Aleksander Skraastad

Assignment 1

TDT4252

Modeling insurance claim at FastClaim with ArchiMate 2.1

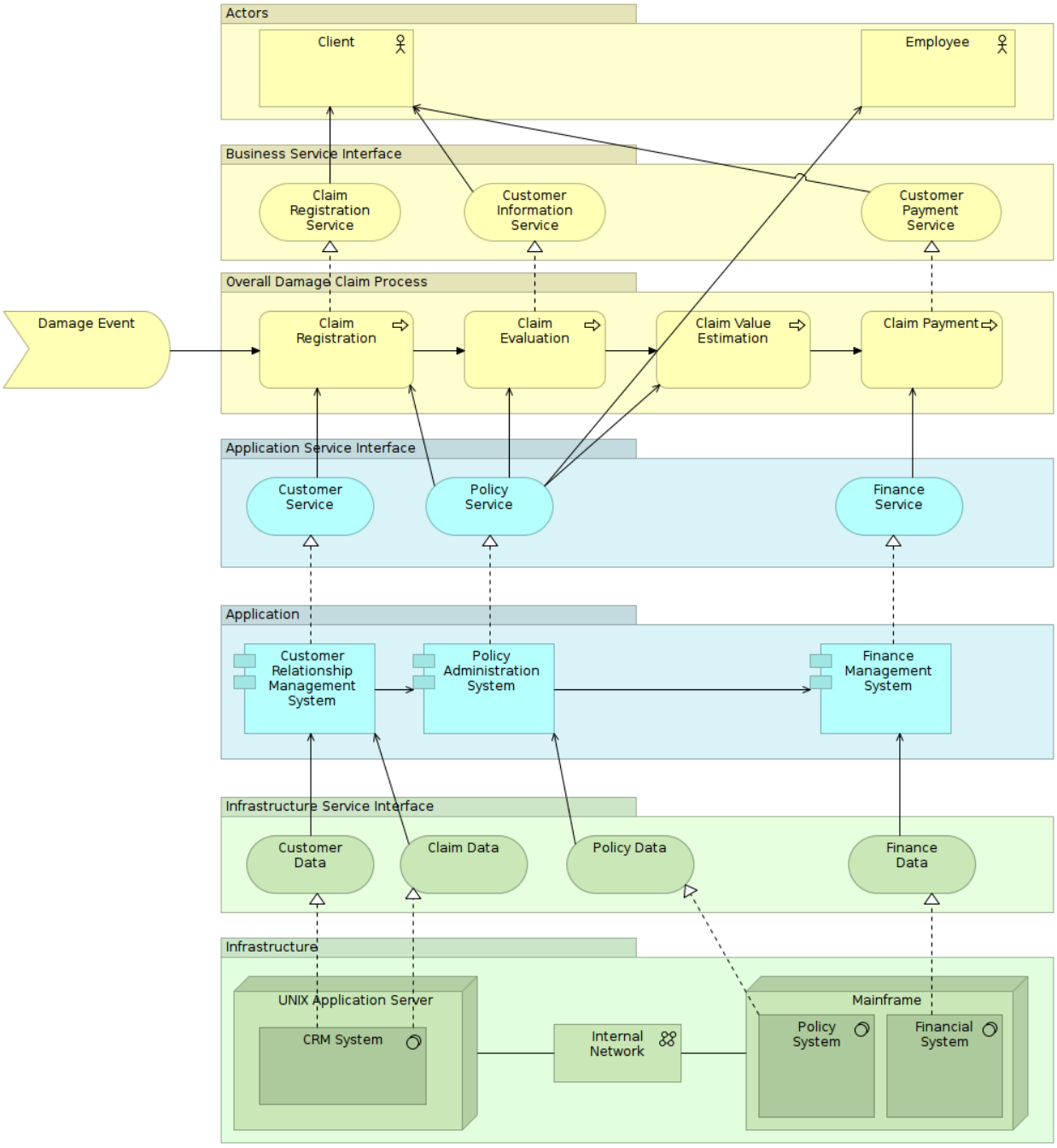
Autumn 2015

Norwegian University of Science and Technology

Table of contents

Chapter 1

ArchiMate model



Chapter 2

Modeling choices and reasoning

2.1 Modeling choices

2.1.1 Technical layer

In the technical / infrastructure layer, I chose to model the two hardware components of interest as nodes, with their respective relevant systems running on the nodes.

The UNIX Application server instance is connected to the mainframe via a network box with a descriptive label, representing that the connection is over the company's internal network. This seemed like a clear and precise way to indicate how the infrastructure should be set up.

I identified 4 different types of data that should be accessible from the infrastructure. Inspiration to use data services in the infrastructure interface was gathered from *archimate.nl*. They represent the different data needed by the applications in the form of data stored in some form of DBMS or similar transaction journal or document store.

Using nodes instead of devices also allows flexibility when physically implementing the infrastructure, with for example the use of virtual machines instead of multiple physical rack servers.

2.1.2 Application layer

From the problem description, I identified three main system components. All three seemed to me like clear candidates to being represented in the model, as they serve specific uses. The CRM Application handles ordinary customer information, as well as registered claims from a cus-

tomer. The policy administration system provides the foundation to the policy service, which handles all CRUD operations related to policies. And finally, the financial system handles the actual payment of the estimated value to the customer through the finance service.

These three were also clear from the problem description. Although their connections were not.

Since i chose to let both customer data and claims be handled by the CRM, the policy administration system needs to use data from the CRM internally in the application layer. This is to be able to fetch relevant data during claim validity evaluation and value estimation.

The system components area realized (realization connectors) through application services that can be interacted with (used by connectors) through the business processes.

2.1.3 Business layer

The business layer was not that easy to get a grip on. I found inspiration from a similar model on Wikipedia, and chose to expose three business services all connected to the customer actor. The first, claim registration, is to account for clients to register claims on their own, using a web browser. The second, is to provide the user with followup status of their claim, if auto-evaluation through the policy service inference engine was not possible for some reason. In this case, manual inspection and evaluation from an employee is needed. The customer information service allows the client to track progress, or view history of previous damage claims et cetera. Finally, the payment service handles the actual transaction from from FastClaim to the client, through use of the financial system.

In my mind, these main three business services seems to cover what the problem description layed out.

Employees are also able to create, read, update and delete policies through the policy service. This to allow claims that are pending due to edge cases not covered by the registered policies can be processed.

As with the application layer, the different parts of the overall claim process are realized through business services to the client. The exception is the value estimation, which is autonomous, and only depends on the policy application service through a *used by* connector.

The business subprocesses are all connected with a triggering connector, indicating that the initial damage event triggers the registration subprocess, which in turn triggers the next subprocess until payment has been completed.

2.2 Goals

My goals when attempting to model this architecture was primarily to keep the model as simple as possible, without losing the precision required by the problem description. Using principles of service oriented architecture and separation of concerns to group functionality into modules was also a goal.

2.3 Supplementary information or questions

I have a few questions regarding my model:

- Was my use of the business event"Damage event" correct?
- What do you think of letting both customer information and claim information be handled by the CRM?
- Is it OK for me to have used by relations within the application layer as I have modeled, or do they need to expose services through realization relations before i can use the used by relations?