SPEAKER

THOMAS SUNDBERG

A TESTABLE ARCHITECTURE

AGILE & AUTOMATION DAYS | CONFERENCE & WORKSHOPS | 28-29 OCTOBER 2019 GDANSK, POLAND

A testable architecture

Thomas Sundberg

Think Code AB
http://www.thinkcode.se

@thomassundberg
thomas@thinkcode.se

What do I do?

Delivering software

- *Build automation*
- *Test automation*
- *Deployment automation*

Implementing software

- *Test-Driven Development, TDD*
- *Behaviour-Driven Development, BDD*
- *Clean and maintainable code*

Mentoring and teaching

- *Developers*
- *Testers*
- *Behaviour-Driven Development*
- *Test automation*

DevOps

- *What ever is needed*

Definition of done

Working software, in production.

Used by real users.

Goal

Understand what to build

Implement it with confidence

Make it available to our users

How is confidence created?

Trustworthy

adjective

adjective: **trustworthy**

able to be relied on as honest or truthful.
"leave a spare key with a trustworthy neighbour"

Similar: reliable, dependable, honest, full of integrity, worthy of trust, honourable, upright, principled, true, truthful, as good as one's word, ethical, virtuous, incorruptible, unimpeachable, above suspicion, responsible, sensible, level-headed, loyal, faithful, true-blue, staunch, steadfast, trusty, constant, unswerving, unwavering, tried and trusted, tried and true, safe, secure, sound, guaranteed, unfailing, foolproof, never-failing, reputable, on the level, sure-fire, straight-up

Opposite: untrustworthy, unreliable, shifty

How do you know that a program is trustworthy?

You test it

Is everything testable?

Not without a lot of problems

You need an architecture
that simplifies testing

A testable architecture

You need an architecture
that simplifies testing

A testable architecture

Today, I only talk about ***functional*** testing

Not

Capacity/Load

Penetration

There is always an **architecture**

Hard to test

Easier to test

Why are some architectures hard to test?

Cultural problem

Testing isn't a first class citizen*

Some see testing as SEP**

*A member of a class of individuals that receive fair treatment.

** Somebody elses problem

Douglas Adams,
Hitch hikers guide to the galaxy.
A trioligy in five parts

Why are some architectures easier to test?

Culture*

Tests is a first class citizen

Testing is a first class activity

Testing is your probelm, not SEP

Tests are used to drive the development

* No matter what they tell you, it's always a people problem.
Gerald Weinberg

What is a testable architecture?

It is an architecture that supports

The possibility to create confidence that the system works as intended

Short feedback loops

Low cost maintenance

The possibility to create confidence tha

Short feedback loops

Low cost maintenance

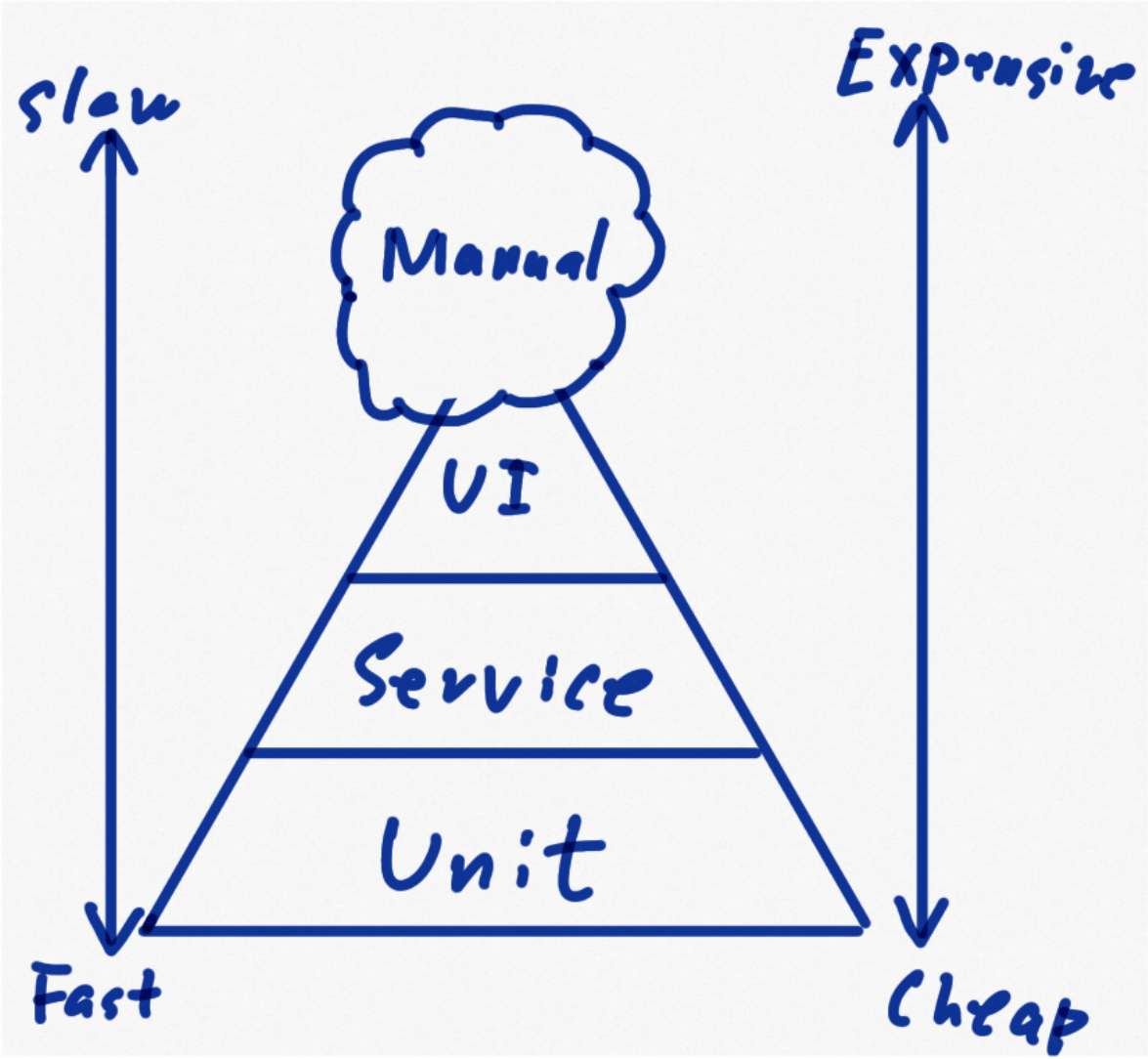
You can't have your cake and eat it

Confidence	✓	✗
Speed	✗	✓
Low cost	✗	✓

Can we get the best out of both worlds?

Is it possible to combine

- **Confidence**
- **Speed**
- **Low cost**



Cheap as in low maintenance

Take control over the world

Dependencies must be ***configurable***

- Inject the dependencies needed
- Usually through a constructor
  - I usually avoid magic

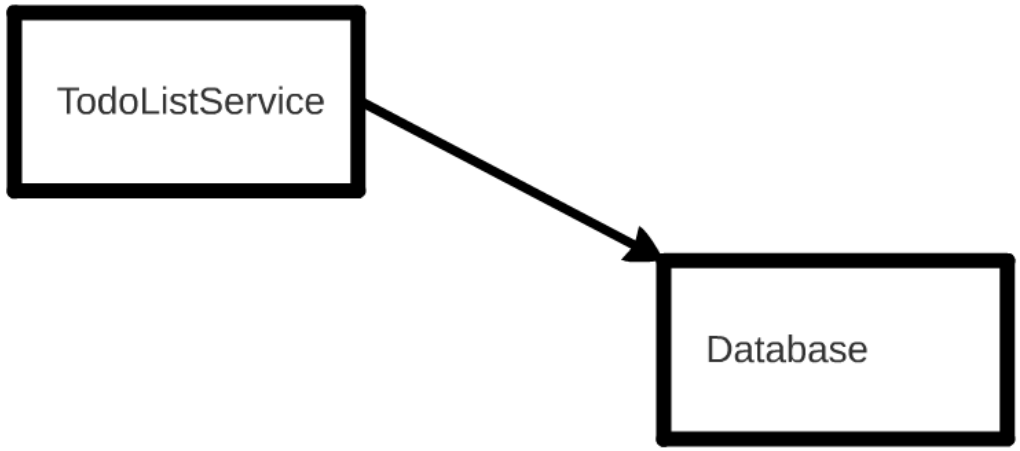
An example

A todo list

- Add tasks
- Get tasks

The one where Thomas remembers to buy cat food.

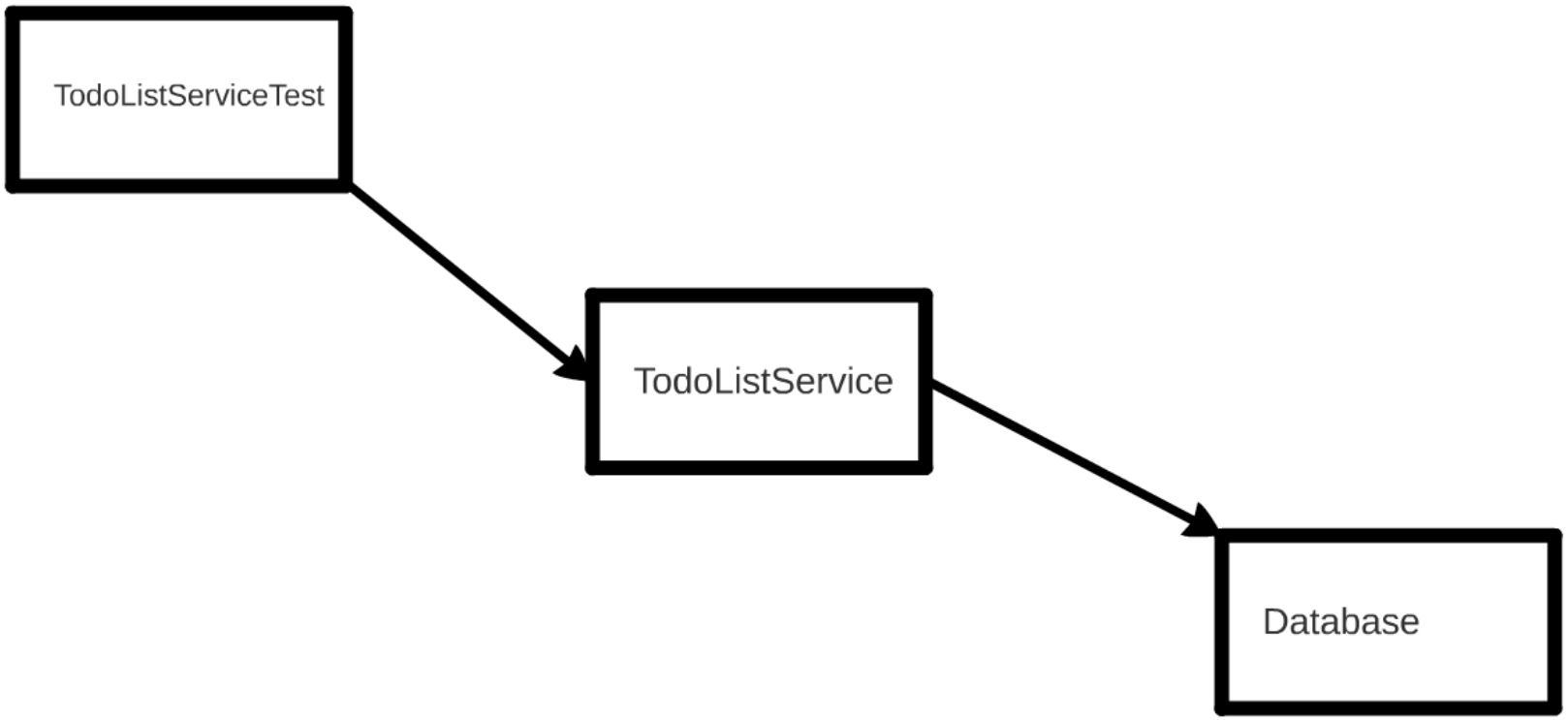
Stakeholders


TodoListService
Database

Configurable dependencies

Push the dependencies you need into the objects that need a collaborator

```
public TodoListService(Database database) {
    this.database = database;
}
```


TodoListServiceTest
TodoListService
Database

```
@Test
public void add_task() {
    Database database = mock(Database.class);
    TodoListService todoListService = new TodoListService(database);
    Owner owner = new Owner("Thomas");
    Task task = new Task("Buy food for the cat");

    todoListService.addTask(owner, task);

    verify(database).addTask(owner, task);
}


@Test
public void get_tasks() {
    Database database = mock(Database.class);
    TodoListService todoListService = new TodoListService(database);
    Owner owner = new Owner("Thomas");

    todoListService.getTasks(owner);

    verify(database).getTasks(owner);
}
```

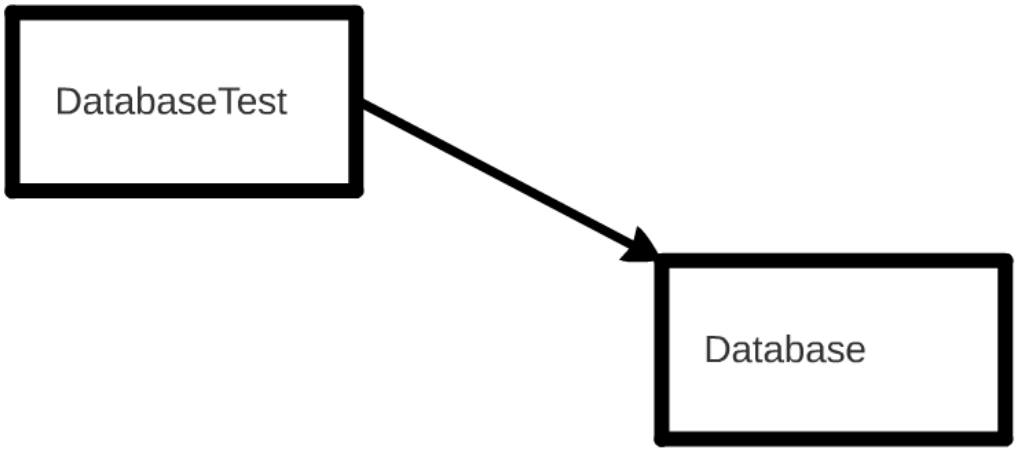
```
@Test
public void add_task() {
    Database database = mock(Database.class);
    TodoListService todoListService = new TodoListService(database)
    Owner owner = new Owner("Thomas");
    Task task = new Task("Buy food for the cat");

    todoListService.addTask(owner, task);

    verify(database).addTask(owner, task);
}


@Test
public void get_tasks() {
```

```
    verify(database).addTask(owner, task);
}


@Test
public void get_tasks() {
    Database database = mock(Database.class);
    TodoListService todoListService = new TodoListService(database)
    Owner owner = new Owner("Thomas");

    todoListService.getTasks(owner);

    verify(database).getTasks(owner);
}
```

Confidence 

Speed 

Low cost 


DatabaseTest
Database

The database is *slow*

Contract testing to the rescue

Verify the **real** database and a **fake** database with the same test suite

If they are **equal** from a functional perspective, I **trust** the fake

Plan:

Implement the database as an interface

Create a proper implementation - **SqlDatabase**

Create a fake implementation - **InMemoryDatabase**

Confidence	✔	✖
Speed	✖	✔
Low cost	✖	✔

```
@Test
public void should_add_a_task() {
    Owner owner = new Owner("Malin");
    Task task = new Task("Buy Minutuu");
    database.addTask(owner, task);

    List<Task> actual = database.getTasks(owner);

    assertThat(actual).containsExactly(task);
}
```

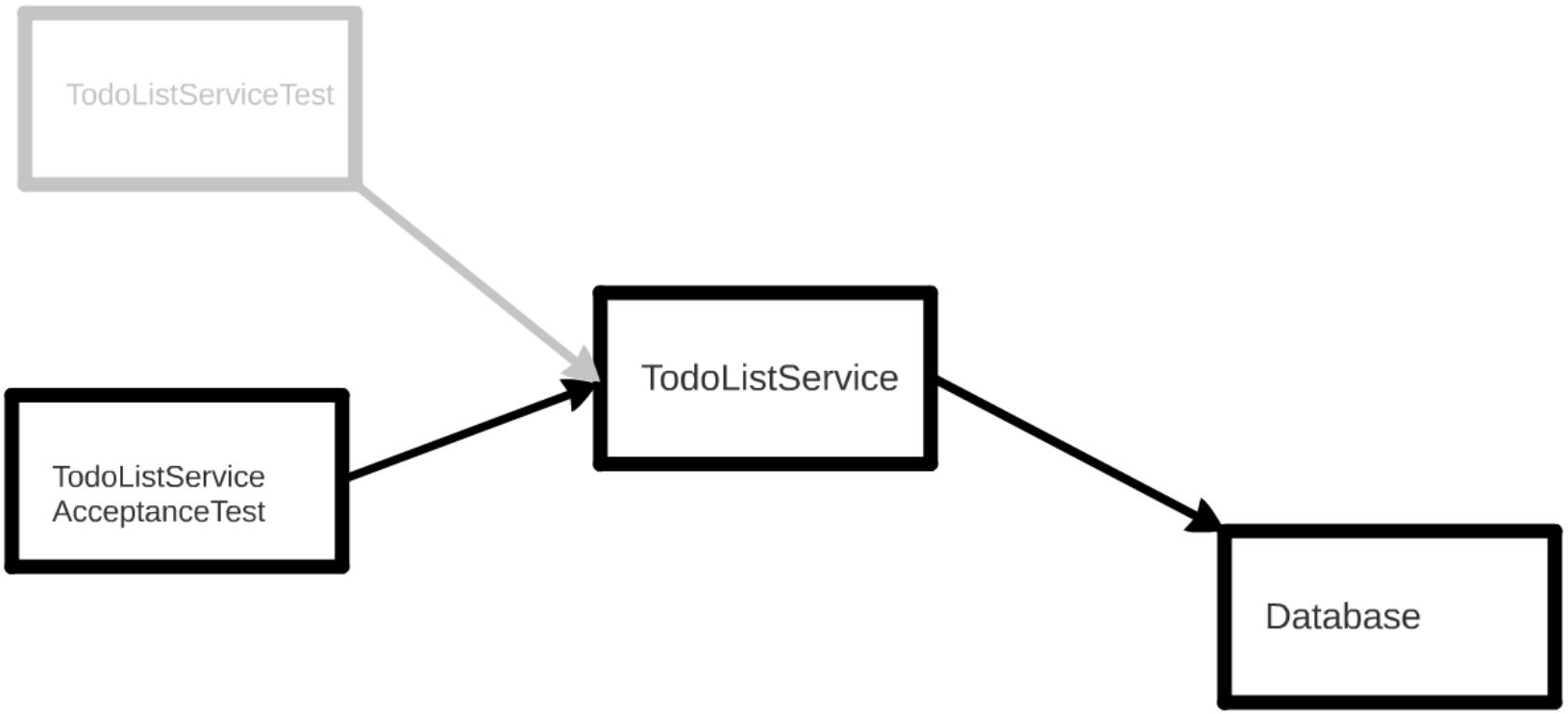
```
public abstract class DatabaseTests {
    Database database;

    @Test
    public void should_add_a_task() {
        Owner owner = new Owner("Malin");
        Task task = new Task("Buy Minutuu");
        database.addTask(owner, task);

        List<Task> actual = database.getTasks(owner);

        assertThat(actual).containsExactly(task);
    }
}
```

Inheritance allows me to verify the real thing and the fake

```
public class SqlDatabaseTest extends DatabaseTests {
    public SqlDatabaseTest() {
        database = new SqlDatabase();
    }
}
```

```
public class InMemoryDatabaseTest extends DatabaseTests {
    public InMemoryDatabaseTest() {
        database = new InMemoryDatabase();
    }
}
```

Confidence	✔	✖
Speed	✖	✔
Low cost	✖	✔


TodoListServiceTest
TodoListService
AcceptanceTest
TodoListService
Database

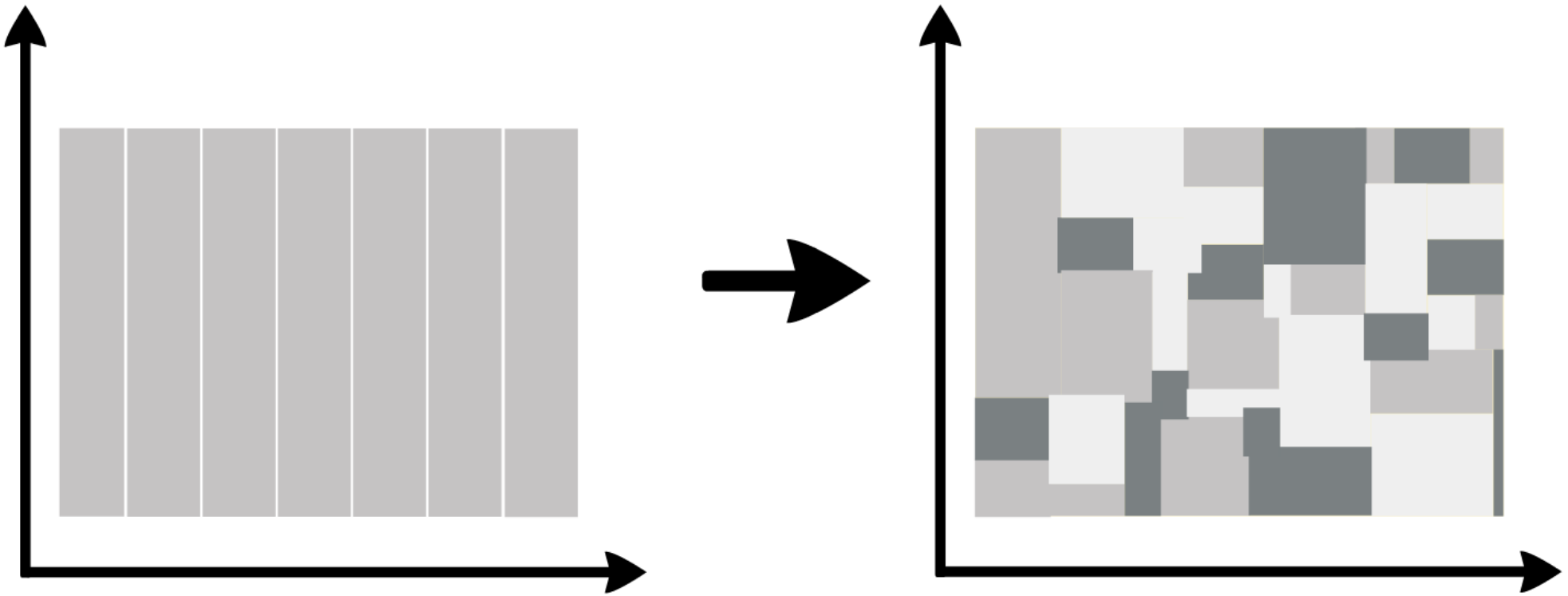
```
public class TodoListAcceptanceTest {

    @Test
    public void should_remember_to_buy_cat_food() {
        Database database = new InMemoryDatabase();
        TodoListService todoListService = new TodoListService(database);
        Owner owner = new Owner("Thomas");
        Task task = new Task("Buy food for the cat");

        todoListService.addTask(owner, task);

        List<Task> actual = todoListService.getTasks(owner);

        assertThat(actual).containsExactly(task);
    }
}
```

Confidence ✔

Speed ✔

Low cost ✔

Is this idea thrustworty?



Can we trust a patch work?

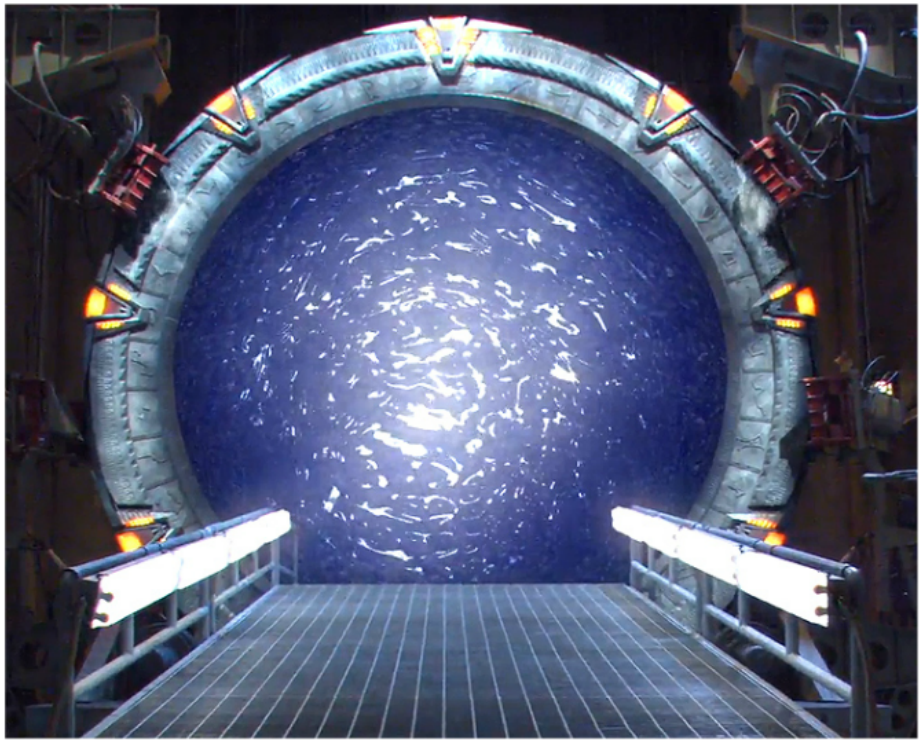
Most tests are small and only covers a tiny part...

I trust it and wouldn't try to sell it to you if I didn't.

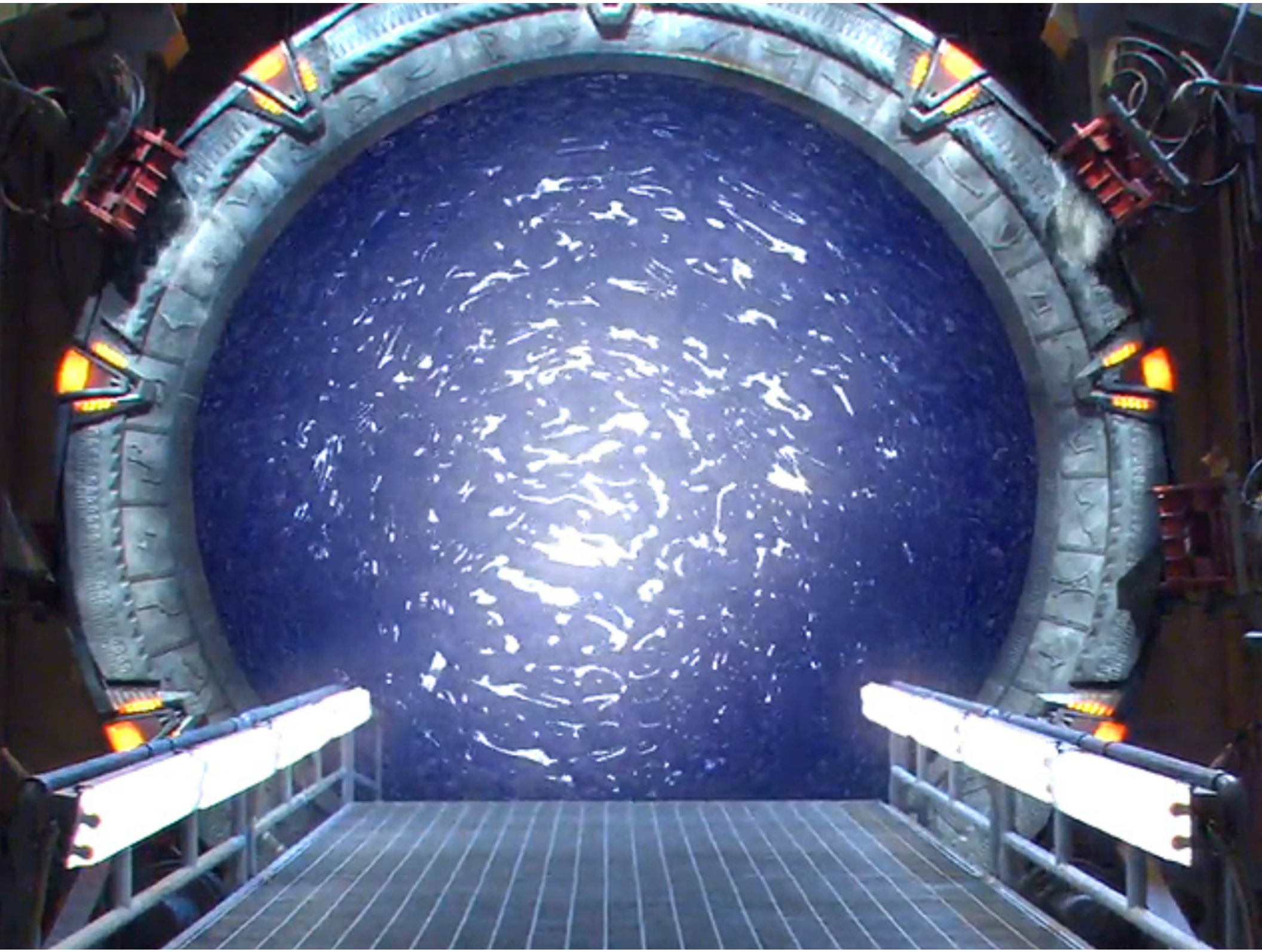
Talk is cheap, show me the code!

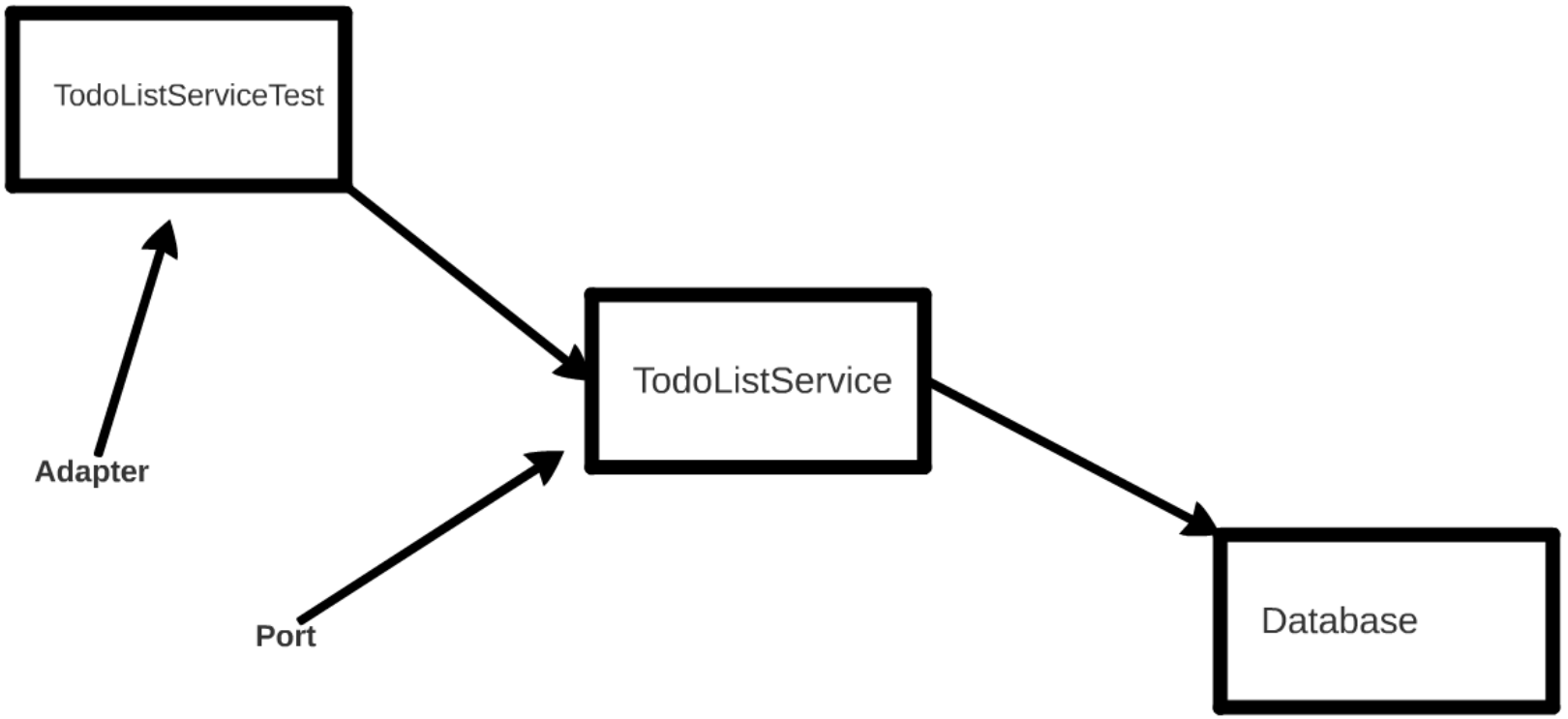
Ports & Adapters

Ports & Adapters

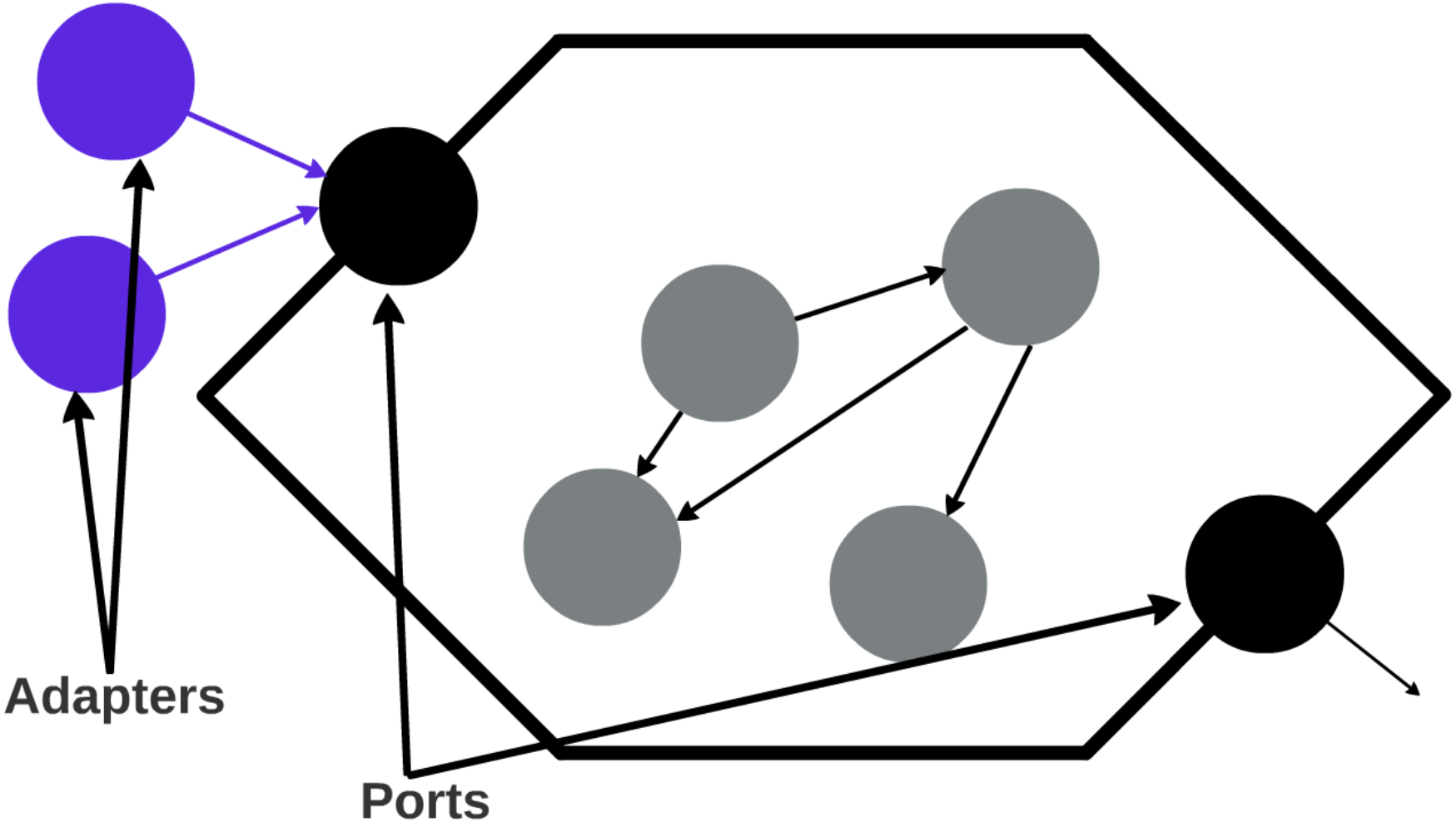


The port at
Stargate sg1

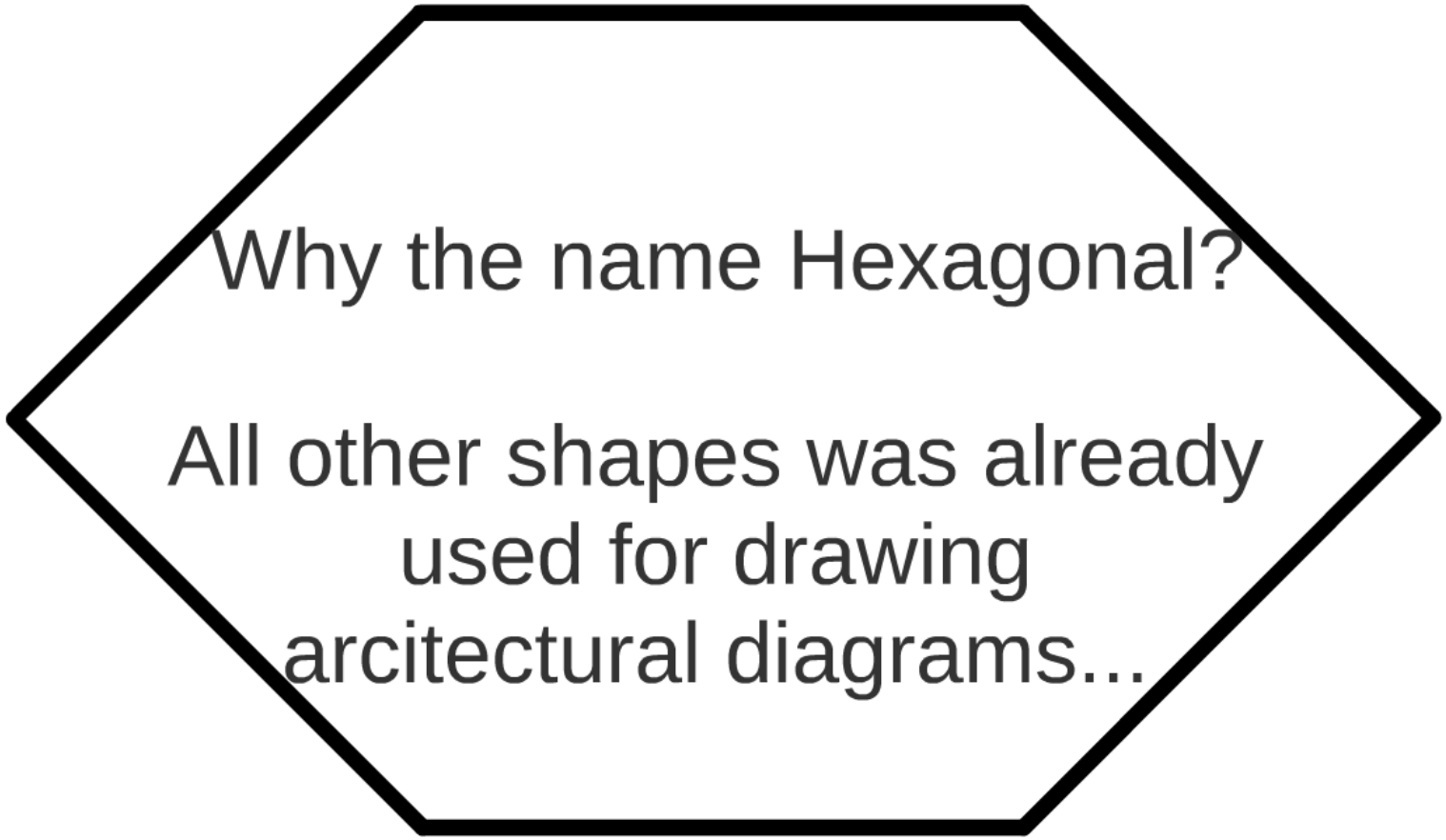



TodoListServiceTest
TodoListService
Database
Adapter
Port

Hexagonal architecture



Hexagonal architecture



Hexagonal architecture

With configurable dependencies

Conclusion

Maximize the work not done*

Don’t automate something that didn’t need doing in the first place

Take control over the world

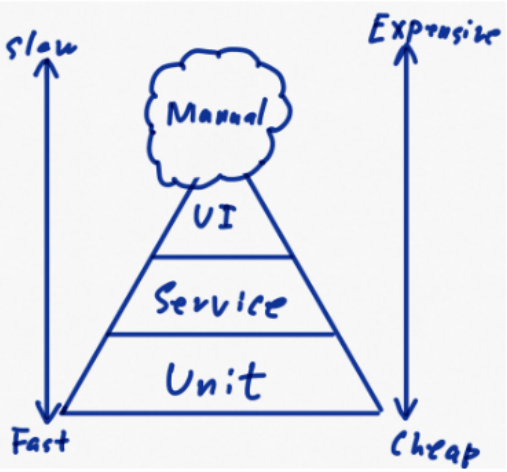
Use configurable dependences

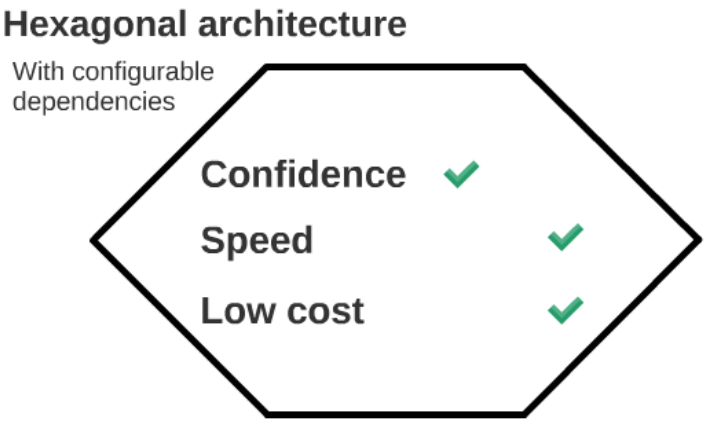
Focus on

Confidence

Speed

Low cost





*Laziness wins in the end - Viktor Olsson

A testable architecture

Thomas Sundberg

Think Code AB
http://www.thinkcode.se

@thomassundberg
thomas@thinkcode.se

The example:
https://github.com/tsundberg/a-testable-architecture

THANK YOU FOR THE ATTENTION

DID YOU LIKE THIS PRESENTATION?

RATE IT IN A&A DAYS MOBILE APP!