Software Architecture

Chiel Peters, Omar Pakker, Mary Gouseti, Cindy Berghuizen

September 29, 2013

1 Introduction

This document provides the architectural decisions together with the learning process. In the first section, multiple definitions of an architecture and our own view of an architecture is given. Then the stakeholders are identified together with their concerns. The third section contains multiple viewpoints on the architecture of the FlyWithUs system. Finally, Appendix A contains the results of the domain research performed early in the designing process and Appendix B contains the design decisions more formally.

2 Definition of Architecture

First four definitions of software architecture are stated and their differences and similarities are discussed. Second, our own view on software architecture is given.

2.1 Definitions on architectures

1. Software Architecture is the set of structures needed to reason about the software system, which comprise the software elements, the relations between them, and the properties of both elements and relations.[1]
2. Software Architecture is an abstract system specification consisting primarily of functional components described in terms of their behaviours and interfaces and component-component interconnections[2]
3. Software Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.[3]
4. Software architecture is the study of the large-scale structure and performance of software systems. Important aspects of a system's architecture include the division of functions among system modules, the means of communication between modules, and the representation of shared information.[4]

These definitions share the same fundamental idea that software architecture is a set of structures/components/modules including their properties and the relationships between them. However, each of the definitions highlights certain aspects of architecture more than the others.

- In [1] and [2] the focus has been put on the end result. The architecture is defined as the components themselves (their properties/interfaces) and the relationships between them (interconnections).
- The definition stated in [2] lays a focus on the functional components of a software architecture. With words like functional components, behaviours and interfaces underlines the importance of declaring the way the components interact with each other.

- The definition of [3] is limited only to the systems components and their functionalities and connections but also includes in the definition the process of designing and maintaining an architecture. It explicitly states that the design process and the evolution are part of the architecture itself.
- The last definition ([4]) presents architecture as the study a system's structure. As a result architecture contains the modules and the connections between them that build the system. This also becomes clear from the title of the article *Studying Software Architecture Through Design Spaces and Rules*.

2.2 Our view on architecture

As the definitions mentioned above, we also agree on the point that a software architecture is defined as a set of components, their properties, behaviours and the relations between. However, we think that architecture is not only the structure of a software system but also contains the main design decisions that would explain why the system consists of these components and include the stakeholders' quality attributes that lead the architects to those decisions.

Our definition of software architecture would be: Software architecture is the structure of a software system along with the decisions that lead to this structure. To elaborate architecture contains the components by which the system is build up, their functionality, the relations between them and the reason of their existence meaning which requirement of the stakeholders or quality attribute they accomplish.

3 Stakeholder Concerns

In this project there are four different stakeholder :

1. The Initiator
2. AirFrance - KLM
3. The Dutch Government
4. EU Claim

Each of them had different concerns within this project which are stated in the next subsections.

3.1 Initiator

The initiator is the one who started this project. He wants FlyWithUs to be a success and become the number one rating site people will go to. He want to be able to collect data from social media, other rating websites, news, weather and every other source that can be of importance. This data an then be used to provide the airlines with powerfull statistics and can give users a final rating. Users that make use of FlyWithUs have to be able to post reviews and ratings on the website and search for them. What makes FlyWithUs unique is the fact that airlines can get in touch with the users by sending them messages.

3.2 AirFrance - KLM

This stakeholder wants to have a reporting tool. With the tool he has to be able to see what recent reviews have been posted about his airline. Also, AirFrance - KLM wants to see statistics and be able to see what causes a sudden decline or increase in the rating. Furthermore, AirFrance-KLM wants to be able to enter flight information and by doing so influence the weight of review. This means that the weight of a rating has to be less when bad ratings are due to for example environmental issues (bad weather etc.) and have nothing to do with the airline companies services.

3.3 Dutch Government

Privacy is an important issue for the Dutch Government. The server needs to be hosted in the Netherlands so FlyWithUs will be led according to the Dutch Privacy Law. Also, the Dutch Governemnt would like to see the project to be a "Green IT" project.

3.4 EU Claim

EU Claim wants to make certain that the privacy if the user is guaranteed. Furthermore the airlines have to behave on the website and do not mess with the results or bribe the users. Fairness is thus also an important issue to this stakeholder.

4 Viewpoints

This section contains three viewpoints on the architecture. These viewpoints were mainly due to the stakeholders concerns. The privacy view provides EU-Claim and the Dutch government the information related to their main concerns. The data-flow indicates how the system will be able to handle all the data from external and internal sources which is one of the biggest challenges in designing this architecture. The last view shows the functional requirements of the system of all the available users.

4.1 Privacy viewpoint

- Related stakeholders: Dutch government, EU Claim
- Related Concerns: Privacy of the user

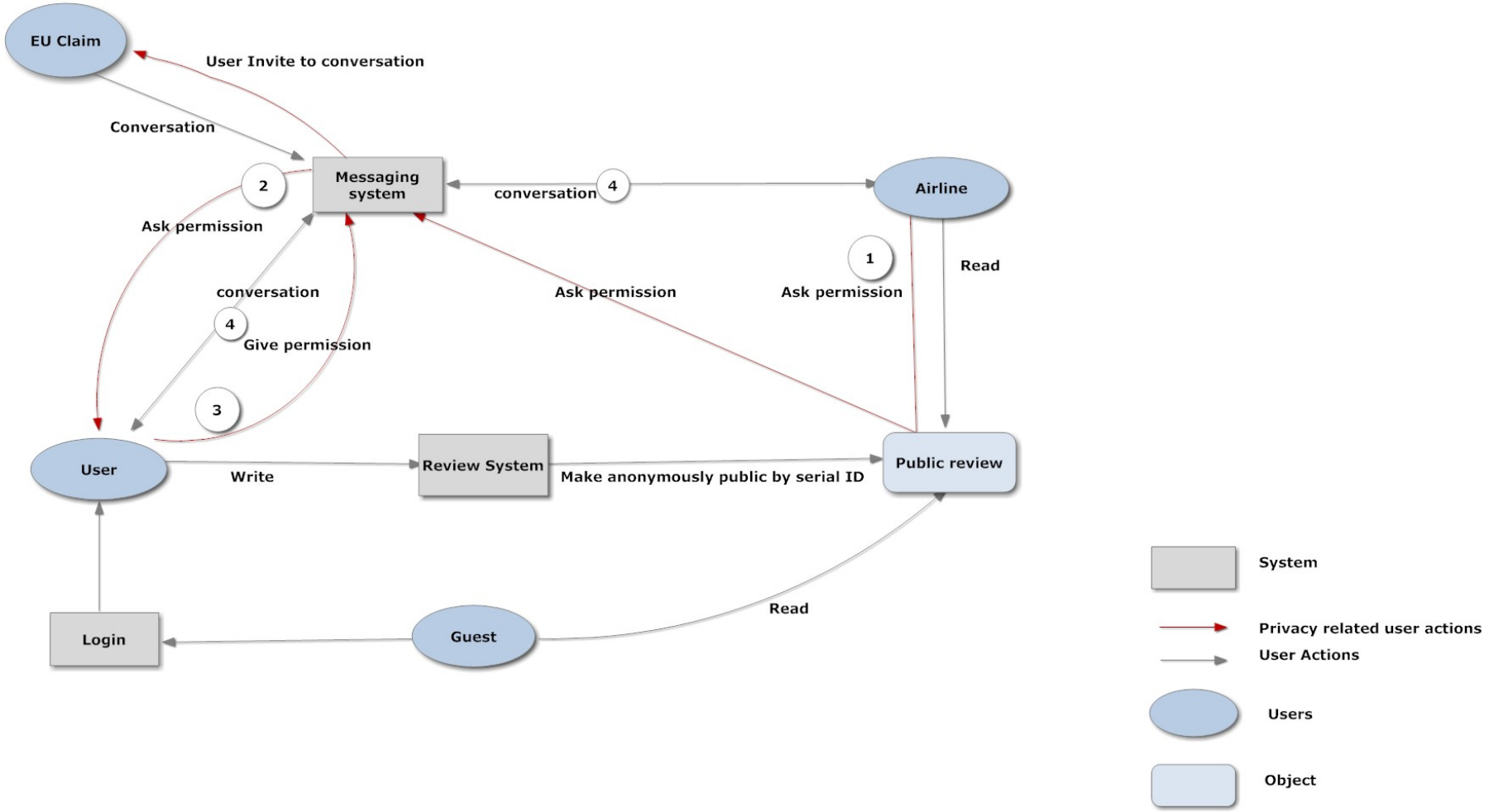

Figure 1: Privacy viewpoint

The privacy viewpoint in figure 1 shows how the privacy of the user is guaranteed in the system. A user can write a review that is shown to the public anonymously. The review will have a serialID instead of the user name so the user will stay unknown to the public.
An airline should be able to have a conversation with an user if the airline wants to elaborate on a specific case or wants to adress the user personally. If the airline wants to get the user's information for a conversation it should ask permission of the user. When a user accepts to have a conversation with an airline the user's contact information is visible to the airline so they will be able to have a conversation. If the user is not satisfied with how the airline company responds on requests or remarks in the conversation, the user is able to invite EU Claim to the conversation so they will be able to see the conversation and decide to act upon it.

4.2 Data Flow Viewpoint

- Related stakeholders: KLM, Initiator
- Related Concerns: Data Integrity, Performance, Scalability
- Related design decisions: How is the data split up into multiple databases?; How do we handle large data-sets?; Recalculate or Combine rating?; Incremental or Time-interval update?; Flight-Information

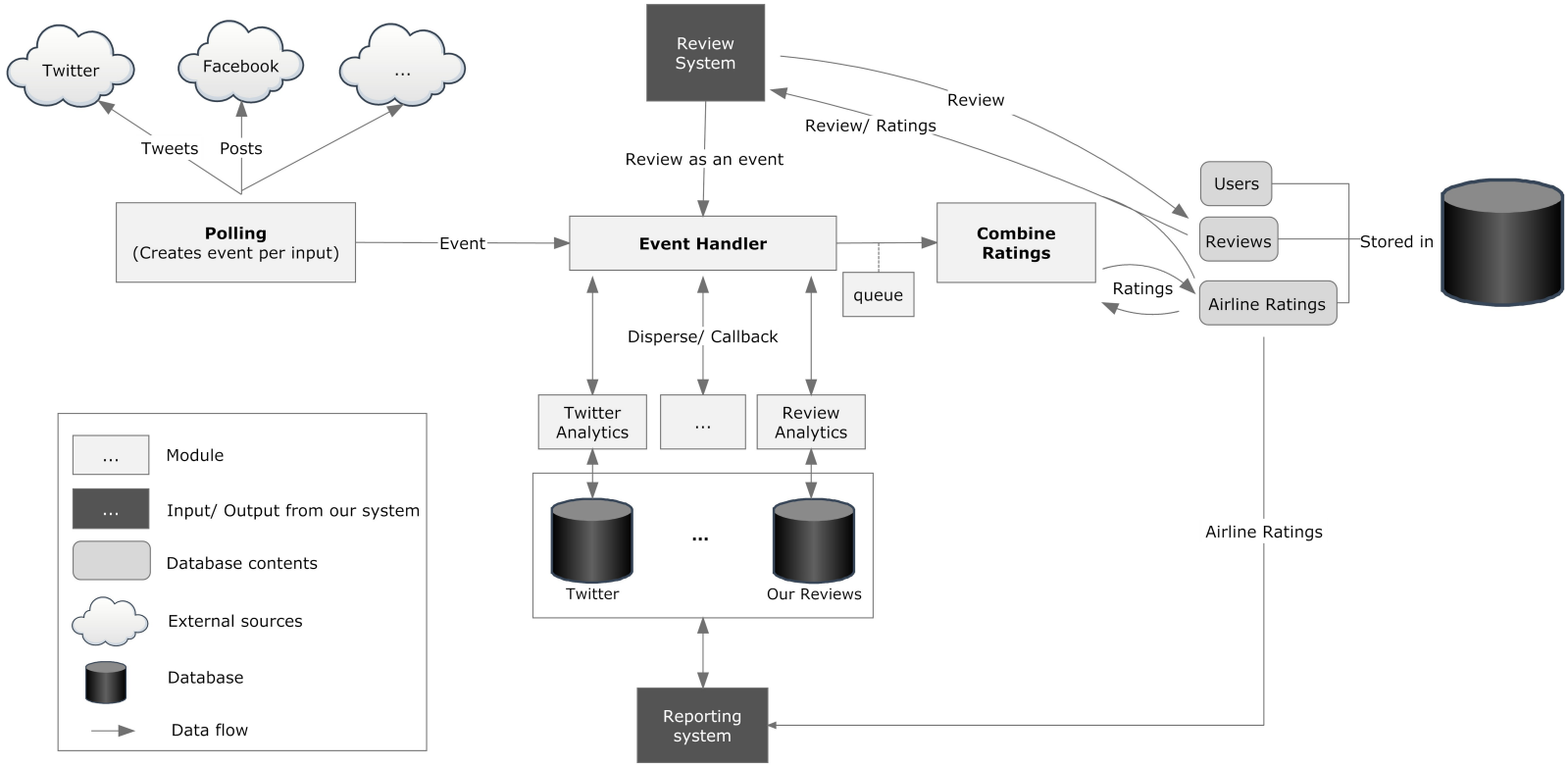

Figure 2: Data Flow Viewpoint

This Data Flow viewpoint 2 illustrates the flow of data within the system. In this picture the review and reporting system are shown as black boxes to illustrate that their functionalitity is of no importance in this picture. In the related design decisions it was argumented that combining the rating with incremental updates was best for performance and scalability. Note that in the flow only the back-end of the system is given, because of the importance of handling the large data sets. The viewpoint indicates the following flow of data:

1. **Polling** The external sources are polled a fixed time-schedule an all the individual reviews (e.g. tweets, posts) are sent off the event handler.

2. **Event Handler** The event handler send every review into their own specialized pipe according to source. This creates paralism and allows for horizontal scaling. After the event handler has sent of the reviews it waits for the callback from the analytics. After the callback is given it sends off the analyzed data into the queue for combining into a final rating.

3. **Analyzing** Every source has its own analyzer. This way the unique format of the source can be maintained. Once the reviews are analyzed and contain a rating there are stored in specialized databases, but also given back to the event handler.

4. **Combine** Once the reviews are analyzed and contain a rating they can easily be combined by a weighting algorithm. In the design decision "Recalculate or Combine rating" it was argumented that combining the new reviews with the old ratings was better for performance. Therefore the combine also obtaines the old final ratings from the database and updates this with the new reviews.

Based on several design decisions the above design was created. The above design focusses on performance and scalability. This design splits data-mining and stats from the main website data. This allows for performance gain by gaining the ability to use specialized databases. It also allows us to improve scalability by only having to scale either data-mining and stats or the main website database.To balance the system load, we use a queue for the data. This allows you to use less powerfull servers as you don't have to handle spikes in resource requirements.

4.3 Functional Viewpoint

- Related stakeholders: KLM, Initiator, EU-Claim
- Related Concerns: Users of the system, Available functionality to each user group, Grouping functionalities
- Related design decisions: EU-Claim can see private conversation after invitation, Online payment system (Discarded), Functionalities of Reporting System.

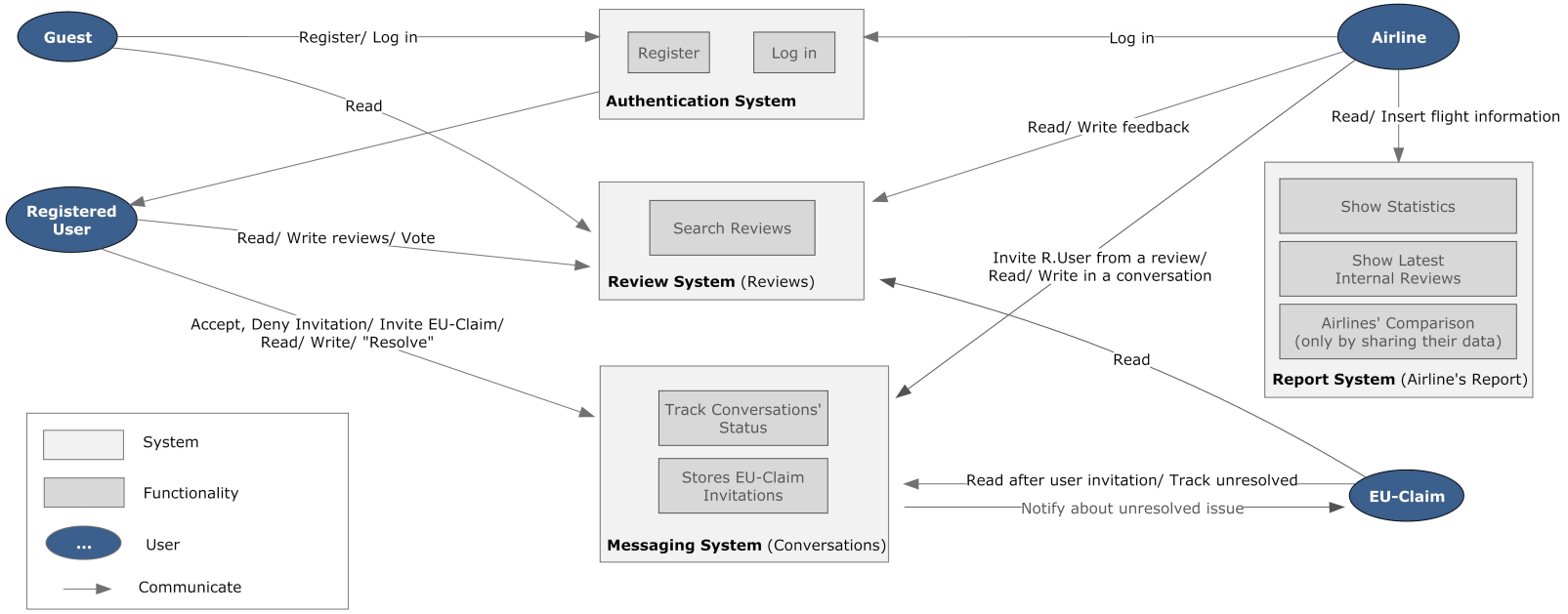

Figure 3: Functional Viewpoint

The functional viewpoint (Fig. 3 illustrates the users and the functionalities of our system. The functionalities that are based in the same entities of our system are grouped together to help the reader understand their connections. For example, all the functionalities provided only to airlines are included in a *report* as a result these functionalities are grouped in the *Report System*. Furthermore, the viewpoint also presents the availability of the functionalities to each user group and the communication between these groups through our system. For instance, a guest can register or log in to our system and become a registered user, or an airline can invite a user to a private conversation in order to resolve a potential complaint; then the user can accept the invitation and he/ her can also invite EU-Claim to monitor their conversation to ensure the integrity of this process.

A Domain Knowledge - Design Decision

In this appendix we will present the domain specific problems related to airline reputation management system. These provide a basis for making and finding the design decisions in Appendix B. To divide the workload four separate directions were defined. These directions are highly interconnected and might not appear as separate in the final design. The four directions are:

1. User Experience
2. Competition and Functionality
3. Data storage
4. Data Gathering

A separate section has been dedicated to each of the direction.

A.1 Domain Specific Problems Related to User Experience

A.1.1 Important Questions

The purpose of this section is to ask domain specific question related to user experience. The interface consists of all front-end systems that are directly in contact with the users. In order to get a better insight over the domain a market research was performed wth related airline websites. In this research the importance was placed on the users and what they are looking for in a rating site. The questions that are important to the interfaces which affect user experience are:

1. What types of customers are distinguished in the system?
2. What kind of interactions should each customer be able to perform?

A.1.2 Design Decisions

During our meetings with the stakeholders it was made clear that there are two types of customers:

1. Clients: In general, the people who write and read the reviews and ratings.
2. Business Clients: The airline companies. The airline companies want to see a general overview of their companies ratings and reviews with some additional features such as the reporting system (discussed in the next section)

An overview of the functionalities available to these groups of user is described below:

- Clients (Registered Users):
    1. They can write (*anonymously*), read and vote the reviews (Review System).

2. They can see and compare the ratings of different airline companies.
3. They can accept the invitation of an airline based on one of their reviews to a private conversation (Messaging System).
4. They can Invite EU claim to a private conversation.

- Business Clients (Airline Companies):

1. They can write feedback to a user's review.
2. They can invite a user to a private conversation based on a review in order to resolve his/ her complaint.
3. They can see the current ratings and/ or the latest (bad) reviews posted (Reporting System).
4. They can see statistics about their progress.
5. They can see their progress in comparison with other airline's progress (only if they share their information as well).
6. They can insert flight information and search through our databases to find patterns.

The fact that these two groups have clearly different functionalities results in two options concerning the system's interface, either two totally separate sites, or a general page whose content will change depending on the signed in user but the lay out will be the same.
In addition to these groups there two more:

- Guests: Unregistered users who can only search for airlines and read reviews and ratings.
- EU-Claim: EU-Claim can monitor a private conversation after a user's invitation. Furthermore, is notified if a conversation is not "resolved" for a specified period of time.

A.2 Domain Specific Problems Related to Competition and Functionality

A.2.1 Important Questions

The purpose of this section is to find the domain specific questions related to our functionality. To accomplish that, we searched in the internet to learn more about our competition and what they offer in order to adapt our system's architecture to focus on the functionalities that are going to differentiate us from them. The questions that help FlyWithUs understand the domain are:

1. Who is our biggest competitor?
2. Do we need to handle online payment?
3. How do the feedback and the messaging systems work?
4. How will the reporting system work?
5. How can we protect our results?
6. How do we ensure the users' privacy?
7. Which features of our review format are going to ensure users' privacy?

A.2.2 Design Decisions

Competition Our greatest competitor seems to be http://www.airlinequality.com/. This site contains reviews about 681 airlines and 725 airports and uses a star rating system called SKYTRAX. SKYTRAX has been used since 1999 and right now seems to be the only globally accepted airline rating system (is recognized as a global benchmark for airline standards). Fortunately, it does not provide feedback and reporting system. Consequently it is recommended to focus these systems.

Online Payment It was requested that the services to the airlines will be available after payment. The payment can either take place online or offline. In the first case, the stakeholders are going to have an extra cost (fee: a percent of every transaction or predefined + 10$ 25$ every month). On the other hand since the payment is addressed only to airline companies it is possible to do that offline through contracts, invoices etc.

Feedback & Messaging System The platform of FlyWithUs provides to the airlines the opportunity to respond to a review if they believe it is necessary. (Feedback system)
Additionally, an airline will be able to invite a user to a private conversation based on bad review in order to address an issue or to try to compensate him/ her. The user can choose if he/ she wants to continue and if he/ she wishes to provide his/ her information.
It is also significant to decide whether preserving the integrity of the system's messaging service is desirable. To accomplish that, EU-Claim could be able to monitor the private conversation. Although the user will be notified about this, it may still be against his/ her privacy. EU-Claim proposed this functionality to be available only if the user enables it and not by default and KLM representative and the initiator agreed as well.
Furthermore, it is desirable for such a conversation to have states for example resolved and pending, then a user can change a its state from pending to resolved. As a result EU-Claim can be notified by the system if a report has been unresolved for a certain period of time. Additionally, the state of a conversation allows us to calculate the resolved issues and include that result in our website in order to encourage people to use our system.

Reporting System The reporting system is a functionality only available to airlines. Each airline will be able to monitor its rating status through a report webpage which will comprise of the following components:

- Statistics concerning its progress through graphs and diagrams.
- Latest reviews or bad reviews from our site.
- Cross-reference flight information with the reviews stored on our system.
- Comparison between airlines (only if the airline shares her information also)

Prevent Aggregation Copyright The copyright of the data that will be collected should be further investigated according to the policy of each source. If the legality of harvesting data from the other sources is resolved and even if the data gathered are public, the collection of the data on our system can be protected. This can be achieved because even though the data are public a collection of them is copyrightable. Additionally, we can use a file called robots.txt to prevent data aggregation from specific crawlers. However, the effectiveness of this file lies entirely on the crawlers because they can ignore the file. It appears that there are no architecture relevant questions on this question.

Users' Privacy As it was mentioned in the section Feedback & Messaging Systems, users' privacy is an important issue. This issue was raised by three of the stakeholders Dutch government, EU-Claim and the initiator. A user's data and conversations with airlines are private, this means that these information are available only after user authentication and in the case of the private conversation available only to the airline of interest (maybe to EU-Claim as well if the user requests that).

Review's Format Since user's privacy is a big issue the review's format should allow anonymity. This is accomplished by omitting the name, username or any other author identifier from the public representation of a review.

A.3 Domain specific problems related to data gathering and Analysis

A.3.1 Import questions

The purpose of this section is to explain domain related problems related with this task. The data gathering is responsible for polling external sources and providing the data to the storage system. This implicates two important question which are further elaborated next:

1. How is the data from different external sources combined?
2. How is the combined data used in order to obtain a rating?
3. Is there any filtering process on the external data?

The external sources vary from airline companies sites to review sites to social networks. These all have their own way of inputting a review. For instance, Twitter does not have any format related to the review and only consists of lines text, while review sites already have a format in place that allows the user to rate some of the attributes (Food, timeliness, service etc.) on a given scale. The final application should be able to digest all these kinds of reviews and output the results on a scale . A separate , however still important, question relates to the quality of the data. The external sources may or may not have systems in place that assure the quality of a review. Since the data is not inspected

A.3.2 Design Decisions

Combining external sources The sources have their own unique format as explained earlier in the document. If the system where to parse this into a general format information might get lost or lot's of information is undefined. However the general format helps decrease the complexity of analysing, because that system does not need to account for all the unique formats. This decision requires further collaboration with the data storage as that needs to be able to handle the data.

Analysing external sources Not all external sources give a rating on a scale (e.g. Twitter) therefore these reviews need to be analysed first in order to be useful. There are multiple options available when analysing these results:

1. Drop all reviews that have no rating attached to them. This would lead to a huge data loss, but decrease the complexity of the system. The ratings can then easily be combined by computing a weighted average (or by another formula) of the individual ratings.
2. Transform the reviews without ratings by performing a sentiment analysis. After the analysis the ratings can be combined the same as in the other option.

Filtering Some reviews may not be considered useful, because of certain properties. The system could apply some business rules in order to filter those reviews out. The business rules governing these reviews are however currently unknown.

A.4 Domain specific problems related to data storage and Analysis

A.4.1 Import questions

This section lays out the domain related questions with data storage. In the earlier section analysis was inspected from a logical perspective. In this section the analysis is mainly inspected from a performance perspective. The system is responsible for storing several kinds of data both from the local system as well as data from external sources. This implies several questions:

1. What kind of data does the system need to store?

2. How big is the data from external sources?

3. How does the system read and write the different data types?

The system has to store information such as users, reviews, statistics, data from external sources, etc. Because of the variety of data types it is important to understand these data types and how the system utilizes this data.
The data categories defined are:

1. Users: The system requires a user management system to log in and out on the web application. This user data needs to be stored persistently. An important quality attribute related to this user data is privacy, because it holds sensitive information.

2. Reviews: The users should be able to read and write reviews on the website.

3. Data from data-mining: The polling module (discussed previously) supplies the system with data from different external sources (e.g. Tripadvisor, Twitter). Because of the large amount of data scalability is a key quality attribute.

4. Analysis results: The data from external sources is analysed to form a rating and calculate statistics for the airline companies. Performance is a key quality attribute here as these results are shown to a potentially large number of end users, as well as scalability as the statistical data grows fast over time.

To get an idea of the amount of data we get from external sources, we use the following number. There currently exist over 600 airline companies. To calculate values for 600 companies one specific company (KLM) is taken and the values are extrapolated over the other airline companies:

1. Twitter: There are 1.4 Tweets per minute on KLM. This would indicate that there are approximately 600 * 1.4 * 60 * 24 = 1.008.000 tweets per day on airline companies

2. Facebook: There are 101277 talking about KLM on Facebook. This would indicate that in total 60.766.200 posts are about airline companies

3. Tripadvisor: There are currently 470 reviews on KLM. This indicates that there are 282.000 reviews on airline companies.

The size of the data already shows that great care must be taken in the data-mining and analysis step regarding to performance and as new external sources may be added over time, scalability is also of great importance.

A.4.2 Design Decisions

One or more databases? The system has to handle data from multiple sources and different usage types. For instance, the data-mining and statistics require a lot of write commands as opposed to reads, whereas the main website data, such as reviews and users, requires more read commands as opposed to writes. For this reason the usage of specialized databases for each would increase performance and scalability.

How many databases? If more than one database is used than the data needs to be split up while still keeping the system maintainable. A this time we have several options:

1. Split up the main website data and data extracted from data-mining.

2. Split up the main website data, data-mining data and statistical data.

3. Split up the main website data and for each external source have a separate database containing the data-mined data.

How do we handle large data-sets? The data collected from external sources require a great amount of storage over time. In addition to that, all the incoming data has to be analysed to create new ratings and statistical data. For this data both performance and scalability are the most important attributes. By utilizing an event type system the load can be balanced and only write data after it has been analysed, thus decreasing the amount of reads and writes the database has to run. The implication is that data in the event handler/queue is not persistent to the database and may be lost in system failures.

Always recalculate ratings or combine with the known value? When new data is collected from external sources or when a review is made, the rating for the airline company needs to be updated. While a complete recalculation allows you to always change the weight of your external sources and reviews, it also means that over time more and more data has to be read from the database. Because of that the performance will deteriorate over time. To prevent this from happening, utilizing known values can greatly improve performance. By keeping track of the amount of tweets/reviews/etc received and the current rating, you only add the new input to those values by calculating the weight and the share it has on the total. By doing so you don't traverse the whole database thus greatly improving performance. If a recalculation is required this could always be initiated separately.

Incremental or time-interval updates? When new data is received, the rating and statistics need to be updated. This can either be done on a time-interval, such as once an hour, or using a queue and continuously update the data. To get the best performance and to safe on hardware cost, the latter is the most optimal solution. By spreading out the load, the machine does not need to handle high peaks and therefore you don't need to have a machine build to handle the peak when it happens. This means that the system required to run the updates can be run on a less powerful system thus saving cost. It also means that the data is always being updated and that the users get the most recent information possible.

Flight-data usage The stakeholder interest in flight-data was the decreasing of the weight of ratings during delays caused by things out of the hands of the company, such as bad weather, and to see a relation in the statistics between flight-data and the height of their rating. To ensure the integrity of the reviews and to improve the performance of the system, we advice against the use of flight-data to decrease the weight of reviews. By doing so we don't have to read from the database to get the flight-data to filter reviews thus saving resources and gaining performance. It also ensures the data integrity as it may be possible that flight-data is entered after bad reviews have already been added to the system. It is also the case that if a flight is delayed or cancelled the service from the airline company to accommodate the passengers is being put to the test. This should not be weighted less because the flight-data says the delay is due to bad weather.

B Formal Design Decisions

This appendix states the design decisions that were taken during the process of designing an architecture. Figure 4 shows the tree of the design decisions that will each be further elaborated on this section.

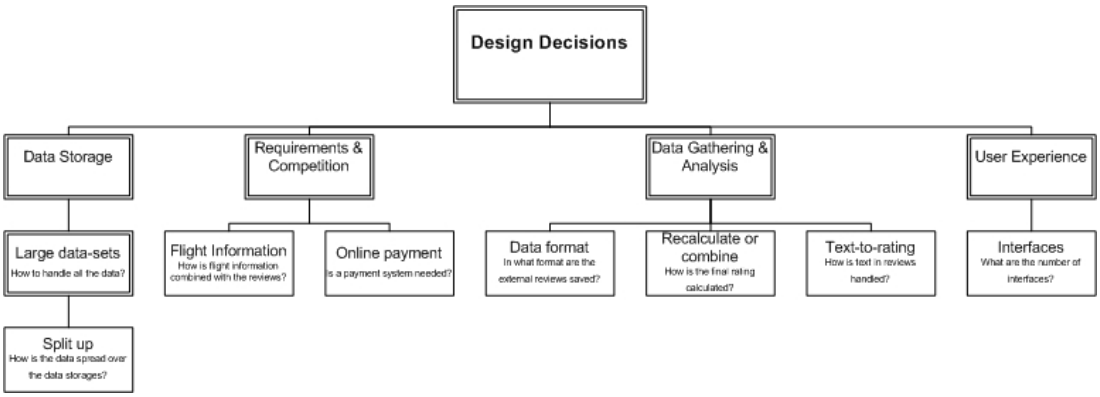

Figure 4: The design decision tree

Design Decision 1: Data format

Issue	The external sources have a unique format related to them. The system needs to be able to combine all these reviews into a final rating. What is the approach to unique format?
Decision	The unique format is kept in before it is analyzed. The analyzing step converts the unique format into a rating on a category (e.g. food, timeliness) which the system can use and combine
Status	Decided
Group	Analysis of data;
Assumptions	The data from external sources cannot easily be combined because they all have a significantly different format.
Constraints	None
Positions	Keep the unique format before analyzing the data. Parse the unique format from external sources into a general format the analyses module can use.
Argument	The external sources contain different layouts parsing them into a general format would either lead to data loss or lot of undefined fields. However the analyses module does not need to account for different kinds of data which decreases its complexity. The potential loss of data (and maybe meta-data) would however decrease the effectiveness of the analyses module. Therefore the decision was made to keep and store the data in a unique format.
Implications	The storage system needs to be able to handle the data from external sources. The analysis module needs to handle multiple lay-outs of the external sources.
Related decisions	
Related requirements	

Design Decision 2: Flight Information

Issue	It may occur that a flight is delayed due to external factors outside of the airlines responsibilities (e.g. weather). A bad review may affect the airlines rating which may not be considered a desirable effect.
Decision	The status is still pending awaiting conversation with the involved stakeholders.
Status	Pending
Group	Analysis of data;
Assumptions	Users will write reviews even if factors are outside of the airlines responsibilities.
Constraints	None
Positions	Cross-match flight information from airlines into the rating algorithm to avoid the influence of those reviews towards the rating Let the airlines search for reviews based on flight information, but not change the rating algorithms. Leave the problem out the scope of the system
Argument	If airlines can influence the rating algorithms than this greatly decreases the transparency of the system. This problem applies to each airline so it does not decrease the fairness of the system. The second option gives airlines the ability to spot the influence of those bad reviews, but not alter their influence. This keeps the system transparent.
Implications	The implications are given after the decision is taken.
Related decisions	
Related requirements	

Design Decision 3: Interfaces

Issue	The airlines and customers use different services on our systems as a result they will have different options on their interface. This can be accomplished in two ways, either they can have entirely different interfaces or use the same lay-out.
Decision	Two different external sites.
Status	Pending
Group	User Experience;
Assumptions	None
Constraints	None
Positions	Have one website, when logged in the interface changes depending on the type of user. Have two different log in pages so the airlines will never come across the regular user website and vice versa.
Argument	The second option is more appealing to the airlines, beause they have there own unique landing page. This also limits the users knowledge of the airlines and might help any security issues related to users trying to login as airlines.
Implications	The implications depend on the decision that remains to be taken.
Related decisions	
Related requirements	

Design Decision 4: How are large data-sets handled?

Issue	Data collected from external sources requires a great amount of storage over time. It also needs to be analyzed and have statistics generated/stored.
Decision	Event type of dispatching.
Status	Decided
Group	Data storage
Assumptions	The number of incoming data sets is that of a big data type.
Constraints	None
Positions	Pipeline model Event type of dispatching
Argument	In a pipeline model each step (Storing, analyzing, Storing statistics + rating) is performed in a sequence. Every review from every external source goes through the same system pipe. This is bad for scalability reasons as more and more reviews would clog up the system. The event type of dispatching can disperse reviews through a channel depending on their source. This creates paralism in the system and is more easily scalable. By collecting the data and sending it to the analyzers, FlyWithUs can generate the rating and score before storing it all. This way there is no need to read and write twice from intermediate storage which improves performance.
Implications	An event handler needs to be developed with clear contracts towards the other components within the system.
Related decisions	How is the data split up into multiple databases?
Related requirements	

Design Decision 5: Online Payment

Issue	The airlines should pay FlyWithUs for using the services this can be done online and offline (automatic based on contract).
Decision	Payment will be done automatic.
Status	Decided
Group	User Experience; Requirements
Assumptions	None
Constraints	None
Positions	Online payment via for example Ideal. Automatic payment by the airline based on a contract.
Argument	In order to support online payment the database needs to be ACID which will restrict our options for databases. The AirFrance-KLM stakeholder informed us that companies are used to paying automatically, through bank, and preferred this over using Ideal.
Implications	No payment system needs to be developed.
Related decisions	
Related requirements	

Design Decision 6: Recalculate or combine rating?

Issue	When receiving new information from external sources or our own website, the rating for the airline company needs to be updated. The rating can be recalculated by using all the historical data (full recalculate) or it is possible to combine only the new information that is not in the old rating and combine them (combine).
Decision	Combine the old rating with the new information (reviews) to update to a new rating
Status	Pending
Group	Data Analysis
Assumptions	It is possible to keep quickly keep count of the new reviews that are not within the current rating.
Constraints	None
Positions	Recalculate the whole rating Combine the old rating and new information
Argument	Recalculating the whole rating is a performance expense operation and leads to a lot of duplicative steps. This is especially true when the number of reviews (external and internal) starts to grow. However in the case of combining, if a cross match needs to occur between flight information and reviews then the flight information needs to be available before the reviews come in. This is because once combined it is included in the final rating and cannot be reverted back.
Implications	If the choice is made for combining than the flight information system needs its information imported before any reviews come in.
Related decisions	Incremental or Time-interval updates; Flight Information
Related requirements	

Design Decision 7: How is the data split up into multiple databases?

Issue	How many databases gives a good maintainable system and allows for the best performance for all the different database usages?
Decision	Split between data-mining (external and internal reviews statistics) and website data (users, conversations , ratings)
Status	Decided
Group	Data Storage
Assumptions	At the very least there is website data, data-mining data and statistical data.
Constraints	None
Positions	All data is contained within one large database. Split between data-mining (external and internal reviews statistics) and website data (users, conversations , ratings) Split between data-mining (external reviews), statistics on (external) reviews and website data (users, conversations , ratings)
Argument	Keeping all the data in one large database makes it very hard to scale and generally creates performance issues related with the amount of incoming data. The third option leads to a greater number of databases. However the statistics are heavily related to the reviews themselves (data mining) splitting these two up creates fragmented data.
Implications	There is a need for multiple databases that have some relation to each other. These relations need to be clearly defined in order for the system need to completely get out of synch.
Related decisions	How do we handle large data-sets?
Related requirements	

Design Decision 8: Text-to-rating

Issue	A lot of reviews contain text or are text-only. How is the text within reviews used in order to form a rating on a given scale?
Decision	The system needs to be able to handle text-only reviews. Therefore a sentiment analysis must be done on the text in order to transform it to a scale
Status	Decided
Group	Analysis of data; Data gathering;
Assumptions	The reviews from external sources contain text or are text-only. This is assumption is based on the observations in social media (e.g. Twitter, Facebook)
Constraints	None
Positions	Drop all the text within reviews and only use available ratings from external sources Perform sentiment analysis to transform text to a rating.
Argument	Dropping all the text within reviews leads to a great loss of information and data. Although it would the most objective solution the loss of data is considered to be too big of a loss. Therefore it was decided to review the text as well.
Implications	Because the system needs to handle with text a sentiment-analysis needs to be done. This requires additional functionality to the system.
Related decisions	Data format
Related requirements	NA

References

[1] Bass et al. *Software Architecture in Practice*. Addison Wesley, Boston, 3rd Edition, 2012.

[2] Hayes-Roth, F *Architecture-Based Acquisition and Development of Software: Guidelines and Recommendations from the ARPA Domain-Specic Software Architecture (DSSA) Program*, Teknowledge Federal Systems, 1994

[3] IEEE Std 1471-2000, *Recommended Practice for Architecture Description of Software-Intensive Systems*, United States, 2000

[4] Lane, Thomas, *Studying Software Architecture Through Design Spaces and Rules*. Software Engineering Institute, Carnegie Mellon University, 1990.