stockExchange

SBC SS2015 - Group 3

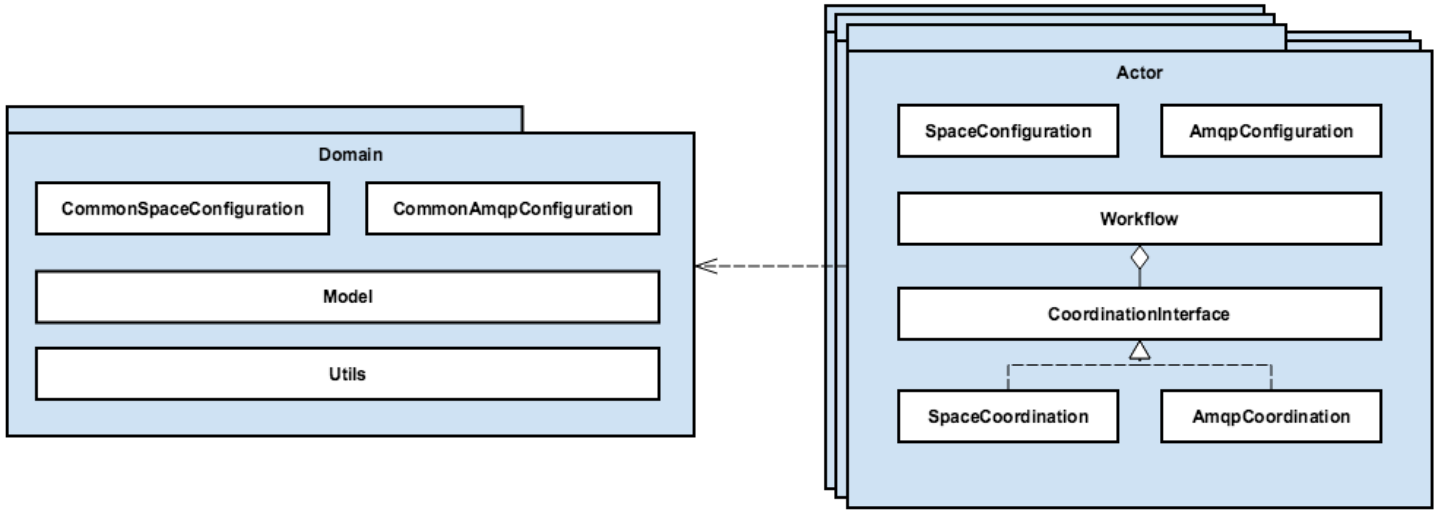
Ramon Lopez, Martin Dietl

Technology Stack

- Spring (Boot) - Application Framework
- Swing - GUI Framework
- MozartSpaces - Middleware
- RabbitMQ - Messaging Middleware (alternative implementation)

App Architecture

Actors: Market, MarketAgent, Broker, Investor, FondManager, Company, (MarketDirectory)



Why Spring?

- easy configuration via Spring Beans

```
@Configuration
@Profile("space")
public class CommonSpaceConfiguration {
```

```
@Bean MzsCore core() {
    return DefaultMzsCore.newInstanceWithoutSpace();
}
```

- Dependency Injection

```
@Component
public class MarketArgsConfiguration<T> {
```

```
@Autowired
private MarketArgsConfiguration marketArgs;
```

- Profile support with annotations

```
@Service
@Profile("space")
public class SpaceCoordinationService implements ICoordinationService {

@Service
@Profile("amqp")
public class AmqpCoordinationService implements ICoordinationService {
```

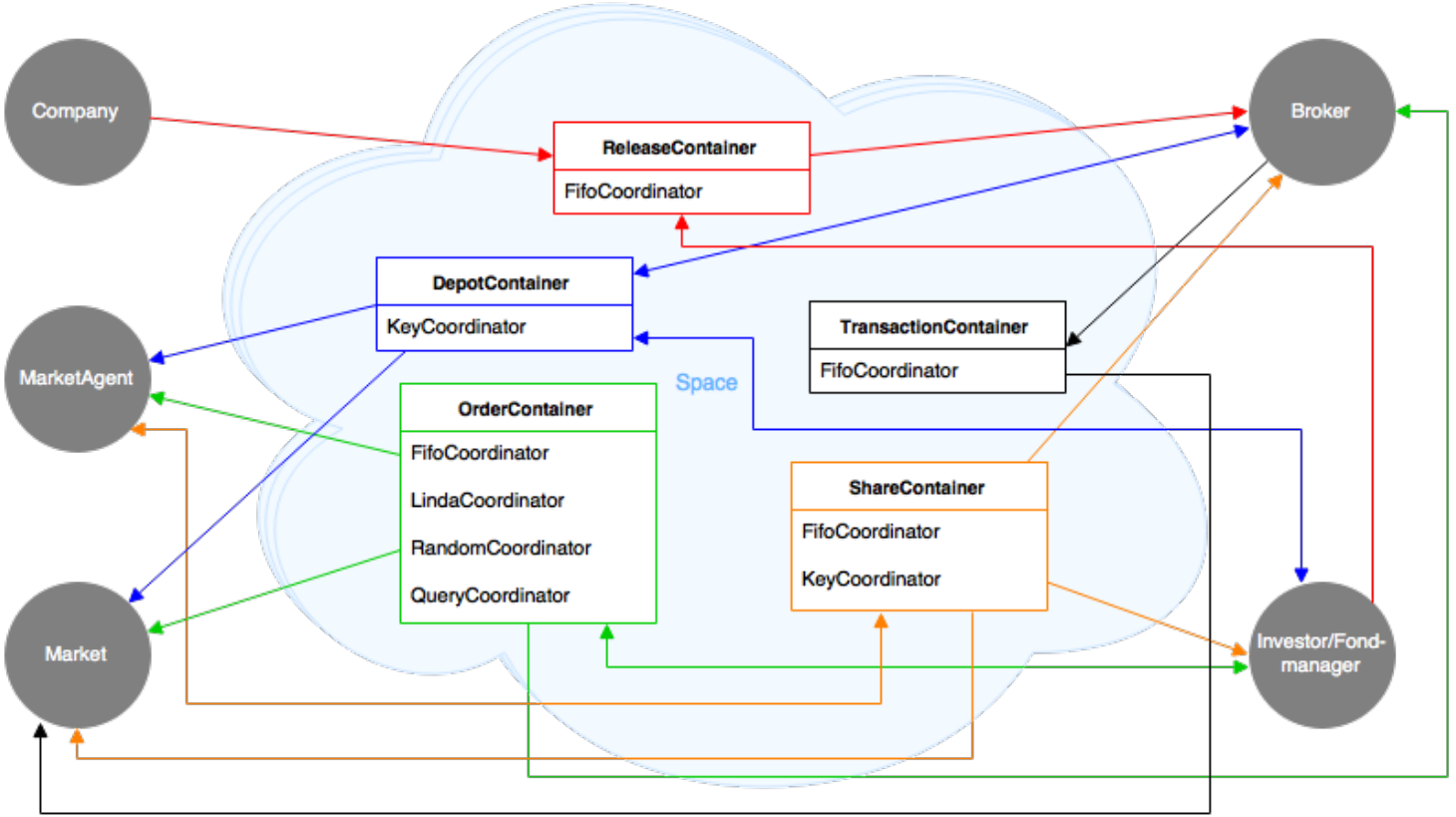
Why MozartSpaces?

Mozart Spaces

- Transaction Management
- Containers and various Coordination-Models
- lightweight API (easy entry access)
- in detail explained in the lectures
- good support from tutors
- ideally suited for given task according to required coordination design patterns

Space Coordination Model

Mozart Spaces



Why RabbitMQ (AMQP)?

- great API in Spring (Spring AMQP Framework)
- no need to configure queues to enable the Request/Answer pattern

```
result = (ArrayList<ShareEntry>)template.convertSendAndReceive(exchangeKey, CommonRabbitConfiguration.MARKET_RPC,
```

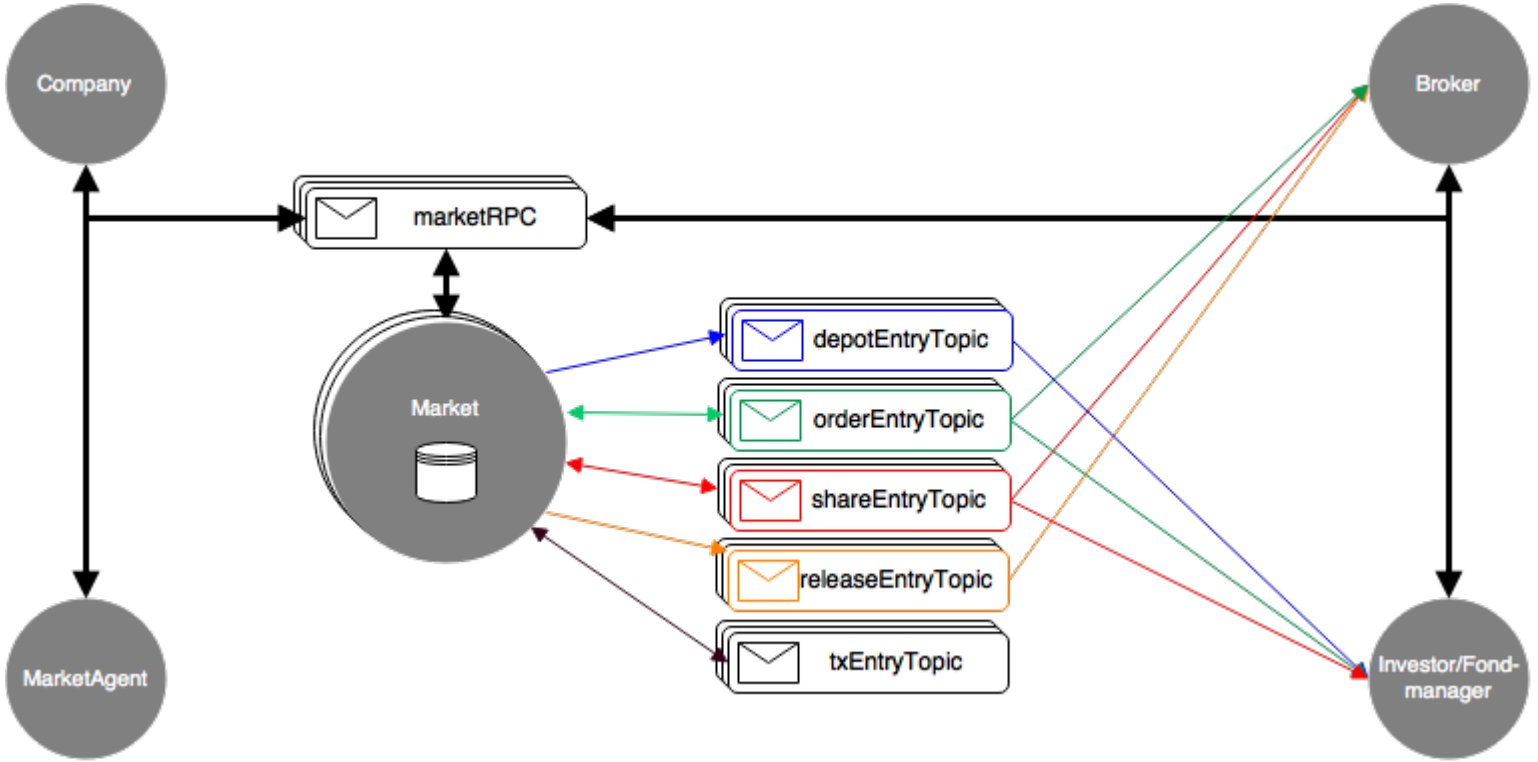
- all group members are experienced in dealing with RabbitMQ
- often used technology in the domain of cloud computing

Trade-off:

- the data store has to be implemented explicitly
- each market has to manage its own data store
- market is responsible to send entry notifications
- transaction management has to be implemented explicitly (we didn’t)

AMQP Coordination Model

RabbitMQ



Lab 2 Extension Issues

- coordination configuration adaption to support multiple markets
  - simple for Spaces - assign containers to space
  - inconvenient for Amqp - support multiple brokers or exchanges on single broker
- coordination interface/implementation adaption (multiple markets)
- reuse/extend the investor app to act as fond manager
- extend broker and investor app to enable prioritized orders
- introduction of a market directory to enable cross market share manipulation
  - each app registers active markets via RMI to the directory
  - market agent calls directory to manipulate “fonds” with distributed shares

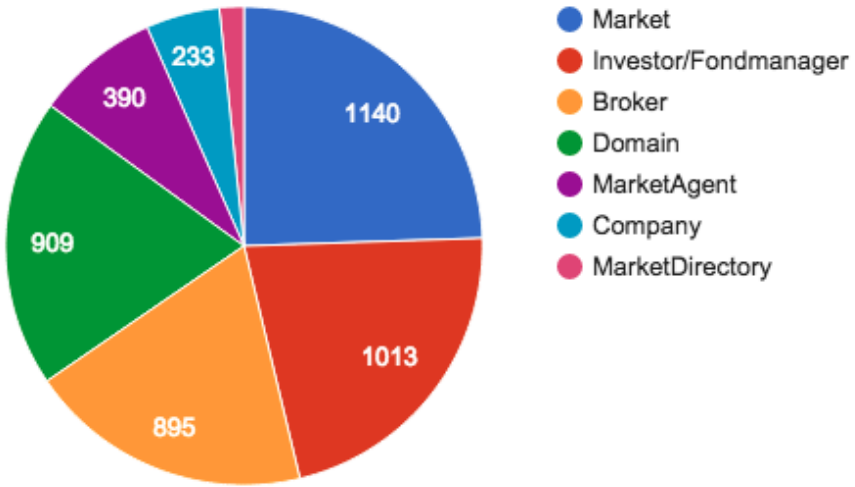
Approximate Implementation Effort

Lab 1	
Task	**Effort [h]**
project infrastructure	7
domain model	3
space configuration	3
amqp configuration	4
investor gui/workflow	4
investor space coordination	2
investor amqp coordination	1
broker workflow	5
broker space coordination	3
broker amqp coordination	1
market gui/workflow	5
market space coordination	3
market amqp coordination	1
market amqp store	4
company workflow	1
company space coordination	1
company amqp coordination	1
marketAgent workflow	2
marketAgent space coordination	1
marketAgent amqp coordination	1
	53

Lab 2	
Task	**Effort [h]**
project infrastructure	2
domain model refactoring (renaming)	2
command line args for markets	2
space configuration adaption	2
amqp configuration adaption	3
investor prioritized orders	1
investor fonds manager extension	3
broker fonds manager extension	2
market fonds manager extension	0.5
marketAgent fonds manager extension	2
investor/fonds manager multiple markets adaption	2
marketAgent multiple markets adaption	1
marketDirectory RMI service	2
marketAgent cross manipulation	2
	30.5

LOC

LOC Actors



LOC Technology