Rainbow
Deal Approval Process

Workflow Modelling and Process Management
Final Presentation

Group 4
Matthias **Wagner** | Sebastian **Frisch** | Lukas **Genßler** | Manuel **Stummerer** | Yi **Wang**

Rainbow is a workflow management application supporting and automating the approval and contractualization of a project request

Overview - idea and output of the application

IDEA

- Processing a standard project request takes a **lot of manual steps**
- Most decisions in this process follow **some sort of heuristic rules**
- **Implement a workflow management application** that takes over these steps
- These steps can easily be automated to **improve time to contract and customer satisfaction**

OUTPUT

- Rainbow takes in user request based on a **questionnaire covering majority of possible projects**
- **General characteristics** of requested project and KPIs of customer company are collected
- These properties are used to **determine outline of project** like required staff, pricing & duration
- Customer is also provided with the **possibility to give feedback** to an initial proposal.

To cover all necessary steps the rainbow process is split into five major tasks while the customer is always kept informed of their current request status

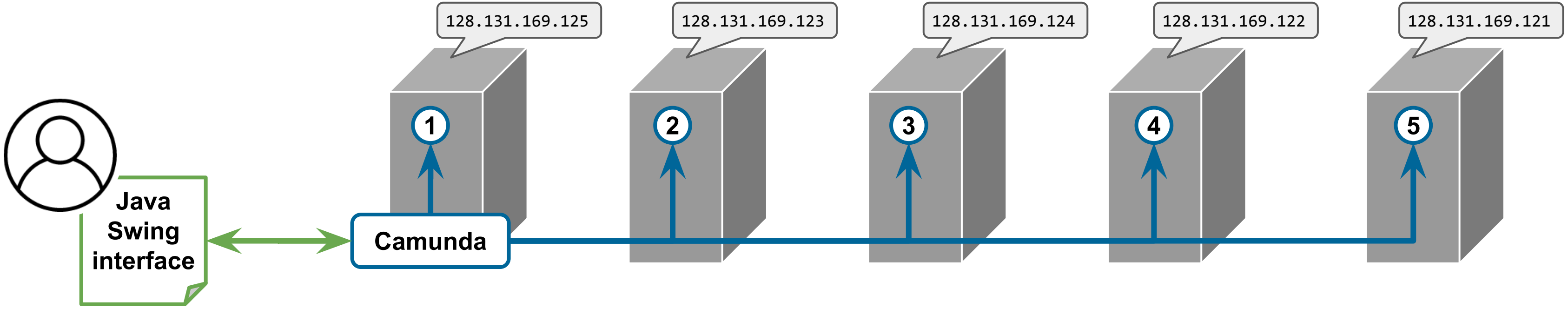
Task structure including handled exception

	1 Deal Size Estimation	2 Strategy Development	3 Solution Engineering	4 Offer Placement	5 Contractualisation
TASKS	• Processing of project request • Rough estimate on potential deal profitability • Assignment to responsible manager according to estimation	• Development of project and implementation strategy • Iterative and agile review loops • Approval of strategy by account manager	• Development of final customer specific solution concept • Calculation of costs, estimation of needed resources and analysis of risks	• Creation and validation of an offer for developed solution • Presentation of the offer to customer • Incorporation of customer feedback iterating over previous steps	• After accepted offer • Presentation of final contract to customer
EXCEPTIONS	• Rejection in case of non profitability	• Cancelation in case of no feasible strategy being found	• Cancelation in case solution parameters would result in non profitable project	• Rejection in case of 2nd negative customer feedback	• No rejection / cancellation possible

Full transparency through ongoing feedback to the customer about the current status of their request

Camunda is managing the five individual tasks implemented with java and deployed on separate servers

Implementation details, tools, server, programming language



- Customer interface implemented via Swing (Java)
- Customers are identified by their unique customer ID (in our case matriculation number)
- Five individual servers communicating with the customer over camunda web API providing necessary information
- Camunda managing the tasks in the process following deployed BPMN
- Tasks implemented as Java applications subscribing to tasks of the camunda process

User icon: https://www.flaticon.com/

Different KPIs have been defined in order to gain insights in how the process can be improved

Evaluation of KPIs, possible improvements

Overall testing metrics

- Testing conducted by fellow students
- 18 participants
- From 15.06.2020 to 23.06.2020
- 2 processes never finished by customers

Expected Profit Rate

- Profit divided by deal size
- Average 35,32% (Min 26,43%; Max 44,43%)
- Reject request if it is not lucrative enough

Customer Cancellation Rate

- Initial offers rejected by the customer
- 3 out of 18 (16.7 %)
- Room for improvement in automated offer generation

Company Cancellation Rate

- Processes canceled by the company
- 2 out of 18 (11.1 %)
- Most cancellations due to unrealistic customer requests

Time to Contract

- Time between initial request and sending of final contract
- Average 2 min 36,5 sec (Min 56 sec, Max 6 min 48 sec)
- Time also depends on user progressing through steps and whether they request changes to the offer

Icon: https://www.flaticon.com/

From the challenges we had to tackle and failures we made, we were able to learn valuable lessons during the course of the project

Feedback and Failures - what did we learn?

- Well working systems are from utmost importance
- Document your meetings properly
- Specify technical and functional system properties so that everyone can relate and knows what is meant
- It is better to check once too much than once too little
- Choose a proper communication tool suited for software projects (e.g. slack instead of whatsapp)