AARHUS UNIVERSITET

Microservices and DevOps

DevOps and Container Technology

SkyCave

Henrik Bærbak Christensen

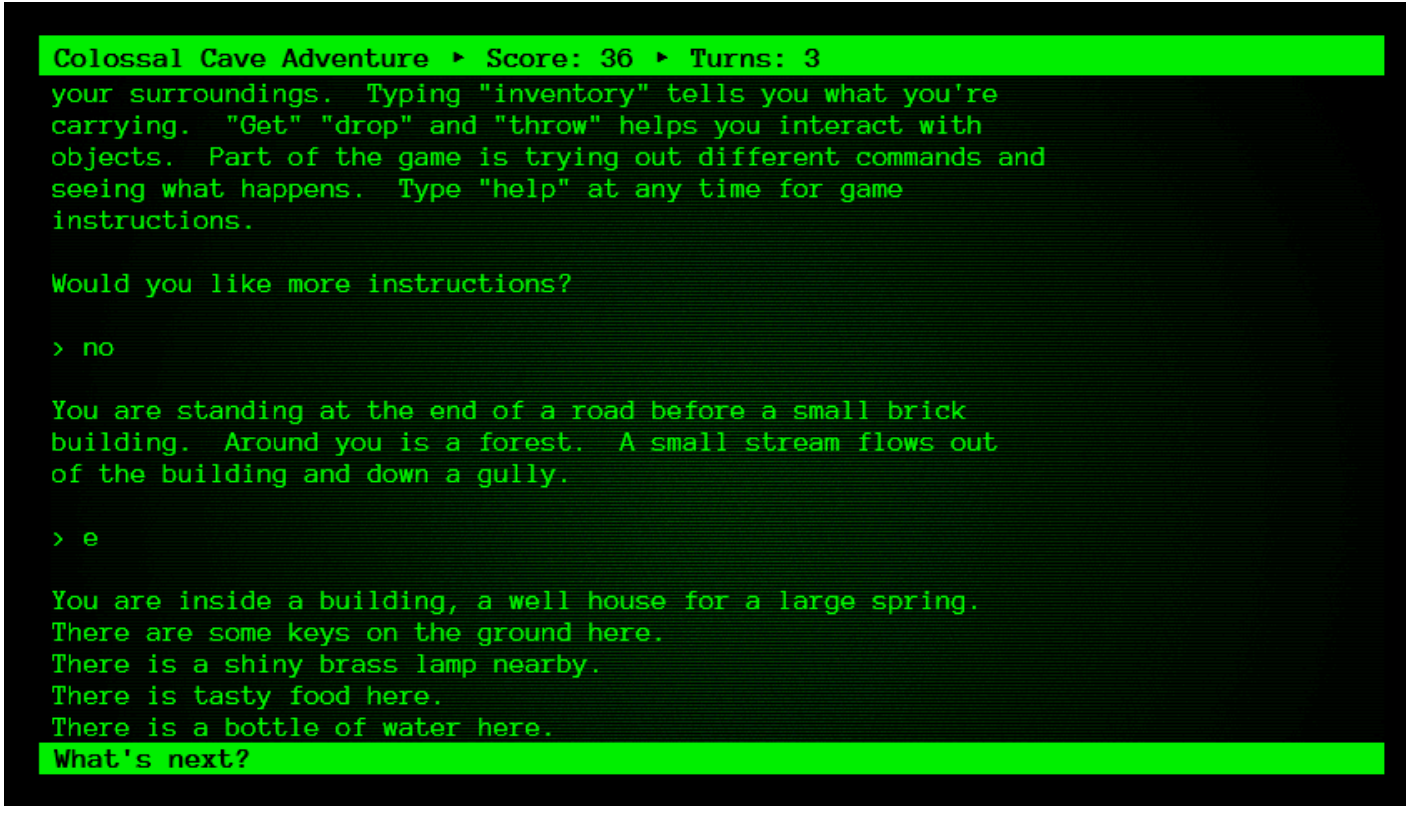
AARHUS UNIVERSITET

Case Study: SkyCave

- SkyCave is a
  - Massive multi-user online, exploration and creation experience with a bit of social networking
  - … with a *horrible* user interface
  - … and quite a lot of disregards for security
    - Hum hum, this year we tack a bit of security on it, though…
- Inspired by the very first ‘interactive fiction’ game for a computer: Colossal Cave Adventure
  - Will Crowder, 1972
    - I played my first game in 1986

AARHUS UNIVERSITET

The History



AARHUS UNIVERSITET

SkyCave is…

- … changed as it …
  - Removes all adventure aspects
    - ‘get lamp’ ‘throw axe’ ‘use key’
  - Allows modifications to cave which is a (x,y,z) lattice
    - ‘dig n You are in a maze with twisty passages’
      - Will create a new room, north of this, with the given description
  - **Massive online / distributed system (MMO)**
    - Not one but 10, 1000(!), 1.000.000 (!!!) players
  - **… through horizontal scalable client-server architecture**
  - Single sign-on
    - Once registered, each player can access every cave

AARHUS UNIVERSITET

SkyCave is…

- … changed as it …
  - Has some weird features with a learning focus
    - ‘quote’ report a famous quote
  - Is highly (re)configurable
    - To support automated testing using *test doubles*
    - … and therefore support *test-driven development*
    - To support *incremental architectural work*
      - HashMap → MongoDB
      - Local call → HTTP → RabbitMQ
    - To support *architectural testing*
      - *Saboteur* implementation of central services
    - To support *automated evaluation by me*

AARHUS UNIVERSITET

And Used to…

- Learn about
  - Broker architecture
    - Implement missing methods in Player
  - HTTP/Restish architecture
    - Connect server to subscription, quote services
      - (and later refactor the whole architecture into microservices)
  - NoSQL databases
    - Implement CaveStorage in MongoDB, Redis, Memcached, or …
  - Container Tech
    - Pack server into containers, deploy on swarm, …
  - CI and CD
    - Automate end-2-end testing and stuff
  - MS architecture
    - Refactor a Monolith into a full MS arch doing DevOps with your peers

AARHUS UNIVERSITET

And…

- As one former student put it
  - *Wildly over-engineered architecture ☺*
- SkyCave is a highly reconfigurable framework
  - Lots of ‘hotspots’, lots of roles, complex dep injection
- SkyCave has history
  - A few ‘fixme’s still around, a few ‘unused ideas’, etc.
- And as always
  - *There may be dragons*
    - *Undiscovered bugs, unhealthy naming, old-school java, legacy, …*

AARHUS UNIVERSITET

Software Architecture Sidestep

AARHUS UNIVERSITET

The 3+1 Viewpoints

- +1: Architectural requirements
- Deployment Viewpoint
  - Focus: What physical/virtual nodes are involved, what software is running on each? **The physical view**
- Component Connector Viewpoint
  - Focus: What processes/objects are executing, how does data and control flow between them? **The runtime view**
- Module Viewpoint
  - Focus: What compilation units are there, what classes, how are they dependent upon each other? **The static view**

AARHUS UNIVERSITET

Deployment and Execution View

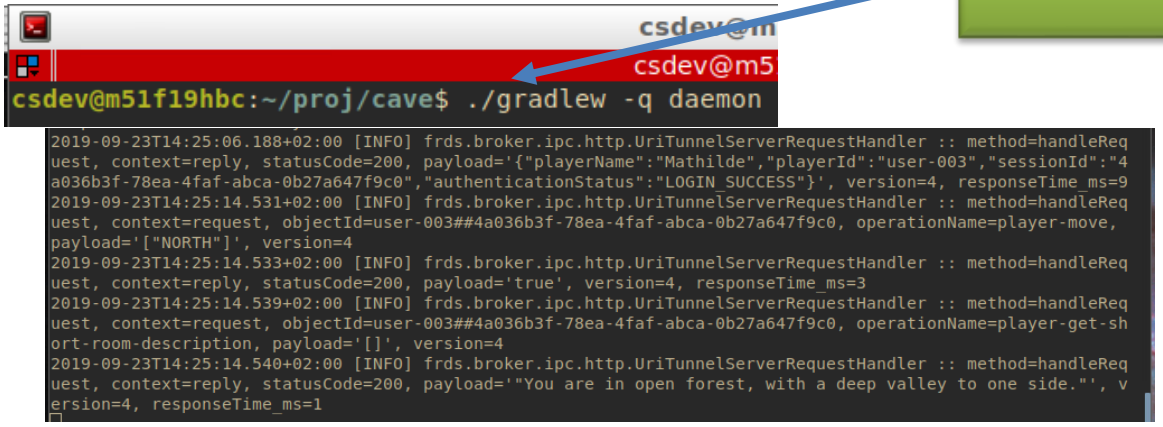
Demo!

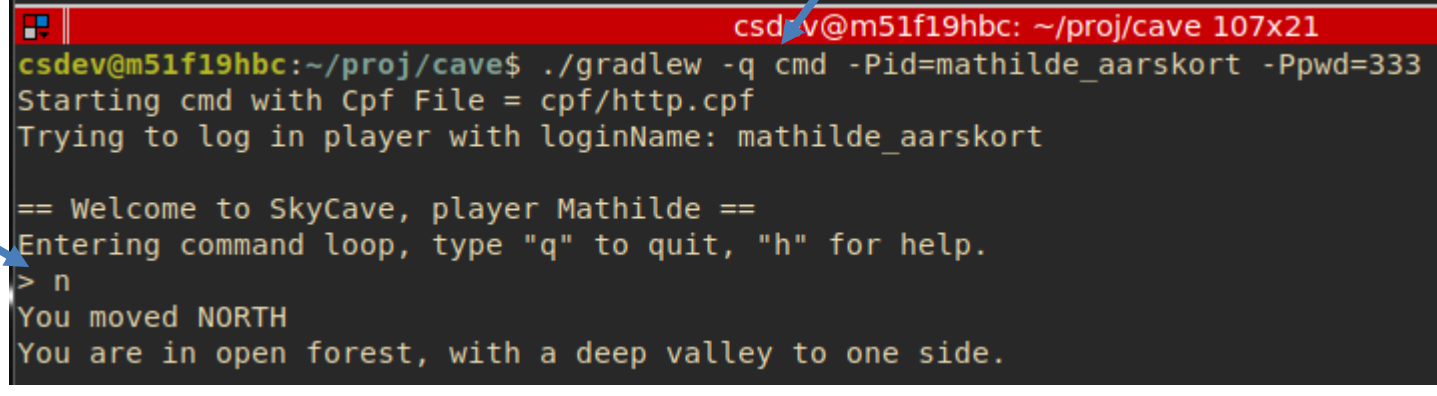
Demo: Local to Mxx

Start ‘daemon’

Start ‘cmd’

Interact.
Type ‘h’ for help



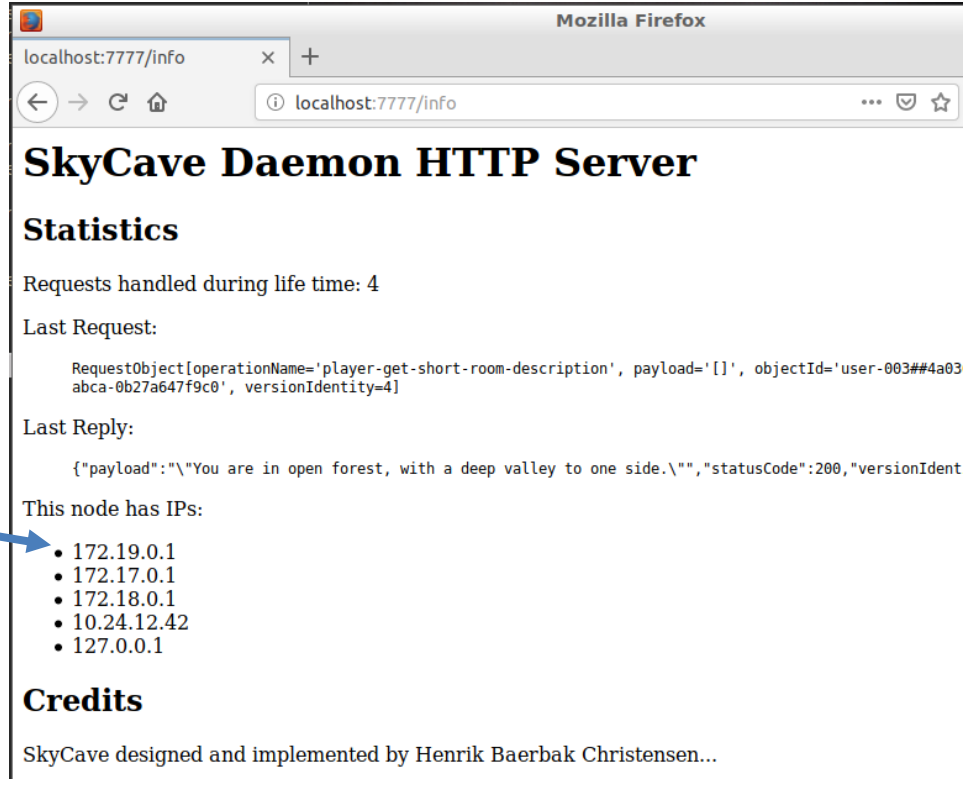


AARHUS UNIVERSITET

Rudimentary Statistics

- Client-server interaction using HTTP on port 7777
  - Path /info provides some stats to review

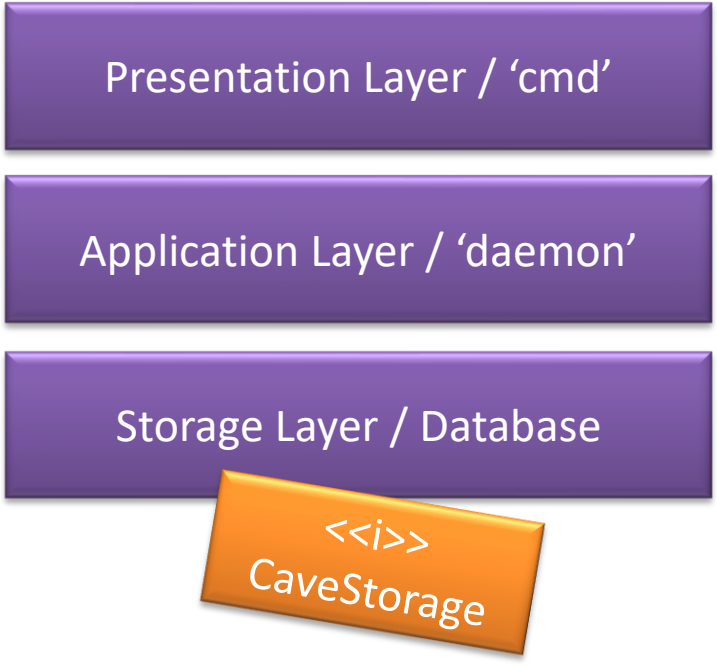
Relevant later, when we scale horizontally...



AARHUS UNIVERSITET

Three Tier Architecture

- Three Tier Architecture



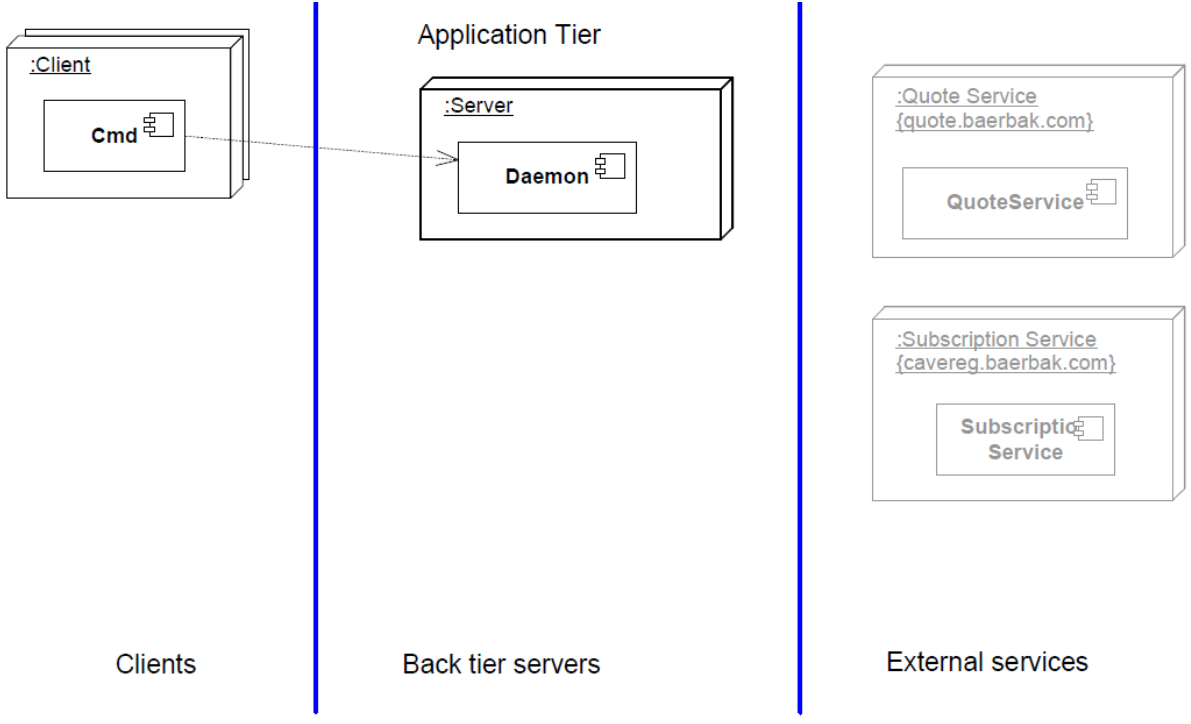
A microservice architecture refactoring is a task in the next course ☺

Fowler: Monolith First Pattern ☺

AARHUS UNIVERSITET

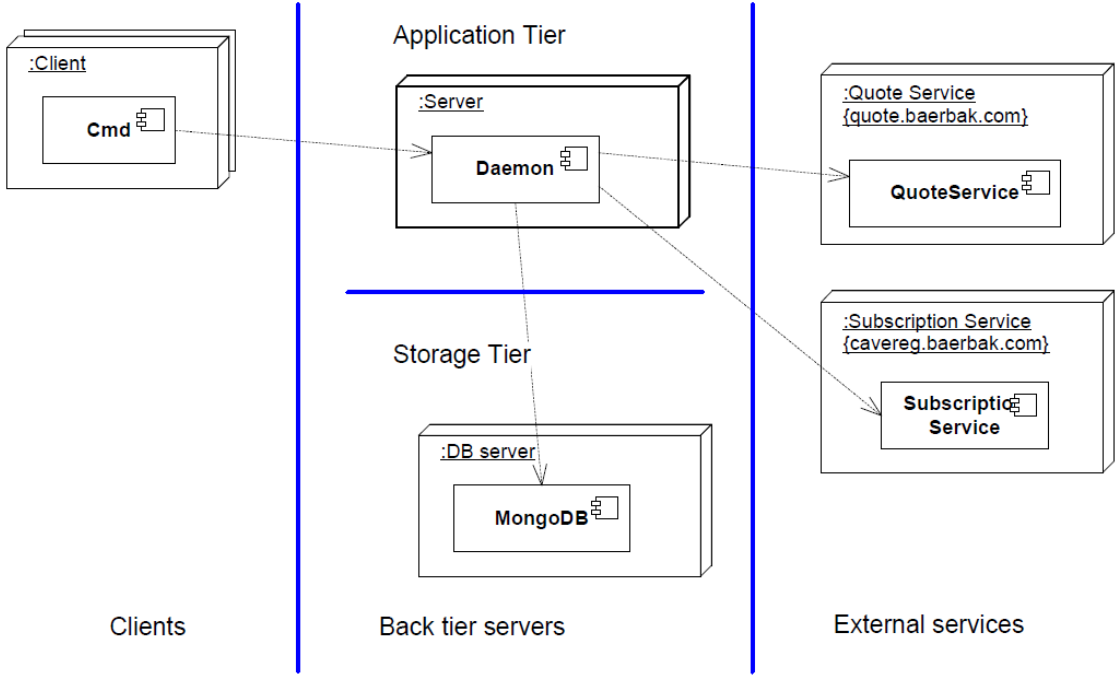
Deployment Viewpoint

- The *handed out* SkyCave (‘http.cpf’)



Deployment Viewpoint

- The *further down the road* SkyCave



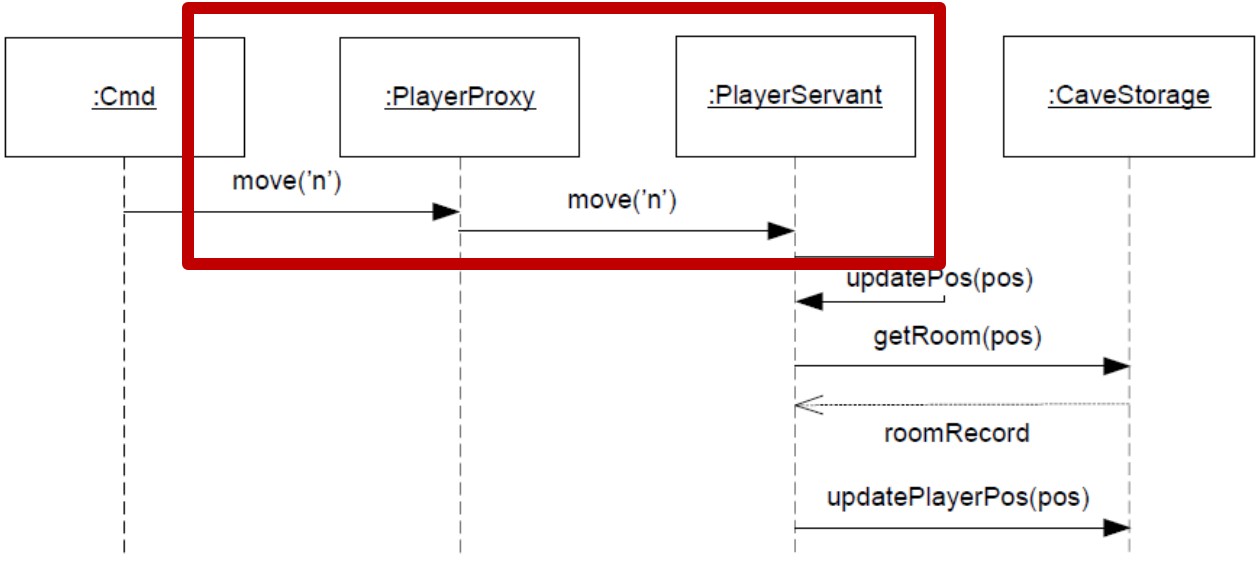
AARHUS UNIVERSITET

C&C View

The runtime view

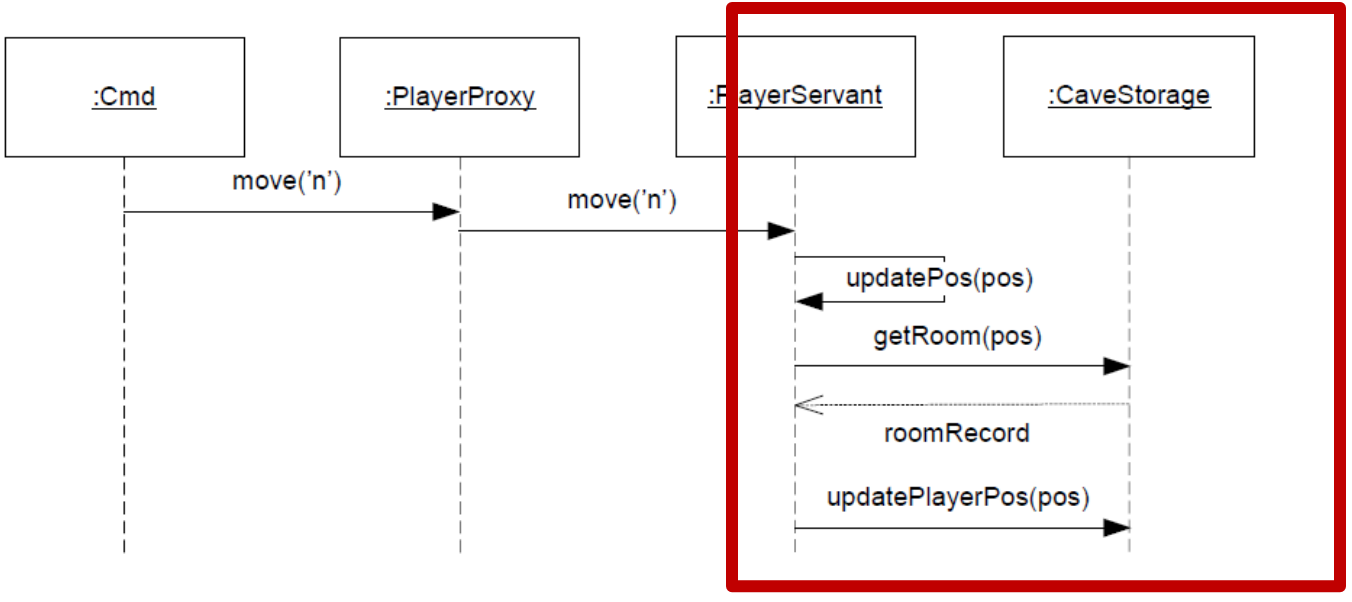
Synchroneous Method Call

- Client-server (cmd-daemon) interaction is just Broker based.



Server Side

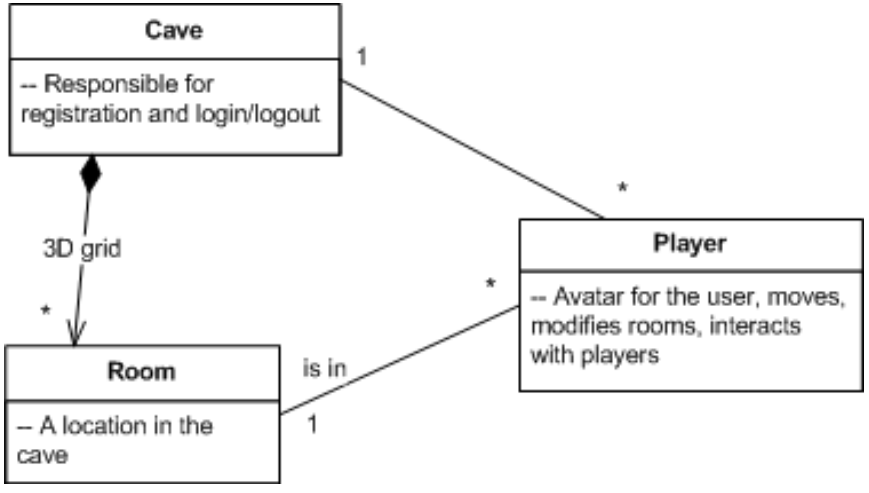
- Server side handling – just interact with persistent storage…



AARHUS UNIVERSITET

Module View

Domain Model

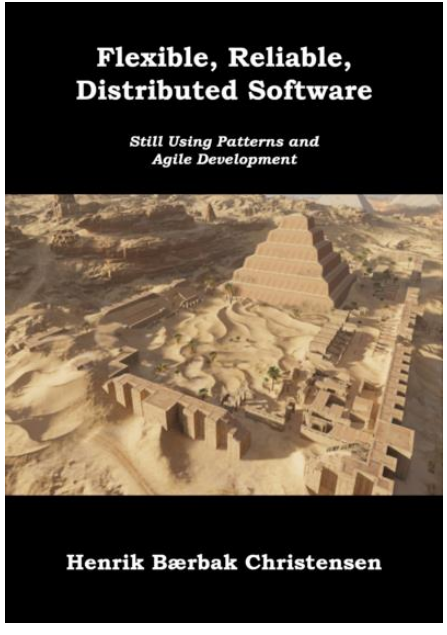


- *A very simple domain model !*
  - (And a good example of the fact that domain modeling helps *very* little in designing a strong architecture!)

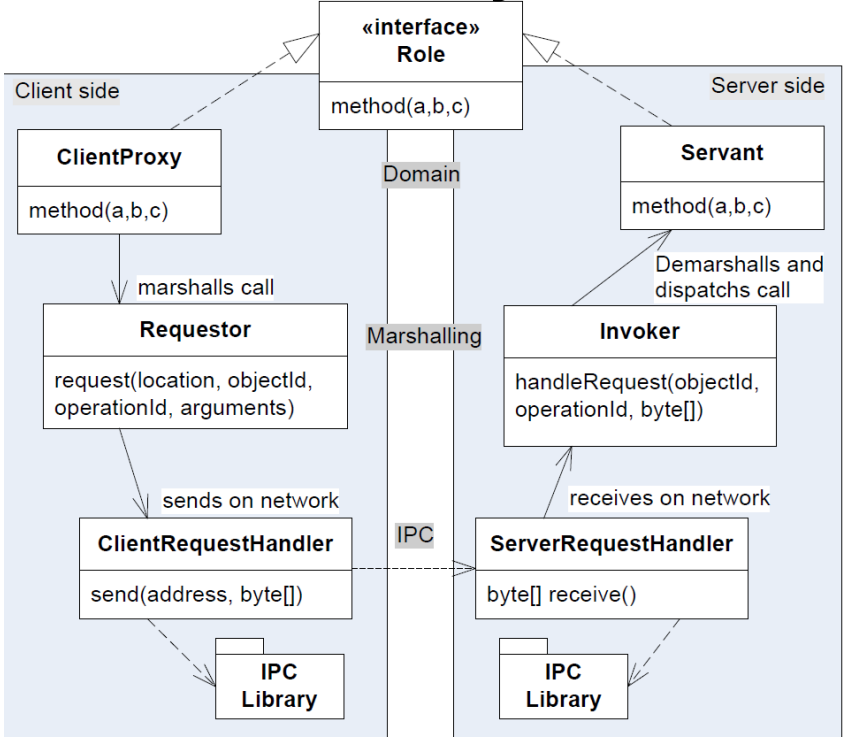
AARHUS UNIVERSITET

Using FRDS.Broker

- Server: **PlayerServant, CaveServant**
- Client: **PlayerClientProxy, CaveClientProxy**



https://leanpub.com/frds



AARHUS UNIVERSITET

Modules

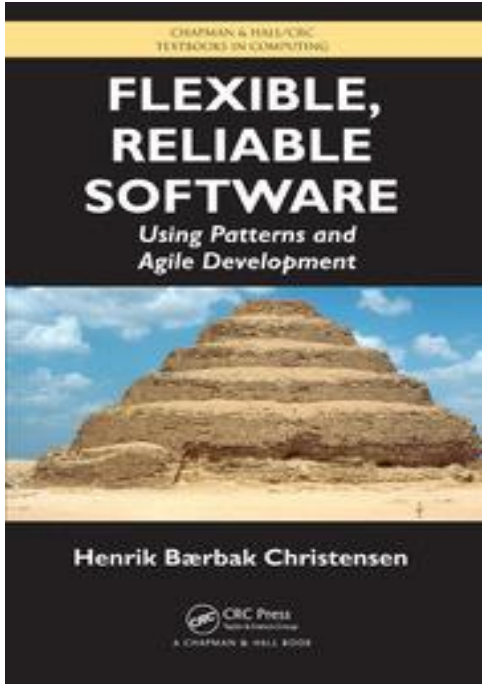
- … are called ‘projects’ in Gradle terminology

```
csdev@m51f19hbc:~/proj/cave$ ./gradlew projects
:projects

------------------------------------------------------------
Root project
------------------------------------------------------------

Root project 'cave'
+--- Project ':client'
+--- Project ':common'
+--- Project ':integration'
\--- Project ':server'
```

Flexibility

- Key Quality I strive for in my books: *Flexibility*
- Core Abstractions
  - Abstract Factory: *Create delegates*
  - ObjectManager: *Lookup delegates*

- **Chained Property Files: CPF**
  - Read at start-up…
  - Defines *all*
    - Delegate implementations
    - Host names and ports



CPF

- Example: *socket.cpf*

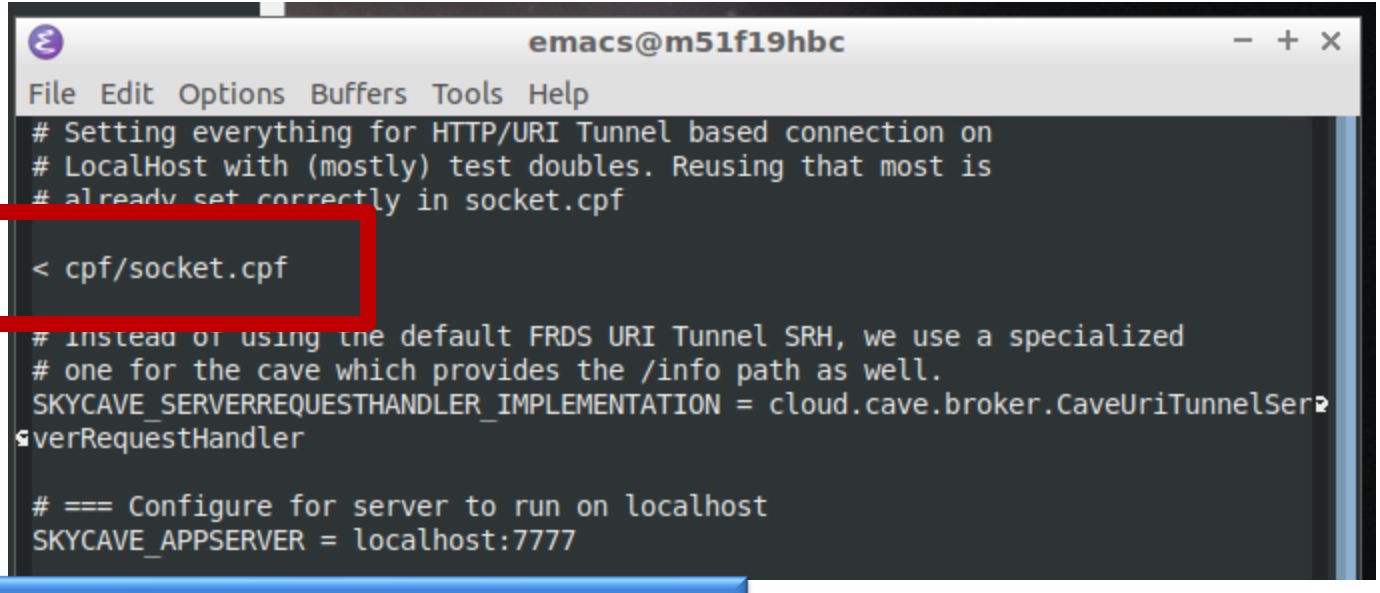
Which impl. to use?
Which host:port?

```
# Setting everything for socket based connection on
# LocalHost with (mostly) test doubles. Also acts as base CPF
# for remote configurations of daemon.

# === Configure for socket communication on server side
SKYCAVE_SERVERREQUESTHANDLER_IMPLEMENTATION = frds.broker.ipc.socket.SocketServerRequestHandler

# === Configure for server to run on localhost
SKYCAVE_APPSERVER = localhost:37123

# === Inject test doubles for all delegates (Note IP endpoints are dummies)

# = Subscription service
SKYCAVE_SUBSCRIPTIONSERVICE_CONNECTOR_IMPLEMENTATION = cloud.cave.doubles.TestStubSubscriptionService
SKYCAVE_SUBSCRIPTIONSERVICE_SERVER_ADDRESS = notused:42042

# = Cave storage
SKYCAVE_CAVESTORAGE_CONNECTOR_IMPLEMENTATION = cloud.cave.doubles.FakeCaveStorage
SKYCAVE_CAVESTORAGE_SERVER_ADDRESS = notused:27017

# = Quote service
SKYCAVE_QUOTESERVICE_CONNECTOR_IMPLEMENTATION = cloud.cave.doubles.TestStubQuoteService
SKYCAVE_QUOTESERVICE_SERVER_ADDRESS = notused:6777

# = Player Name Service - defaults to the simple in memory one which
# operates correctly in a single server/single threaded non-loaded setting
SKYCAVE_PLAYERNAMESERVICE_CONNECTOR_IMPLEMENTATION = cloud.cave.server.InMemoryNameService
SKYCAVE_PLAYERNAMESERVICE_SERVER_ADDRESS = notused:11211

# = Inspector implementation - defaults to the simplest in memory one
SKYCAVE_INSPECTORSERVICE_CONNECTOR_IMPLEMENTATION = cloud.cave.server.SimpleInspector
SKYCAVE_INSPECTORSERVICE_SERVER_ADDRESS = notused:0
```

AARHUS UNIVERSITET

Chaining

- We will start by mostly using ‘http.cpf’



Allows defining new configurations that *inherit* all properties of an ancestor…

AARHUS UNIVERSITET

Solving Exercises

- You must define CPFs for each exercise you solve
  - ‘mongo’ exercise
    - src/main/resources/cpf/mongo.cpf

```
# My solution to mongo on localhost

< cpf/http.cpf
```

```
# = Cave storage
SKYCAVE_CAVESTORAGE_CONNECTOR_IMPLEMENTATION = cloud.cave.DONOTDISTRIBUTE.MongoDBCaveStorage
SKYCAVE_CAVESTORAGE_SERVER_ADDRESS = localhost:27017
```

- Then start your daemon with the proper configuration
  - gradle daemon –Pcpf=mongo.cpf

Testability

Central for DevOps

AARHUS UNIVERSITET

Testing and DevOps

- Speedy implementation and deployment is central
  - So, a ‘quality gate’ of 1.000 hour manual tests is *no-go*
- *Have automated tests in place for all/most code*
- I am myself a sworn *test-driven development* believer ☺

```
// TDD movement of player
@Test
public void shouldAllowEastWestMoves() {
  player.move(Direction.EAST);
  description = player.getShortRoomDescription();
  assertTrue( message: "EastWestMove 1 missing proper description",
      description.contains("You are inside a building, a well house for a large spring."));

  player.move(Direction.WEST);
  description = player.getShortRoomDescription();
  assertTrue( message: "EastWestMove 2 missing proper description",
      description.contains("You are standing at the end of a road before a small brick"));
```

Do no implement anything without having production code covered by tests!!! It is an exam evaluation criteria!

AARHUS UNIVERSITET

ToolStack

AARHUS UNIVERSITET

Open Source

- It is not a cutting edge programming course, so…
- Java 11, Gradle 6, IntelliJ

Gradle

- *Configuration based* to a large extend…
- Know how the folder structure works!
  - src/main/java root of production code
  - src/test/java root of unit test code
  - src/main/resources root of resources
    - (the CPFs must reside in ‘cpf/’ subfolder)

Mxx

- Avoid the hazzle of installing everything yourself
  - And get used to Linux – you will need it for your Docker stuff