Jonas Sjöberg
860224-xxxx
Linnaeus University
js224eh@student.lnu.se
https://github.com/jonasjberg
http://www.jonasjberg.com

Written during:	August 31, 2017 – September 18, 2017
Teacher responsible:	Tobias Ohlsson

Abstract

The first workshop assignment in the course *1DV607 – Object-oriented Design and Analysis with UML* at Linnaeus University during the autumn term of 2017. This workshop primarily covers initial domain modeling for some set of given fictional requirements, problem description and other background information provided by a simulated customer.

Contents

1 Problem Description

The problem description [1] is stated as follows:

> The yacht club “The jolly pirate” has started to get problems. The club has grown to such an extent that the members have a hard time keeping track of all meetings, and other important dates such as the club’s various competitions. It is also difficult for a member to change information in the membership registry, for example change address or reporting that they have a newly purchased boat. The result of this is

that members sometimes are without a reserved berth, when it became time for launch towards the spring.

The reservation of berths is operated by the club secretary and he goes through all members' boats and book their berth in the spring. This is a very "tricky" work, members owning several boats would love to have them as close together as possible, while they want their "usual" place. Much fuss and mess usually follow the announcement of this year's places, and sometimes even "private" exchanges between members occurs, which strongly opposed by the municipality for safety reasons.

The biggest problem, however, has the club's treasurer, it is quite impossible for the treasurer to manage all subscriptions manually via receipts that members come in with when they paid their fees. The fee consists of a fixed part and a variable part, where the variable component is dependent on the number and the size of the boats. Several members probably slips through each year without paying a fee, the result of wich is that the club will soon go bankrupt.

1.1 Actors

1.1.1 Primary Actors

Secretary Wants to book berths for members in a fair and effective manner. Wants to minimize "whining" when advertising of berts and avoid that private exchanges occurs. Would also be able to manage the club's calendar where important events and meetings are advertised.

Treasurer Wants to manage member payments in an efficient manner. It is also important that the Treasurer has access to history in the payments for accounting purposes.

Member A person who is a member of the boat club, probably has one or more boats registered and want a good berth for these. Want to manage their membership data, including boats, as well as to have a smooth overview of their payments. Would also like to be able to participate in various boat club meetings and social activities.

1.1.2 Supporting Actors

Third-party systems for credit card payment A service used to handle payment via credit card. Receives a transaction ID and a sum, will respond with a positive or negative result.

Third-party systems for Direct Payment A service used to manage the direct payment online banking. Receives a transaction ID and a sum, will respond with a positive or negative result. In case of positive results money is deducted from the payer's account 00:00 o'clock the next day, and the transaction may be rescinded by the payer prior to tis alternatively, there is not enough money in the account.

Third party systems for SMS payment A service for payment via SMS. A special message with the transaction id is sent by the payer. This transaction id is provided by the service and includes a checksum so that typing errors are minimized. The payment will be deducted from the payer's phone bill.

1.1.3 Offstage Actors

Municipality Have an interest in knowing what boat owners have what berth. This is important in te event of for example example accidents, thefts or similar.

The tax authority Have an interest in that the current regulations regarding the taxation of income associations are followed.

1.2 Use Cases

1.2.1 Authenticate

A person wants to be authenticated as a role. The person is authenticated and assigned a role.

Precondition: The person is not already assigned a user role.

Main scenario

1. The person wants to be authenticated.
2. The system asks for log in information.
3. The person supplies a user name and password
4. The system controls the username and password and authenticates the person as a user role (Member, Treasurer, Secretary) in the system, the assigned user role is presented.

Secondary Scenarios

The combination of user name and password is wrong

1. The system presents an error message and asks for log in information again.
2. Go to step 3 in main scenario.

1.2.2 Pay Membership Fee

A member wants to pay their dues. The fee and payment options are presented and Member proceeds with payment. The payment is recorded and its status is updated, a receipt is presented.

Primary Actor: User

Main Scenario

1. The member wants to pay their dues.
2. The system presents the calculation of membership and a total amount due.
3. The member chooses to pay via credit card.
4. The system transmits a transaction id and a total to third-party system for Credit Card Payment.
5. Third Party Credit Card Payments System for process the transaction and reply with a positive result.
6. The system updates the Member's pay status to paid and presents a receipt of the transaction.

Secondary Scenarios

The member chooses to pay via invoice

1. The system presents an invoice for printing with a unique invoice id, boat club account number and amount due. The amount due is the total membership fee plus an administrative service charge of currently 45 swedish crowns.
2. The user confirms the invoice.
3. The system updates the member's payment status to invoiced and presents that the payment status changed.

The member chooses to pay via SMS payment

1. The system contacts the Third Party System for SMS payment and enclose the sum total of the transaction.
2. The third-party systems for SMS payment replies with the SMS message to be sent to process the payment.
3. The system updates member's payment status to waiting. SMS message to be sent is presented.

The member chooses to pay via third party systems for Direct Payment

1. The system transmits a transaction id and a total to third-party systems for Direct Payment.
2. Third Party System for Direct Payment process the payment and reply with a positive result.
3. The System updates the Member's pay status to paid and presents a receipt of the transaction.

1.2.3 View Member Profile Information

A member wants to view their member data. Membership details are presented including registreade boats booked berths and payment history.

1.2.4 Register Boat

A member wants to register a new boat and the boat's data. The boat is registered, the membership fee is updated and a confirmation appears.

Main Scenario

1. A member wants to register a new boat.
2. The system asks for Boat Details.
3. The member input the boat's size and type (sailboat, motorsailer, powerboat, kayak/canoe, other) and an optional image of the boat.
4. The system presents the information for the boat to be registered, including cost of berth.
5. The member confirms the correct information.
6. The System registers the boat and assigns a berth using the current rules for berth assigment, updates the membership fee and presents a confirmation.

Secondary Scenarios

The boat is registered during the "offseason" (October 1 to December 31)

1. System assigns no berth and the membership fee for the current year is unchanged.

The boat is registered in pre-season (January 1 to April 1) when no berth assignment has been made yet.

1. System assigns no berth. This is done by the Secretary before the start of the season.

1.2.5 Remove a Boat

A Member To remove one of their registered boats. The boat is removed and a confirmation appears.

1.2.6 Change a boat

A Member wants to change a boat's data. The boat's data is updated and a confirmation appears.

1.2.7 Send Payment Reminder

The treasurer lists and send a reminder to those members who have not paid enough dues for a calendar year.

Main Scenario

1. The treasurer wants to send a reminder to those members who have not paid enough dues for a previous year and indicates the year considered.
2. The system lists the members who have a debt to the boat club during the year.
3. The treasurer choose to print an invoice for each member (see 2.3.1) in the invoice also includes a reminder fee of 25% per year behind schedule.
4. The system prints an invoice per member and records that the member received a payment reminder.

1.2.8 Assign Berths

The Secretary wants to assign this season berths. The system assign and book berths in accordance with applicable rules and update member information.

Main Scenario

1. The Secretary wants to book this season berths.
2. The system presents a proposal on the allocation under the current rules, the available berths and the previous year's allocation.
3. The Secretary approves the proposal.
4. The system assigns moorings for members' boats according to the proposal.

1.2.9 List Berths

The secretary chooses to list all berths. A list of unbooked and booked berths with information about the boat and member is presented.

1.2.10 Manage Calendar Events

The Secretary wants to add, delete or change a calendar event. Boat club calendar is updated.

1.2.11 List Calendar Events

A Member wish to list calendar events in a certain time interval. A list of calendar events are presented with a short title and start date.

1.2.12 View Calendar Events

Member to view all details for a particular calendar event. Calendar event is presented including start and end dates.

1.2.13 Update Payment Status

The Treasurer would like to update the status of payments. The system will contact third party services and check the status of payments and update these.

1.2.14 Update Invoice Status

The Treasurer wants to change a Member's payment status to paid. The system updates the Member's payment status to paid.

2 Analysis

The analysis is based on the given problem description1. All work has been done by Jonas Sjöberg, without collaborators.

2.1 Analysis of Requirements for Grade 2

This analysis delimits the domain model to the following requirements:

- 1.2.1 Authenticate
- 1.2.4 Register Boat
- 1.2.5 Remove Boat
- 1.2.6 Change Boat
- 1.2.8 Assign Berths
- 1.2.10 Manage Calendar Event

- 1.2.11 List Calendar Events
- 1.2.12 Show Calendar Event

2.1.1 Domain Model for Grade 2

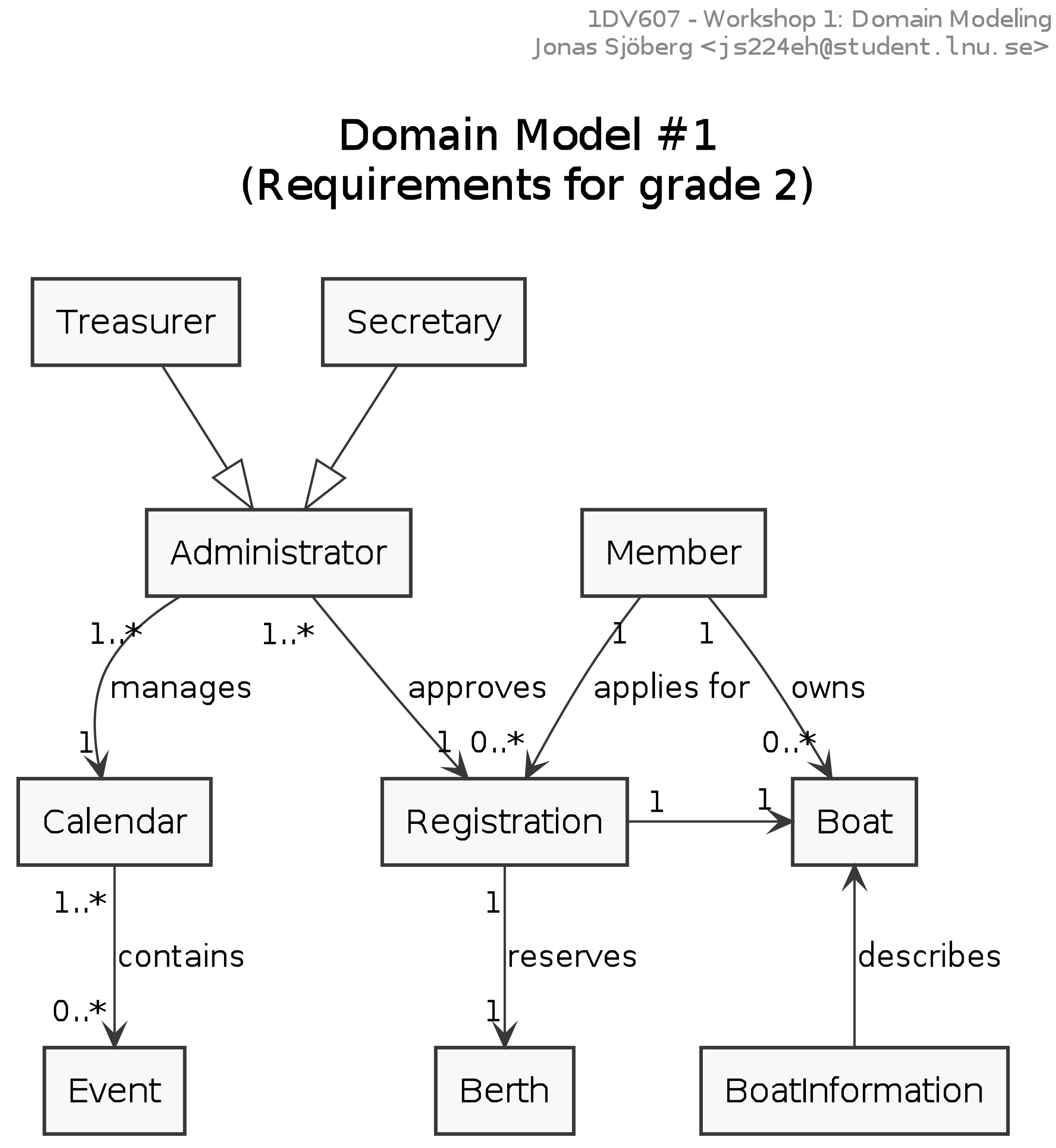

Figure 1: UML Domain Model for Grade 2

2.1.2 Notes on the revised domain model

The first domain model included the concept of "AccessControl" as an attempt to illustrate that the roles of "Secretary" and "Treasurer" are assumed to be equivalent and are both treated as a form of “Administrator”.

However, as fellow students have pointed out in peer reviews, and as I’ve come to better understand the revelant modeling concepts; this class entity has been removed for this version.

I’ve stuck with the decision to reduce the three different roles to just two. The concept of roles dictated by the requirement documents should not be a hard limit, as the role descriptions esentially are arbitrarily chosen names for arbitrarily chosen rules. Even though there are conventions, modeling the user roles exactly as stated in the requirements would be counter-productive.

The administrators have elevated privileges, as compared to the “Members”.

The peer reviews also objected to the lack of attributes in the classes, the “Registration” class in particular. But I feel that adding an attribute to this class would not improve the models readability, nor would it provide any improvements of the conceptual system design. Adding some arbitrary field to symbolize the unique identifier that maps entities related to a registration to each other would confuse the matter.

This conceptual model does not contain this information, because it should not be relevant at this level of abstraction.

2.2 Analysis of Requirements for Grade 3

This analysis includes all of the requirements stated in the problem description 1.

2.2.1 Domain Model for Grade 3

2.2.2 Notes on the initial domain model

The biggest change is the introduction of a payment service provider.

Most of the other requirements involve presentation of various data or managing data in certain ways. This is restricted to the different user-levels, administrators should be able to manage invoices, etc. These requirements could be modeled with “description classes”, which have been omitted for clarity.

2.3 Analysis of Requirements for Grade 4

For this analysis, I’ve chosen to look at my open source project “autonameow”[2]. I’ve tried to model the system of storage and retrieval of arbitrary data. However, this system is already implemented and very complex — the given model is incomplete and includes many conflicting levels of abstraction.

2.3.1 Notes on the main renaming actions in “autonameow”

The application collects as much data as possible from files. This includes the file name, filesystem information, embedded metadata for various format, textual contents, optical character recognition for images, etc.

This data is then used to construct file names and rename files, using some heuristics and user-control.

A highly simplified view of the main file remaing process is modeled in Figure 3.

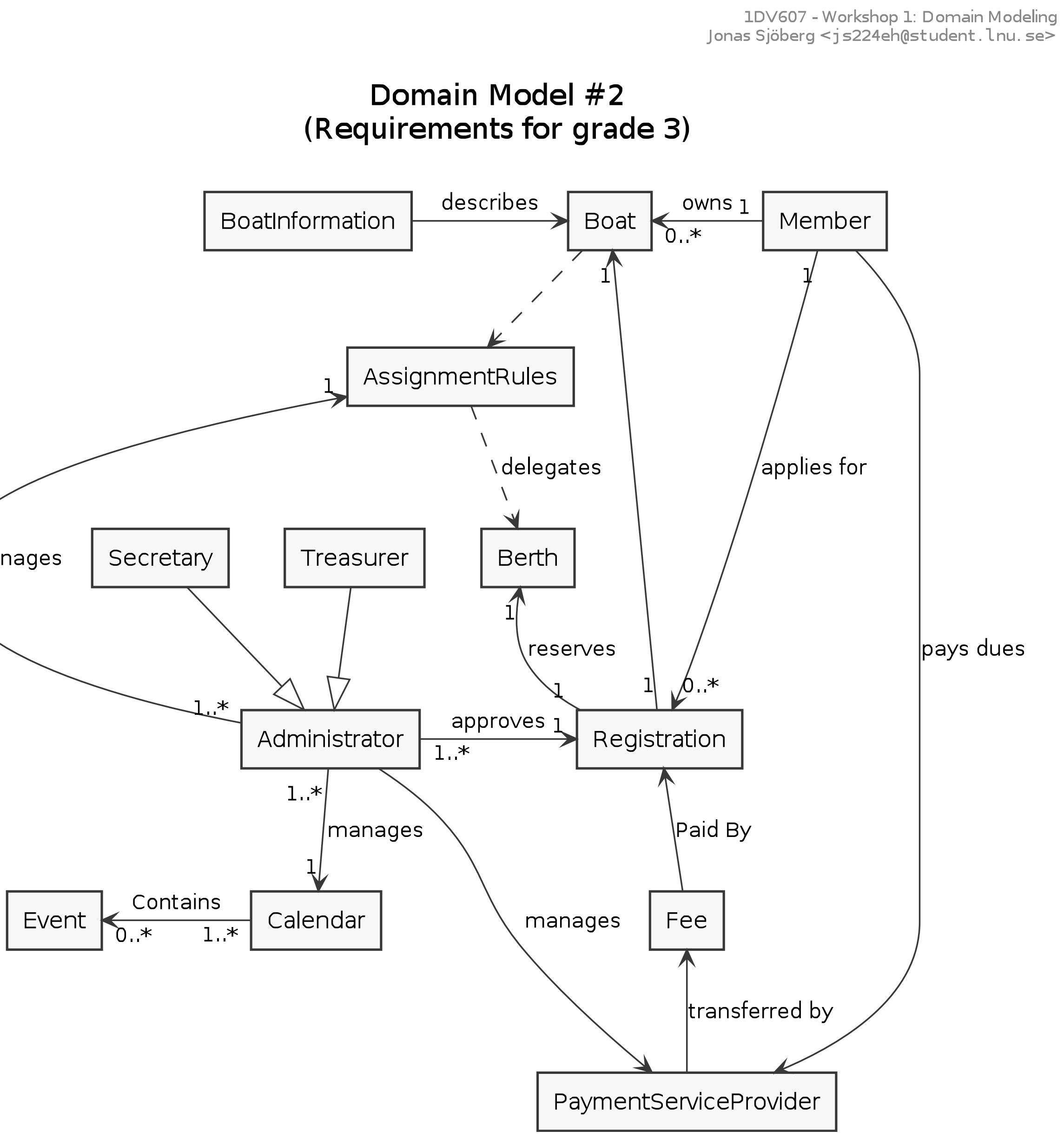

Figure 2: UML Domain Model for Grade 3

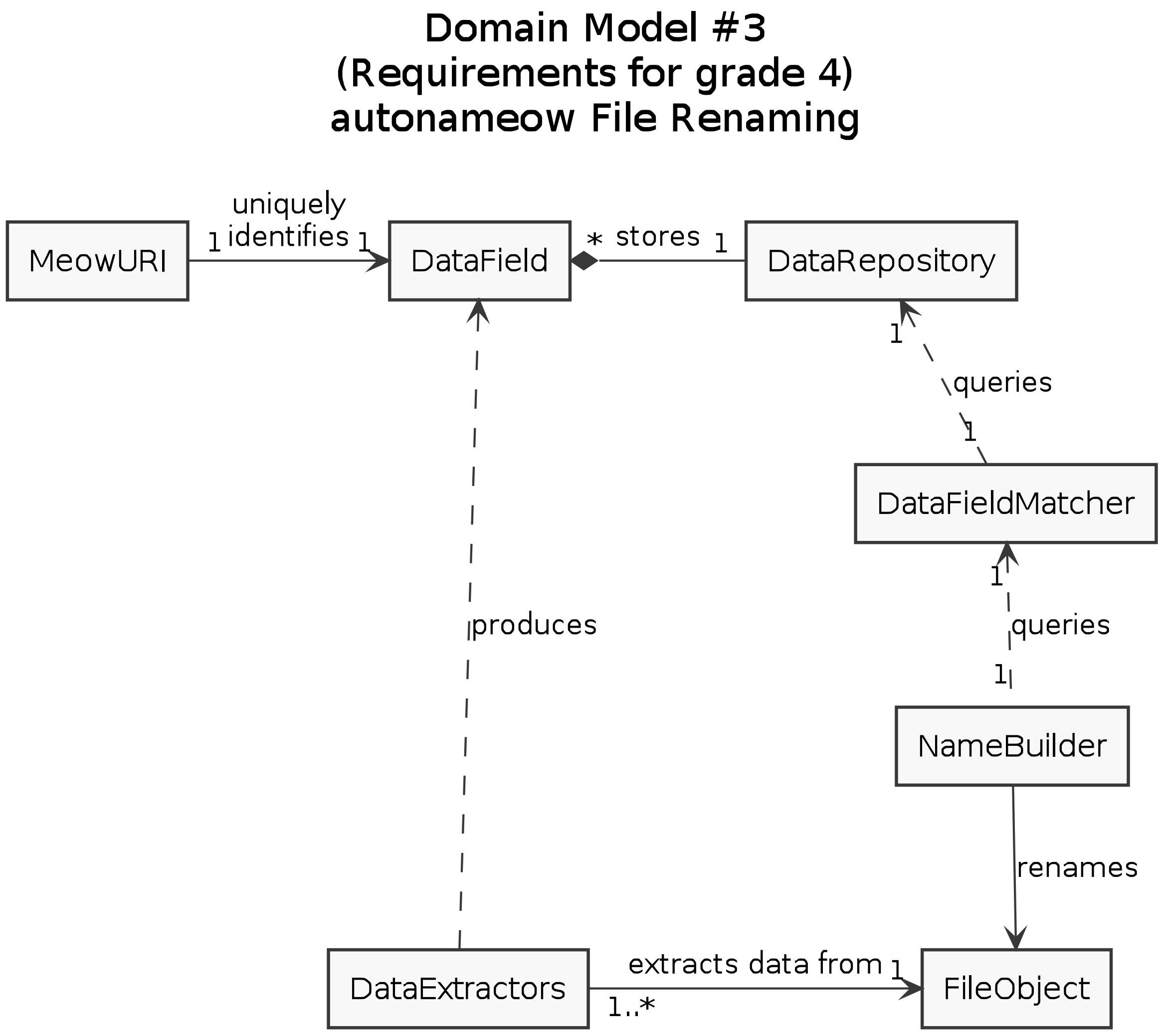

Figure 3: Domain Model of "autonameow" file renaming

2.3.2 Domain Model for Grade 3 – file renaming in "autonameow"

2.3.3 Notes on the "autonameow" data model

Like a library or other storage system, stored data is given an identifier, called a "meowURI"; where "URI" is an abbreviation for Uniform Resource Locator. These URIs provide the main means of querying and storing data.

A library might use ISBN- or EAN-numbers to identify specific items. "autonameow" also uses several identifiers, among these are ISBN-numbers, which should be combined and weighted using some heuristics to produce a suitable final result for any given query. The user might want to include a creation date in the file name, the program must then decide which of all the available date/time-information is most likely "correct".

2.3.4 Domain Model for Grade 3 – data storage in "autonameow"

This model tries to capture the interaction between components. In practice, I find that the interaction is too difficult to model properly at a level of abstraction that would provide any additional insight, over what the source code already provides.

A simplified model of the internal data storage is shown in Figure 4.

I've researched this topic for a very long time. A long list of references and related projects or produces is included with the "autonameow" documentation[3].

Many models have proven incorrect or too complex. I will keep iterating and redesigning the system until it meets the (ever changing) requirements.

I'm currently in the proces of redesigning the "Repository" to better handle contextual information provided with a data item. Extracted metadata itself often get additional metadata added on, containing the data source, probabilities, primitive data types, etc [4]. This has led me to research data repositories — where methods of retrieval, storage and presentation are all abstracted. I'm still reading up on how this works.

3 Conclusion

I have really struggled with understanding this very strange way of designing software. Coming from a background of mostly programming 8-bit PIC microcontrollers and designing discrete logic, the all-encompassing object-orientedness in the methodology does not come easy.

Everything does not belong in a class. Orienting the programming around objects, as in *Object-Oriented* Programming; just seems plain wrong and limiting. This kind of UML-modeling and OO-design just does not seem very useful when implementing systems in a procedural style, with only a handful of classes.

The abstract model often falls apart and become a tangled mess when attempting to model the actually interesting parts, often "cross-cutting concerns".

Most interesting problems and behaviours in non-trivial systems, where one might benefit from modeling tools to aid in design and understanding — are cross-cutting[5].

Looking ahead, I will look at techniques to apply some of the object-oriented modeling strategies to my way of thinking about programming, computing and complex systems. Especially how this all relates to a non-object-oriented approach.

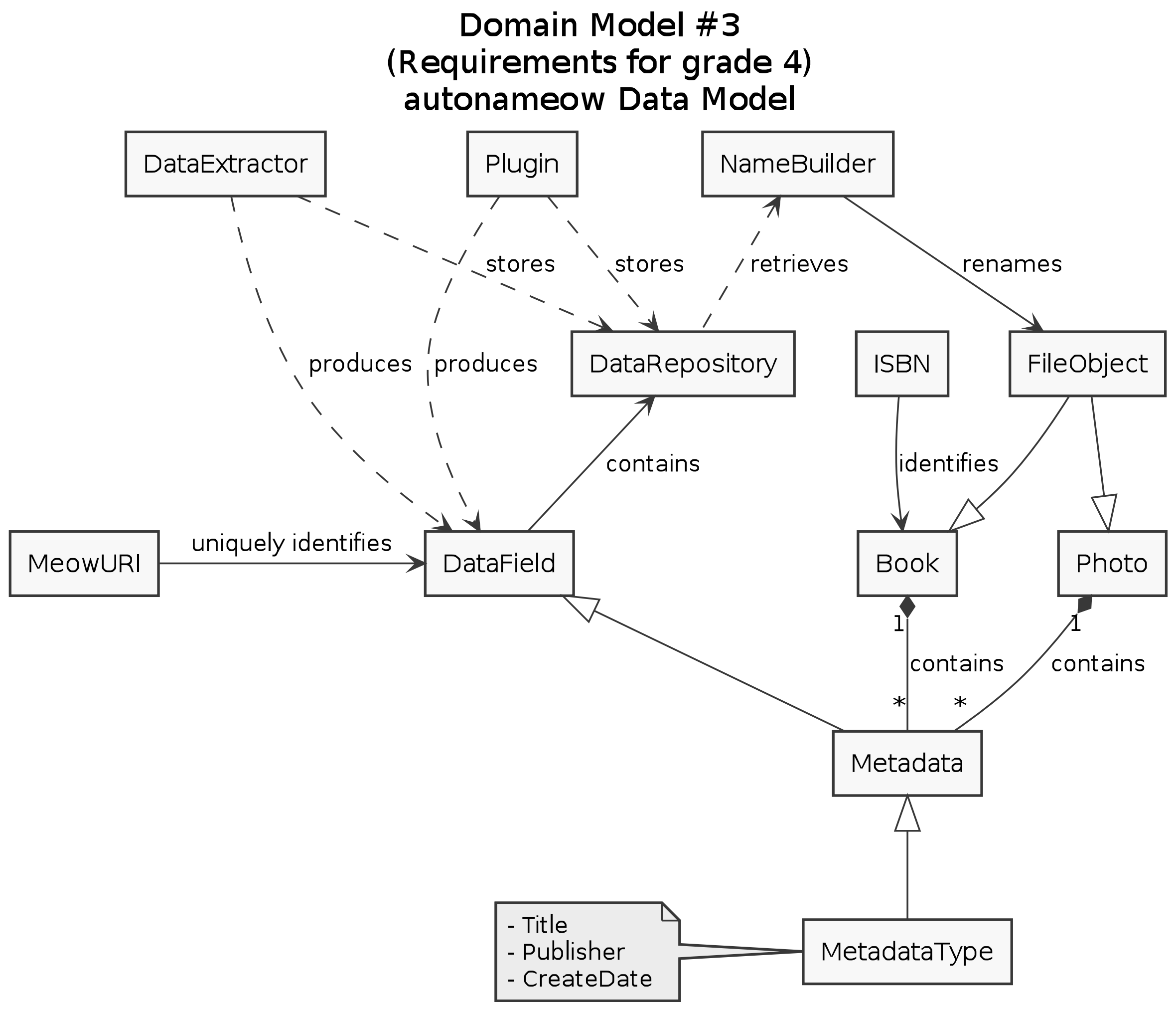

Figure 4: Domain Model of "autonameow" data storage

References

[1] T. Ohlsson, *Workshop 1 — domain modeling*, [Online; accessed 31-Aug-2017], 2017. [Online]. Available: `https://coursepress.lnu.se/kurs/objektorienterad-analys-och-design-med-uml/workshops-2/workshop-1-domain-modeling/problem-description/`.

[2] *Autonameow source repository*, [Online; accessed 05-Sept-2017], 2017. [Online]. Available: `https://github.com/jonasjberg/autonameow`.

[3] *Autonameow documentation*, [Online; accessed 05-Sept-2017], 2017. [Online]. Available: `https://github.com/jonasjberg/autonameow/tree/master/docs`.

[4] G. Alemu and B. Stevens, *An emergent theory of digital library metadata: Enrich then filter*, ser. Information Professional Series. Elsevier, Aug. 2015, ISBN: 978-0081003855.

[5] J. Blow, *Software quality at the csua gm2*, [Online; accessed 18-Sept-2017], 2016. [Online]. Available: `https://www.youtube.com/watch?v=k56wra39lwA`.