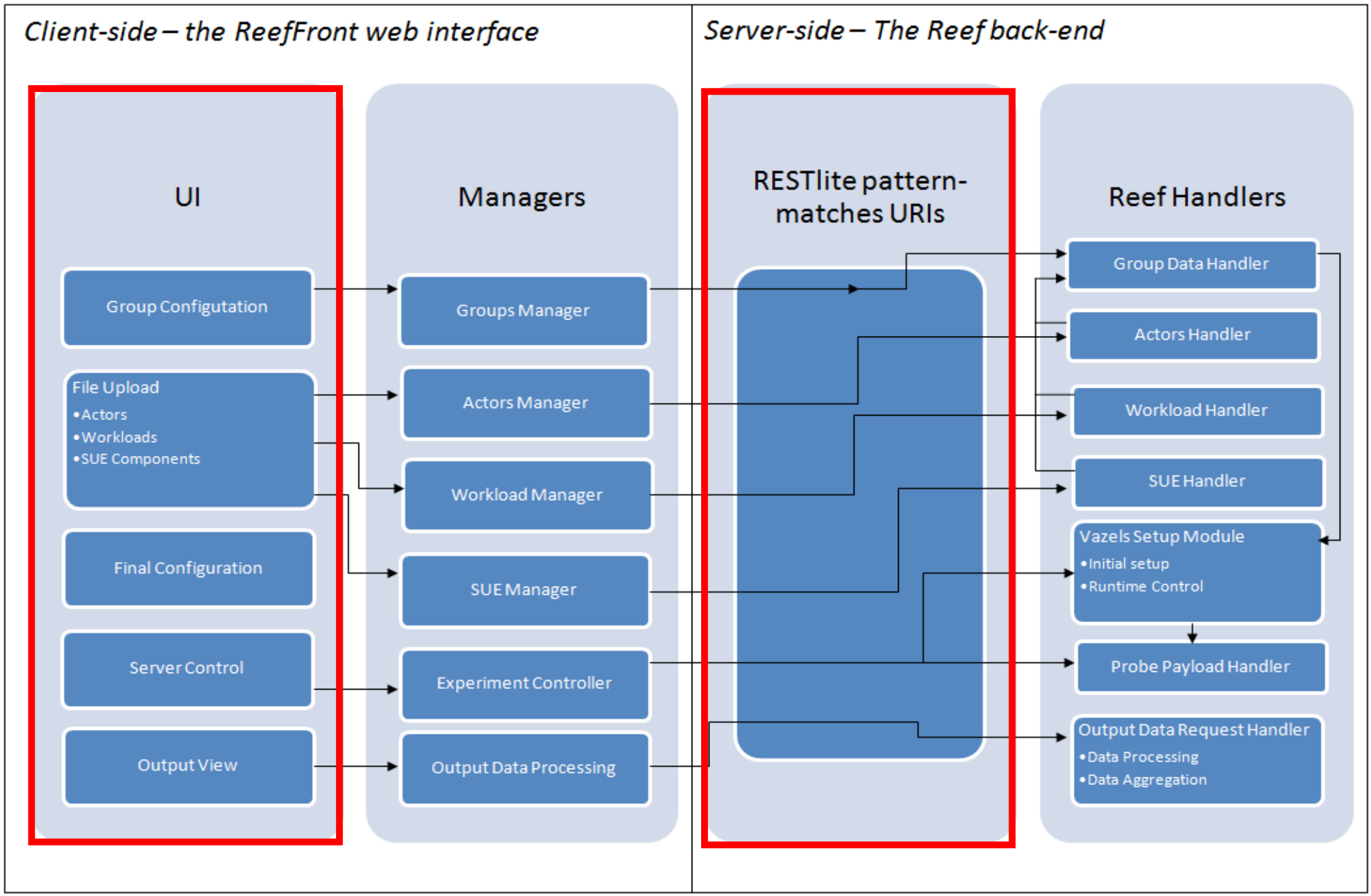
The Architecture - Testing



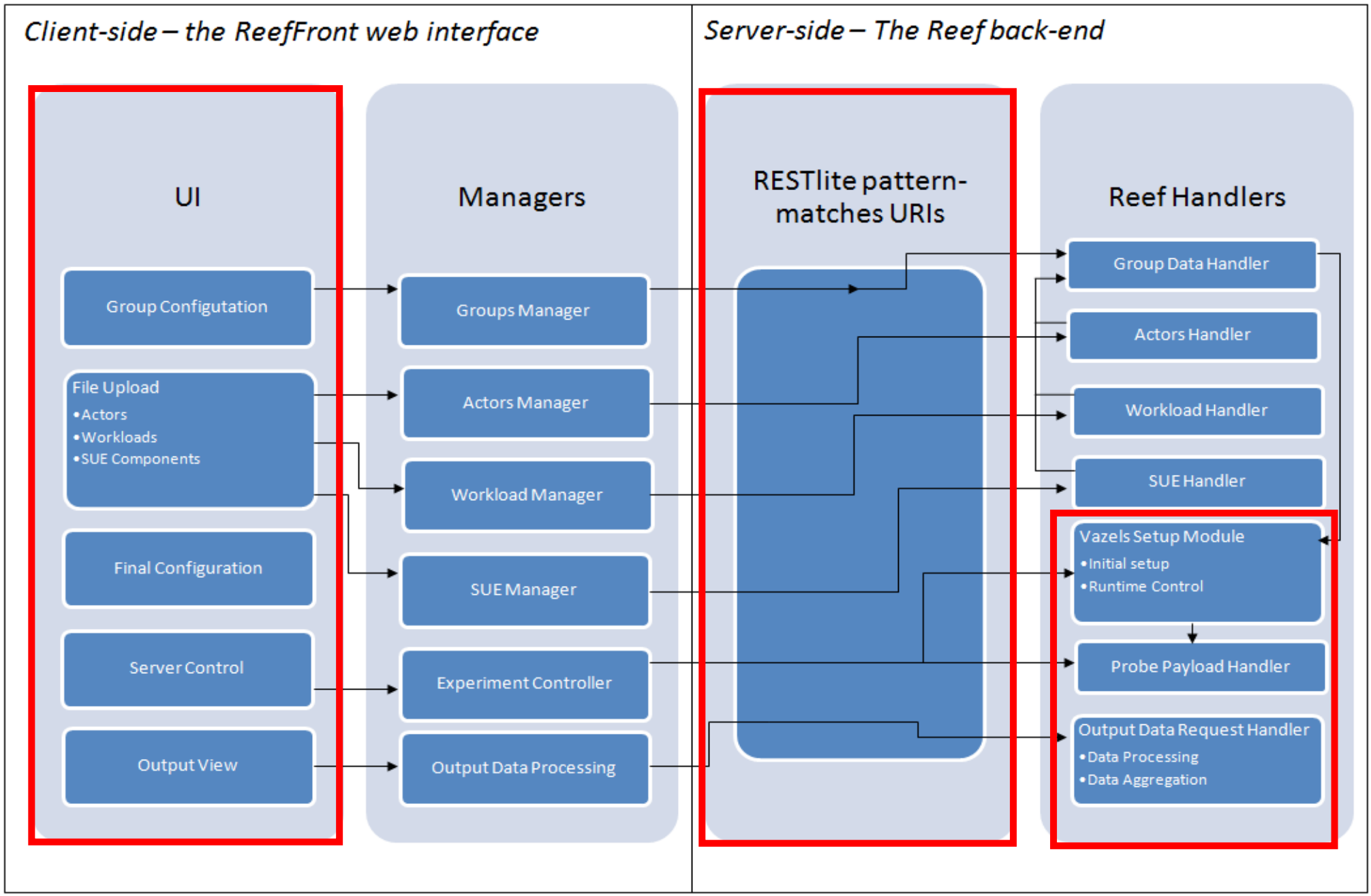
The Architecture - Testing



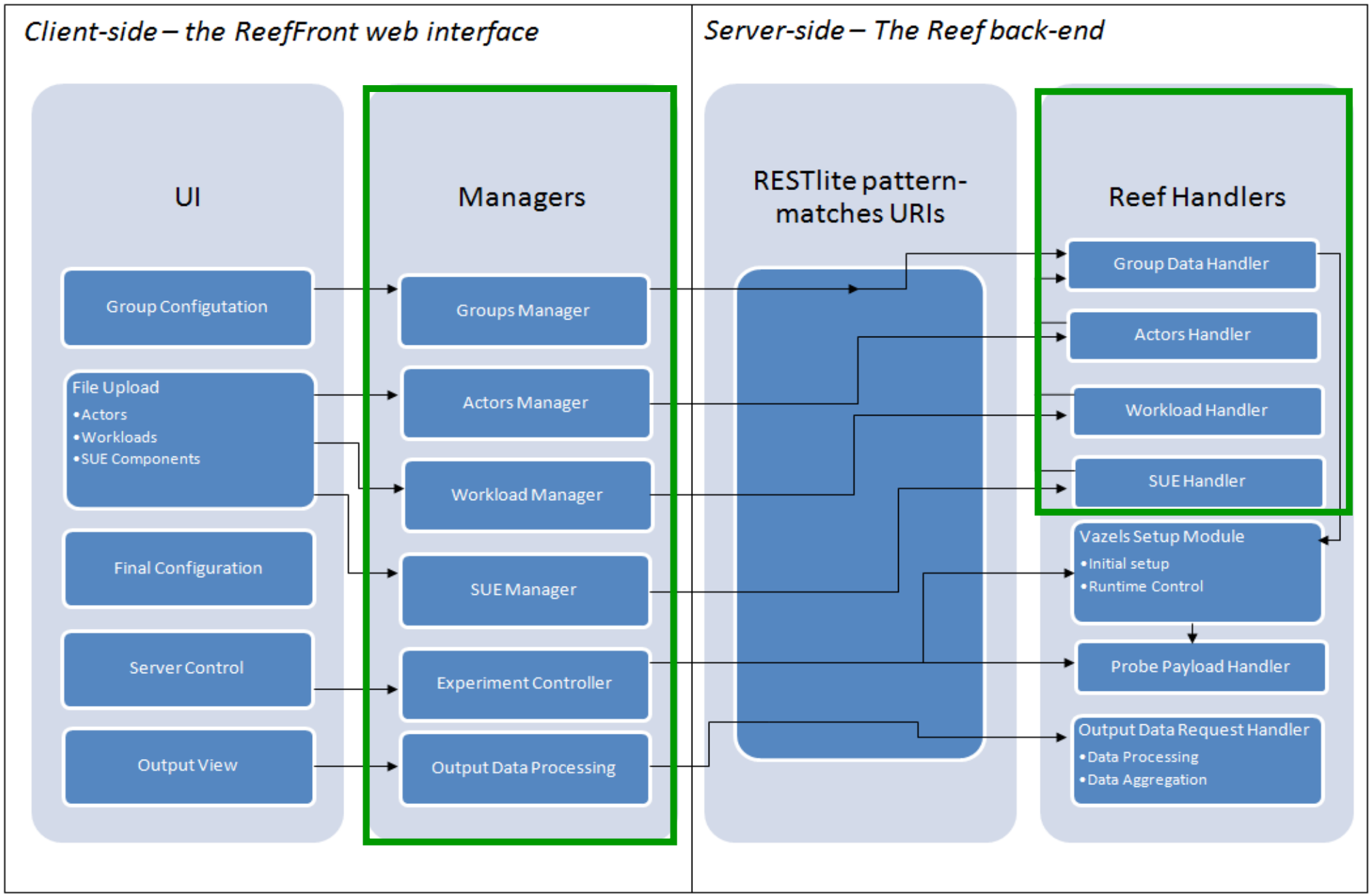
The Architecture - Testing



The Architecture - Testing



The Architecture - Testing



Developing without Unit Testing

- User stories
  - Key is in the detail
  - Be rigorous --or smth
  - Include *every* use case
- Documentation!
- Find other tools
  - cURL

Summary so far…

1. Motivation
   - How to, and why, test distributed systems?
   - What is Reef?
2. Reef Design Overview
   - Vazels overview
   - Client/server structure
3. Running Reef
4. Implementation & Architecture of Reef
   - Low-level Reef overview
   - Integration with Vazels

Summary so far…

1. Motivation
   - How to, and why, test distributed systems?
   - What is Reef?
2. Reef Design Overview
   - Vazels overview
   - Client/server structure
3. Running Reef
4. Implementation & Architecture of Reef
   - Low-level Reef overview
   - Integration with Vazels
5. Evaluation/Conclusion
   - Technical/collaboration challenges
   - Successes/shortcomings

Collaboration issues

- Different schedules ⟶ Doodle
- Different versions ⟶ Git
- Documentation ⟶ Wiki

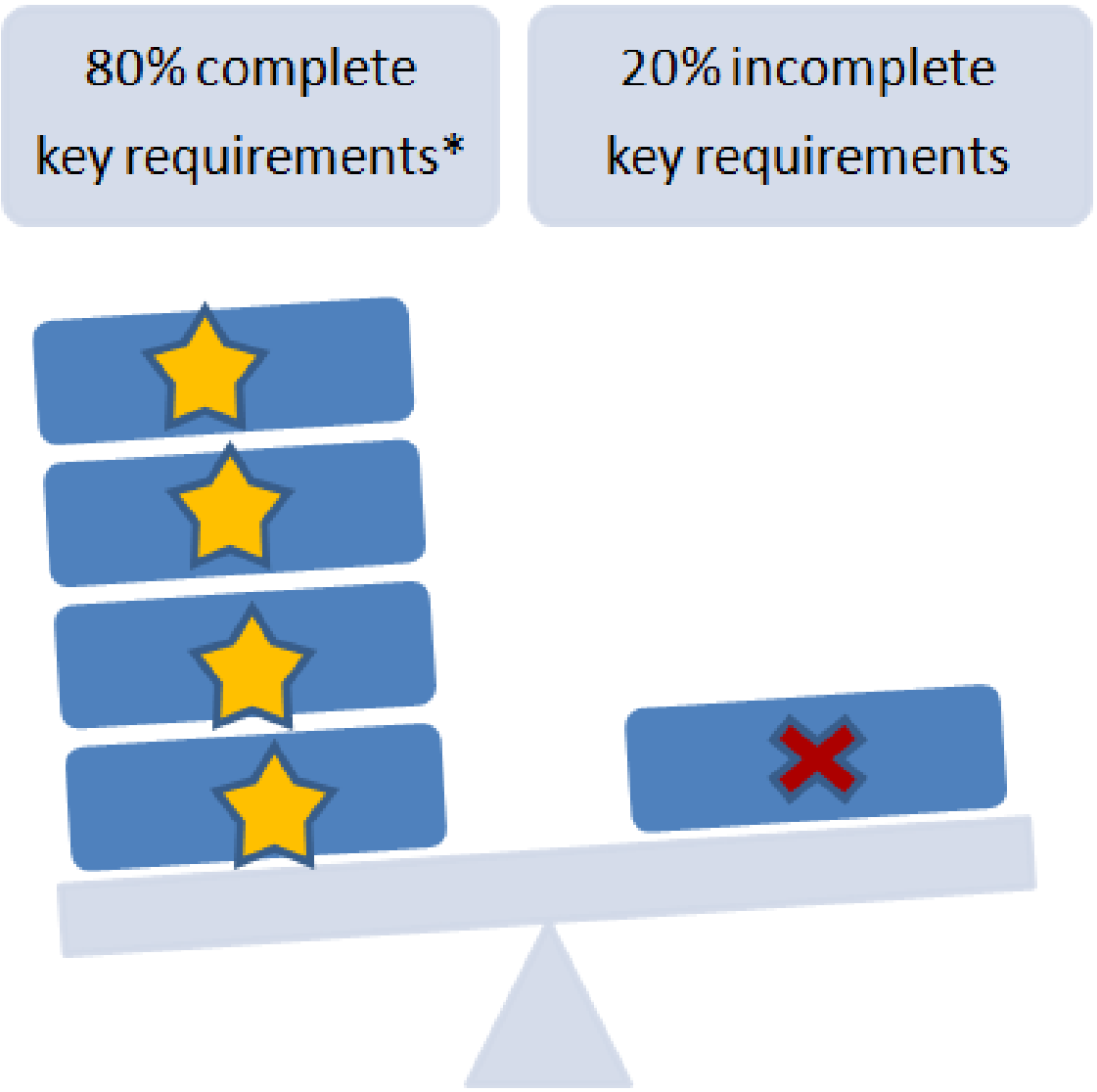
Technical challenges

- Running Vazels
- Operating System
- Unfamiliar tools

Achieved aims

Reef:

- ✓ is platform agnostic
- ✓ has remote and user-friendly access to Vazels
- ✓ saves configurations for a later use



* + 2 extensions partially addressed

Areas for improvement

- Display online/offline hosts
- Error handling
- Supported browser
- Data aware visualization

So why use Reef?

- ✓ usable front-end, delivered through a web browser, providing an easy to use distributed testing platform

So why use Reef?

So why use Reef?

The Happy Geek

What now?

Ongoing Potential

- user & developer documentation (wiki, javaDocs, PyDocs)
- project hosted on Github (https://github.com/jelford/reef)

a Distributed Testing UI

Acknowledgements:

- Prof. Alex Wolf - our project supervisor
- Angel Dzhigarov - author of the Vazels project