M1: Bike Insurance Provider

Michael Raffelsberger (11772903)

March 2021

Contents

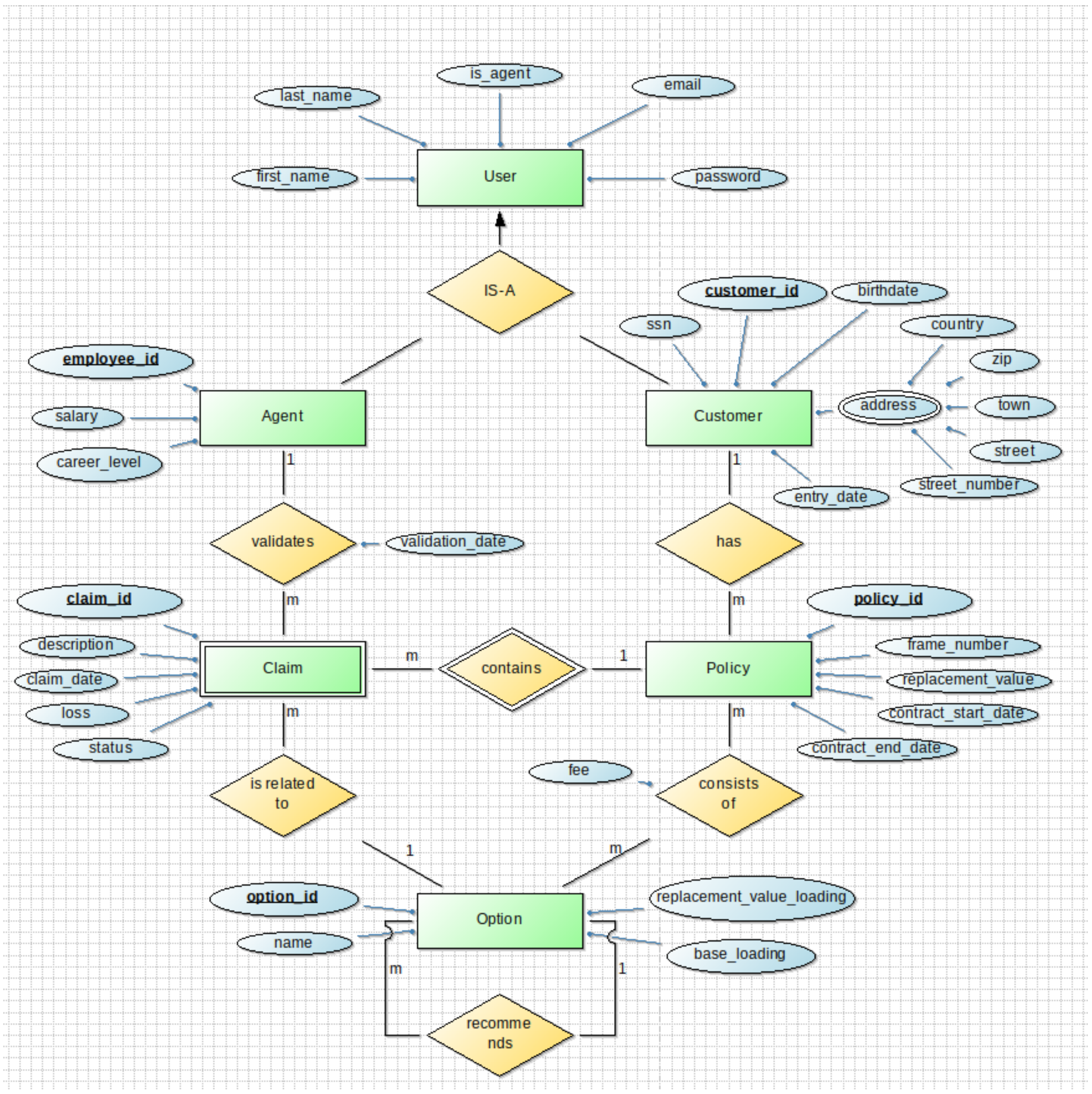
1 Business Model Outline / Description

The idea of this project is to implement a system for a **little bike insurance provider**. It is used by customers to execute simple actions like creating a new policy or reporting a claim for an existing policy. Apart from the customer side, insurance agents will also use the tool to e.g. accept or reject claims. This means it basically manages the business logic of our company, however without dealing with all the payment and financial accounting stuff.

Both **customers and [insurance] agents are users** and can log into the tool with their e-mail address and password. A customer (uniquely identified by his/her customer-id) can have multiple policies.
A **policy** has a uniquely identifiable policy-id, a frame-number[1], the replacement value of the bike as well as a start- and an end-date.
Each policy consists of one or more **options (e.g. theft, fire, vandalism, etc.)** against which the policy insures the customer's bike. Each policy-option combination has a fee that the customer pays on a monthly basis. An option in general has (an option-id, a name and particularly) a base loading and a replacement value loading. These values determine the default fee for an option in a policy. If the base loading is 1 euro and the replacement value loading is 0.002 euros, a bike with a replacement value of 1500 euros will cost $1 + 1500 \cdot 0.002 = 4$ euros (unless the customer got some special offer). Options recommend each other. This means that on creating a new policy choosing the theft option, a customer will be actively offered an insurance against fire damage if theft recommends fire for instance.
A policy contains zero or more **claims** and each claim is related to a specific option covered by the policy (e.g. the fire option in case of domestic coal). It is the job of the agent to investigate reported claims and either accept or reject them. By doing so, the status of a claim will change from "reported" to "in-process" to either "rejected" or "accepted". Claims are the only weak entity in the model. If a policy is deleted, all its claims are also deleted in a cascading way. This might happen if a person under 18 has created a policy without permission for instance. Otherwise, if a policy ends, it is not deleted but gets a non-null value for its end-date.

[1]identifies the bike - not in our system but in general, see: wikipedia-Rahmennummer

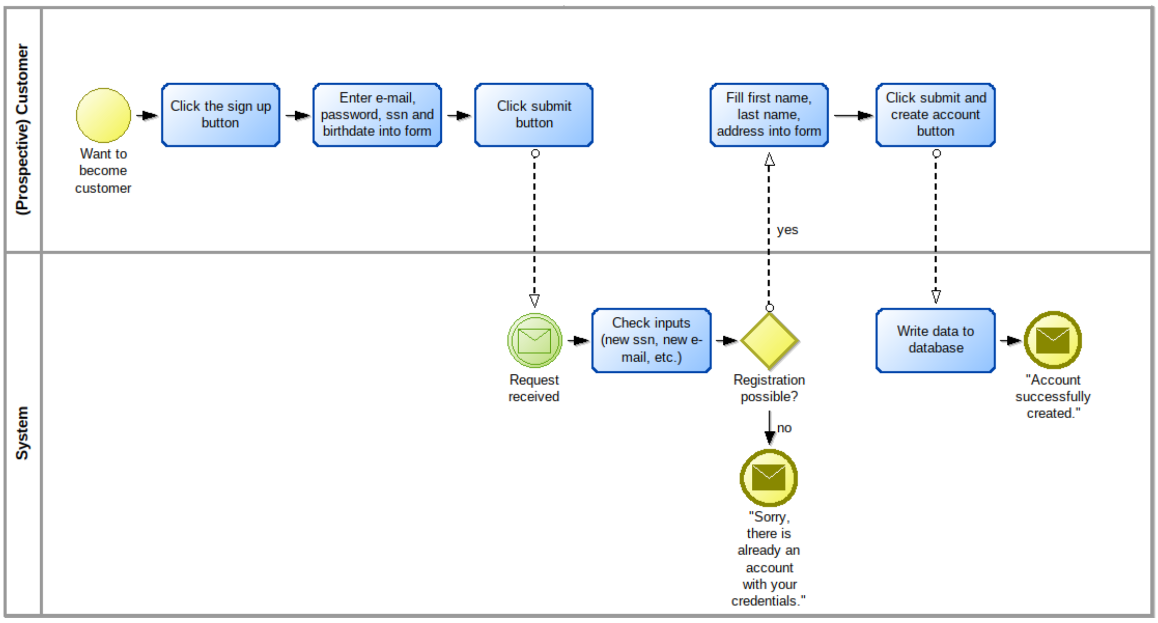
2 ER-Diagram


is_agent
last_name
email
first_name
User
password
IS-A
customer_id
birthdate
ssn
country
employee_id
zip
salary
Agent
Customer
address
town
career_level
street
1
1
entry_date
street_number
validates
validation_date
has
claim_id
m
m
policy_id
description
frame_number
claim_date
Claim
m
contains
1
Policy
replacement_value
loss
contract_start_date
status
m
m
contract_end_date
fee
is related to
consists of
1
m
option_id
Option
replacement_value_loading
name
base_loading
m
1
recommends

3 Use-Cases

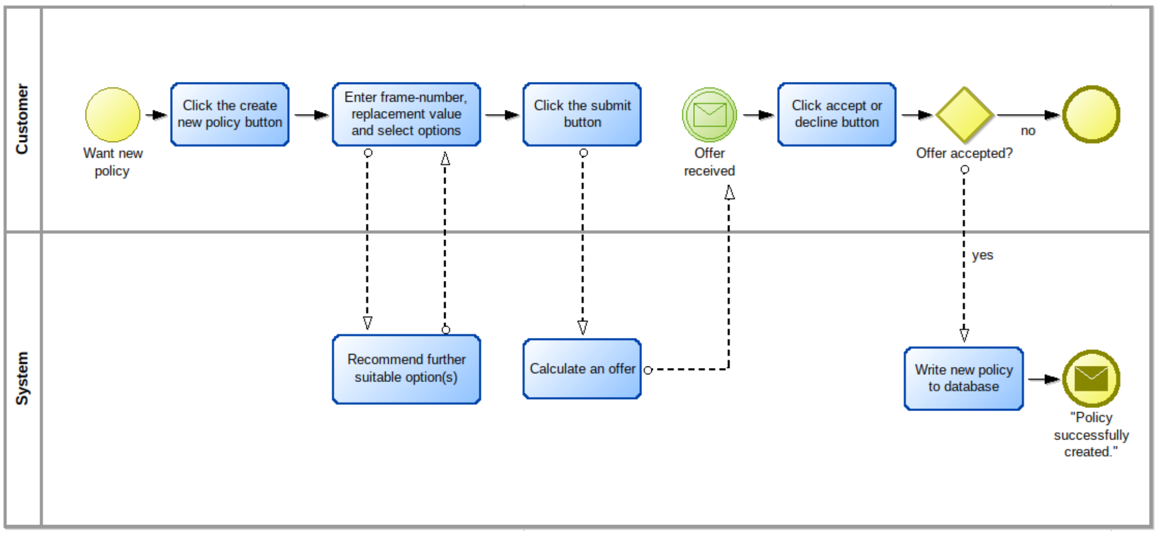
3.1 Use-Case 1: Create a customer account

- **Actor:** (Prospective) Customer
- **Objective:** Create an account and become a customer
- **Short description:** A person willing to become a customer creates an account by providing his personal information and choosing a password.
- **Preconditions:**
  - Person is not a customer yet. (check via social security number (ssn))
  - Person has an e-mail address that is new to our database.
  - Person is 18+ years old. (check via birthdate) (in real life the person would have to provide some kind of identification like a passport)
  - Person is on our login platform. (global starting page)
- **Expected execution:**
  1. Person clicks the “sign up” button.
  2. In a form, person fills in e-mail, password (2x), ssn and birthdate and submits.
  3. System checks if e-mail or ssn do already exist in the database, if the person is already 18 years old and the passwords are identical.
  4. In case of a success, the person has to fill first name, last name and address into a form and click the “submit and create account” button.
  5. The provided information along with the entry date, is-agent = false and a new customer-id are stored in the database.
- **Postconditions:**
  - **success:** New customer is included in the database and can from now on log in with the e-mail address and the chosen password and make use of our services. Success message is displayed. (see: BPMN)
  - **invalid input (known ssn, etc.):** Display “Sorry, there is already an account with your credentials.”
  - **other error:** Display message: “Error, please try again later.”


(Prospective) Customer
System
Want to become customer
Click the sign up button
Enter e-mail, password, ssn and birthdate into form
Click submit button
Fill first name, last name, address into form
Click submit and create account button
yes
Request received
Check inputs (new ssn, new e-mail, etc.)
Registration possible?
no
Write data to database
"Account successfully created."
"Sorry, there is already an account with your credentials."

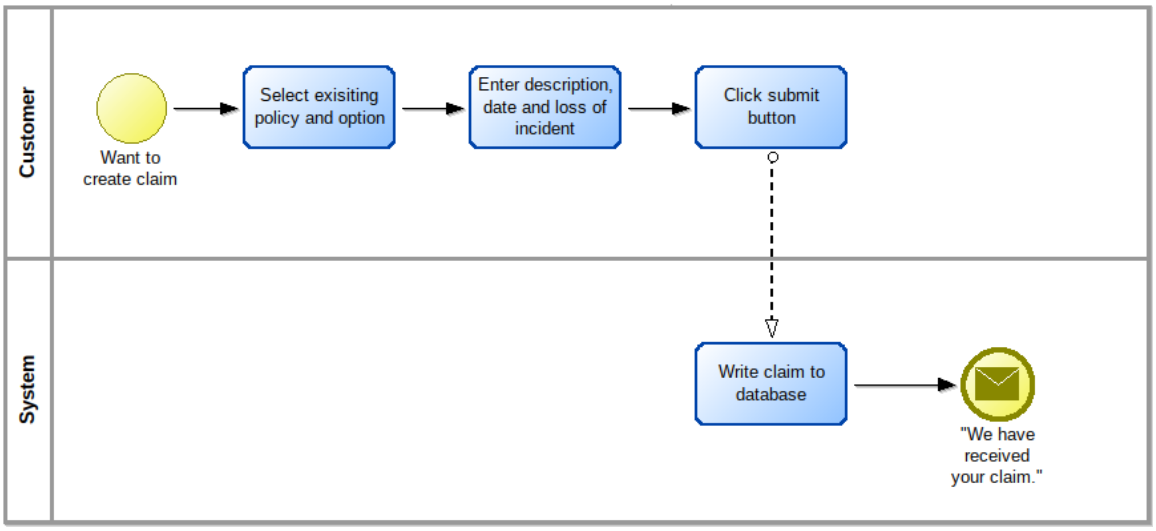
3.2 Use-Case 2: Create a policy (main use-case 1)

- **Actor:** Customer
- **Objective:** Add a new policy to account
- **Short description:** A customer wants to add a new policy to his/her insurance portfolio to insure a bike.
- **Preconditions:**
  - Customer is logged into his/her account.
- **Expected execution:**
  1. Logged in customer clicks the "create new policy" button.
  2. In a form, frame-number and replacement value are entered and desired options are selected. An unselected option might be actively recommended.
  3. Customer clicks the "submit" button.
  4. Offer is displayed.
  5. Customer either clicks the "accept" or "decline" button.
- **Postconditions:**
  - **success:** Policy is added to the database. Bike is insured from the next day on. Success message is displayed. (see: BPMN)
  - **offer declined:** "Nothing happens", i.e. no data is written and the user is just logged into his or her account as in the beginning of the use-case.
  - **other error:** Display message "Error, please try again later."


Customer
System
Want new policy
Click the create new policy button
Enter frame-number, replacement value and select options
Click the submit button
Offer received
Click accept or decline button
Offer accepted?
no
yes
Recommend further suitable option(s)
Calculate an offer
Write new policy to database
"Policy successfully created."

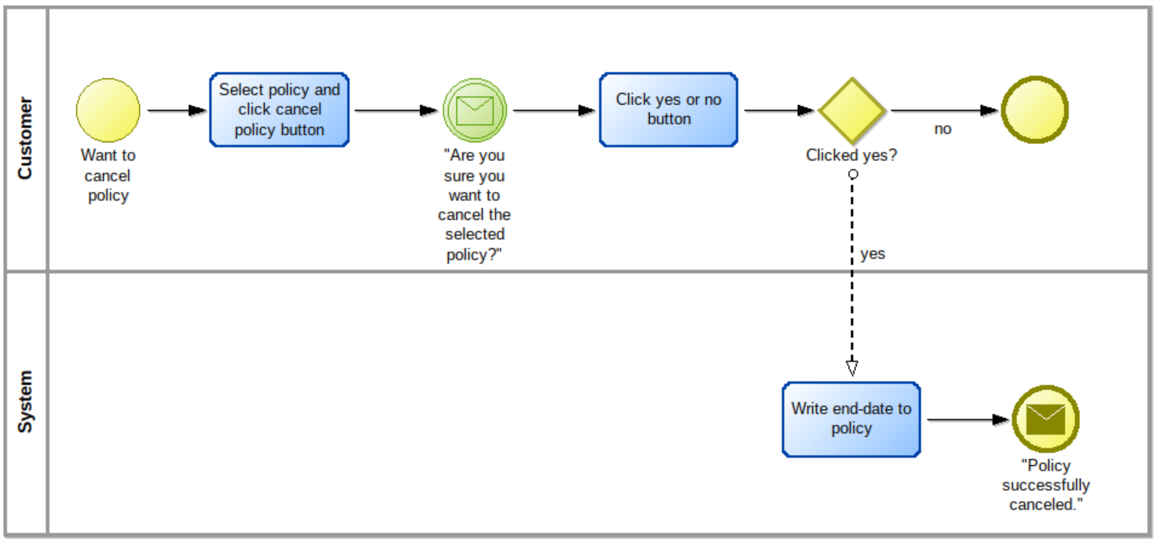
3.3 Use-Case 3: Report a claim (main use-case 2)

- **Actor:** Customer
- **Objective:** Customer reports a claim about an incident with an insured bike
- **Short description:** A customer reports a claim related to an option of one of his/her policies demanding for an insurance benefit.
- **Preconditions:**
  - Customer is logged in.
  - Customer has a policy for the bike including the option he/she wants to report a claim for.
- **Expected execution:**
  1. Customer chooses the respective policy and option.
  2. In a form, he/she enters a description (max. 200 words) describing the incident, the incident date and the loss (in euro).
  3. Customer clicks the submit button.
- **Postconditions:**
  - **success:** Claim is reported and stored in our database. The agents can now see and investigate it. The status of the claim is set to "reported". The message "We have received your claim. Our team will get in touch with you." is displayed.
  - **error:** Print message "Error, please try again later."


Customer
Want to create claim
Select exisiting policy and option
Enter description, date and loss of incident
Click submit button
System
Write claim to database
"We have received your claim."

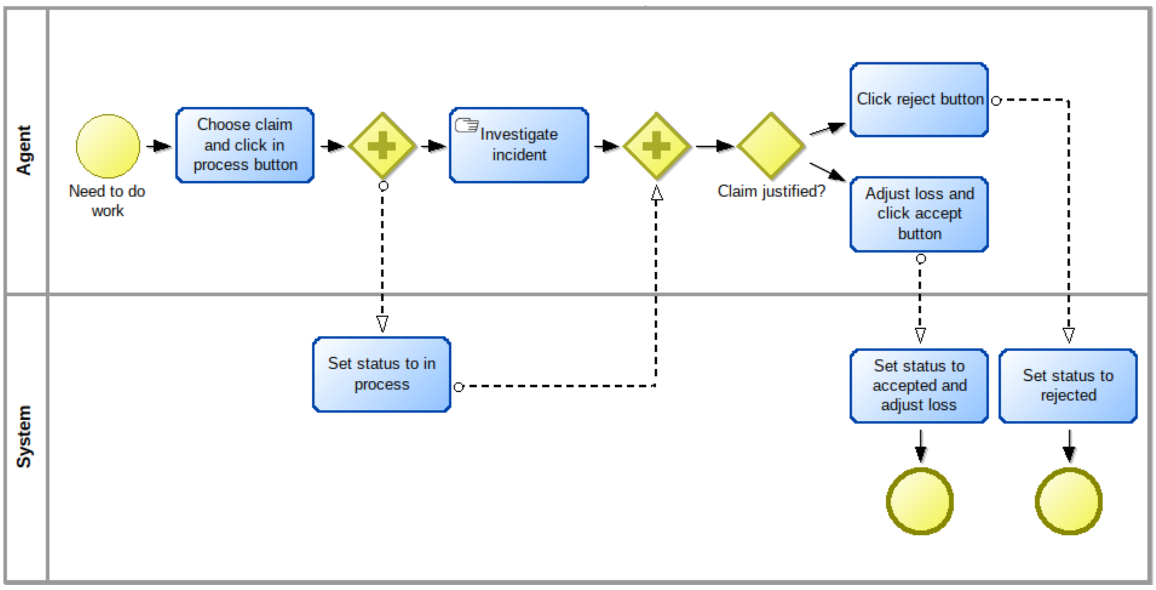
3.4 Use-Case 4: Cancel a policy

- **Actor:** Customer
- **Objective:** End an existing insurance contract (= policy).
- **Short description:** A customer ends an active policy.
- **Preconditions:**
  - Customer is logged in.
  - Customer has an active policy that he/she wants to stop.
- **Expected execution:**
  1. Customer selects the respective policy and clicks the "cancel policy" button.
  2. In a window, "Are you sure you want to cancel the selected policy?" is displayed.
  3. Customer clicks "yes" or "no".
- **Postconditions:**
  - **success:** If customer confirms, the respective policy gets an end-date which by default is the last day of the policy's current period (e.g. 26th of March if the policy has been created on the 26th of some month and the date is between 27th of February and 26th of March.). Customer is still insured and can report claims until the end-date. Success message is displayed. (see: BPMN)
  - **customer clicks no:** "Nothing happens", i.e. no data is written and the user is just logged into his or her account as in the beginning of the use-case.
  - **other error:** Print message "Error, please try again later."


Customer
System
Want to cancel policy
Select policy and click cancel policy button
"Are you sure you want to cancel the selected policy?"
Click yes or no button
Clicked yes?
no
yes
Write end-date to policy
"Policy successfully canceled."

3.5 Use-Case 5: Validate a claim

- **Actor:** Agent
- **Objective:** Validate a reported claim
- **Short description:** An agent, who can see all reported claims, processes one of them (i.e. investigates and either rejects or accepts it) so that a customer can receive his/her insurance benefits in case of an accepted claim.
- **Preconditions:**
  - Agent is logged in (as agent).
  - Some reported and not yet decided claim exists.
- **Expected execution:**
  1. Agent chooses a claim and sets its status to “in process” via a button.
  2. Agent is supposed to investigate the case offline (get in touch with the customer, ...).
  3. Agent either sets the status of the claim to “accepted” or “rejected” via a button.
  4. In case of “accepted”, agent also adjusts the loss named by the customer if necessary.
- **Postconditions:**
  - **success:** Status of the claim is updated and the claim gets a non-null value as validation date (date of “accepted” or “rejected”). If applicable, the loss is adjusted in the database.
  - **error:** The message “Error. Please talk to the system administrator.” is displayed.



4 Reports

4.1 Report 1: Customer claim report

Claims are the greatest threat to our insurance business. Therefore, we want to have a detailed claim report for all our users. The report is sorted by the last name of the customer and displays the number of active policies, the total loss, the number of claims and the date of the most recent claim. We can filter by the total loss (lower bound). The report can help the management find very expensive customers for instance. (includes entities Customer, Policy and Claim)

4.2 Report 2: Customer fee report (by option)

Fees are key to a good performance of our business. Therefore, we want to have an overview over the amount of (monthly) fees we currently earn by customer broken up by options. The report has again one row per customer (with at least one active policy) and a column for every option as well as a column for the total over all options. It is again sorted by the last name of the customer and can be filtered by the number of active policies, which is another column. (includes entities Customer, Policy and Option)

5 References (at least the most important ones)

- `https://camunda.com/bpmn/reference/`
- `https://www.youtube.com/watch?v=7_OWoVRUXXo`

6 Work Protocoll

Date	**Hours**	**Tasks**
14.03.2021	4	get familiar with task look at example solutions
16.03.2021	3	outlook how to possibly survive M2 (flask, docker, ...)
21.03.2021	6	read M1 guidelines and examples first ER-diagram draft think about possible use-cases
24.03.2021	3	finish ER-diagram write business description get familiar with BPMN modeling
28.03.2021	2	design first BPMN model (use-case 1)
08.04.2021	5	finish BPMN models and M1 PDF