

Robotbakery

Gruppe 6 - Matthias Höllthaler, Tobias Ortmayr

Technologies

- Space-Implementation:

XVSM (MozartSpaces)

- Alternativ-Implementation:

JMS with Apache ActiveMQ

- JavaFX (Frontend), Java8

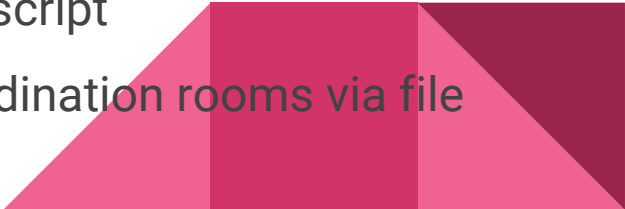


Reasons for technology choices

- Assignment is predestined for a PCO-Pattern
 - XVSM and JMS can be used without major architectural delimiters
- After researching we decided the XVSM was the most interesting choice to implement
- Previous experience with JMS and JavaFX
- Both middlewares are well documented
- JavaFX's observable lists are well suited for a PCO-Pattern



Architecture (1)

- Main idea: provide a generic framework which allows easy integration of both middlewares
 - Split into 4 modules
 - Core
 - UI
 - XVSM
 - JMS
 - Core includes entities, actors and interfaces for services and listeners
 - Own Startup classes for easy actor instantiation via script
 - For testing purposes -> loading models into the coordination rooms via file
- 

Architecture (2)

- Each Actor has a dedicated service interface for interacting with the coordination room (e.g. IBakeRobotService)
- Own generic change listener and notifier classes. (e.g. IBakeryChangeListener)
- Transaction and Transaction Manager interfaces which are used to wrap the middleware specific Transaction classes
- Actors can execute transactional tasks (basically runnables with a boolean return value). If the task returns false a rollback is initiated otherwise the transactions is committed



Architecture (3)

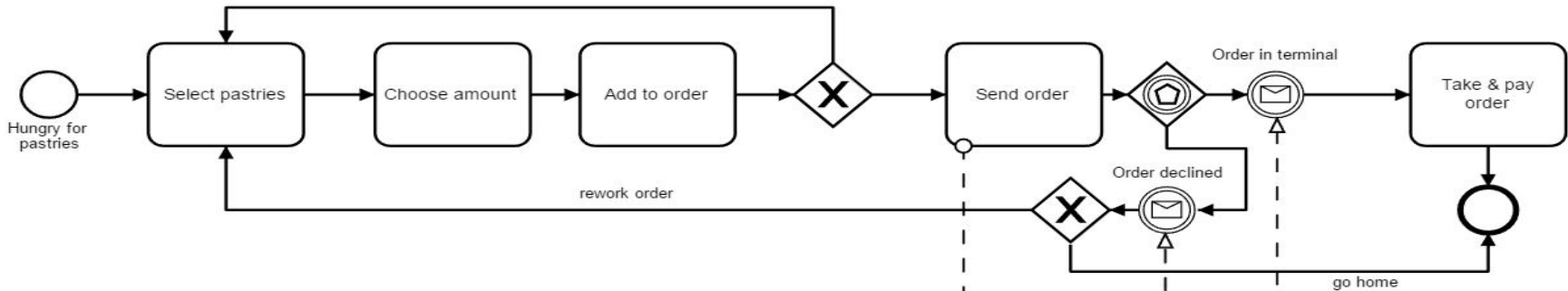
- XVSM and JMS specific implementations of the previously mentioned interfaces
- Dedicated startup classes which inject the concrete implementation into the actors
- Util classes which provide helper method for sending , receiving and counting of objects



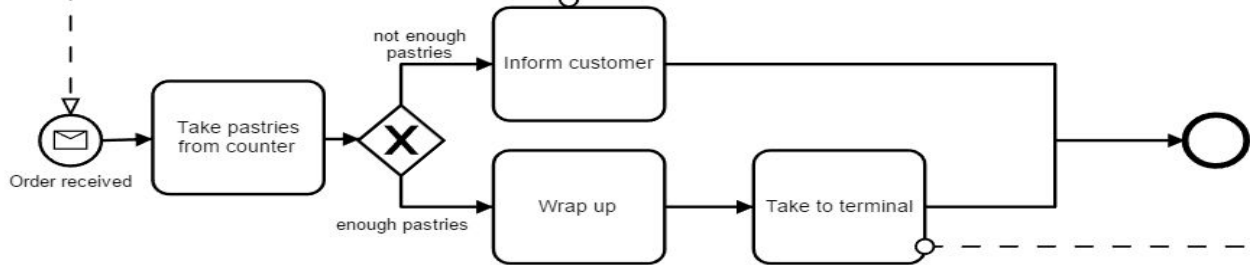
Coordination

- Actors are interacting with the coordination rooms via dedicated service interfaces
- XVSM
 - embedded space for bakery and each delivery customer
 - 4 Containers for Bakery, 1 Container for delivery customers
 - Mostly QueryCoordinators and TypeCoordinators (FIFO-Coordinator in BakeRobot)
- JMS
 - One or more Message Queues for each coordination room
 - Notification topic
 - Use of queue Browsers and message selectors to retrieve functionality as provided by XVSM

Customer

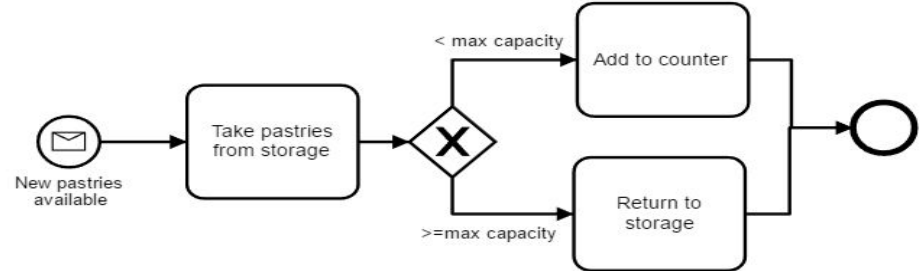


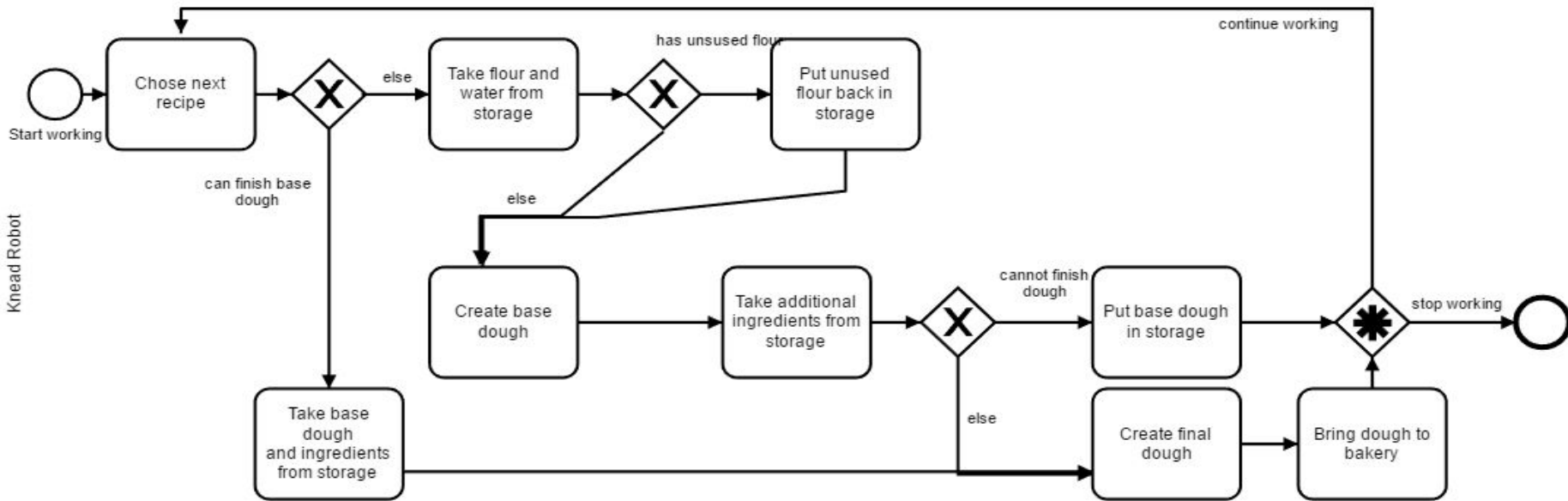
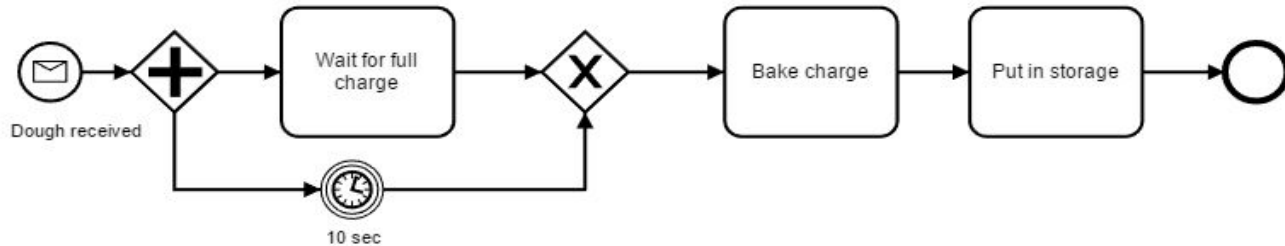
Process orders



Service Robot

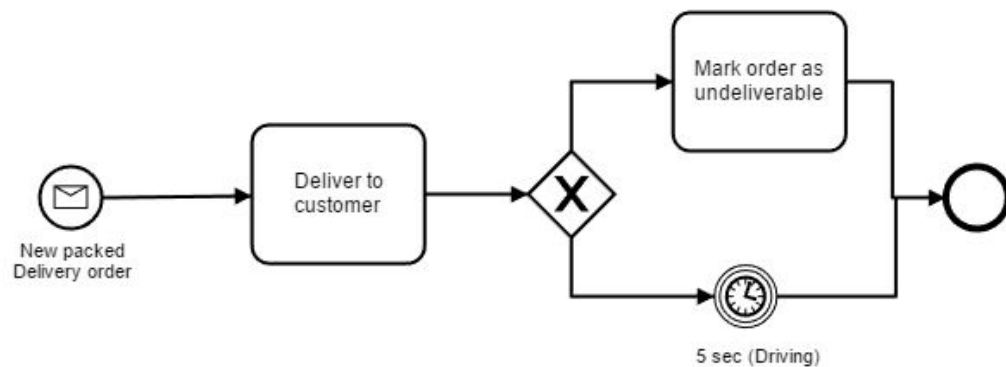
Transfer pastries from storage



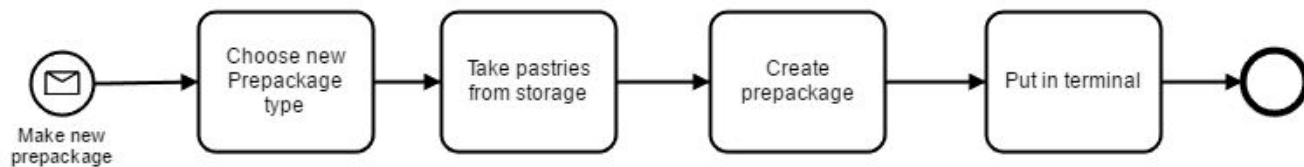


Lab2

Delivery Robot



Create prepackage

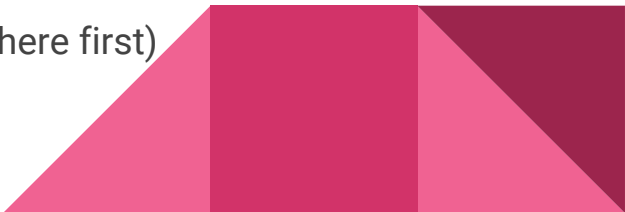


Lab2 - Changes

- Focus in Lab1 was on extensibility -> Changes required for Lab2 could be implemented rather easily
- Most of the time extending the service classes and adapting the actor run method was enough
- For delivery customers some refactorings in the ui module were necessary to avoid duplicate and boilerplate code
- Changes in maven configuration and XVSM setup to enable embedded spaces for clients



Estimated Effort

- Implementation of Space general shorter than JMS
 - e.g. KneadRobot specific parts: 170 LOC vs 240 LOC
 -
 - Task with Core, UI and Notifications ~ 350 - 500 LoCs (10-15 hours)
 - Work Split between Project Parts:
 - Core: 40%
 - UI: 15 %
 - XVSM: 20% (could be less, but we implemented every feature here first)
 - JMS: 25%
- 

XVSM vs. JMS

- Assignment obviously optimized for the spaced-based approach
- No major issues with XVSM
- JMS has certain limitations:
 - Multiple message consumer for one with overlapping message selectors not possible in one session
 - -> workaround with queuebrowser necessary (AL)
 - Session system for transaction management not always working reliably
 - Additional control structures necessary



