1 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.

1.1 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 behaviors 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]

The definitions show a great deal of similarities. In all software architecture is defined as a set of structures/components/modules, 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.
- The definition of [3] is more focused around 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]) focuses more on comparing architectures and how the systems themselves perform. This also becomes clear from the title of the article *Studying Software Architecture Through Design Spaces and Rules*.

1.2 Our view on architecture

We agree on the point that a software architecture is defined as a set of structures, components and modules, their properties and the relations between them. Not only we think it is important to let the architecture show the functional components, it might also be necessary to show some of the quality attributes that the stakeholders find important.

Our definition of software architecture will be: Software architecture is the structure of a software system defining how it is build up, which components has and their relations to each other. A software architecture shows the functional and the quality requirements in regard to the system.

2 Viewpoints

2.1 Privacy viewpoint

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

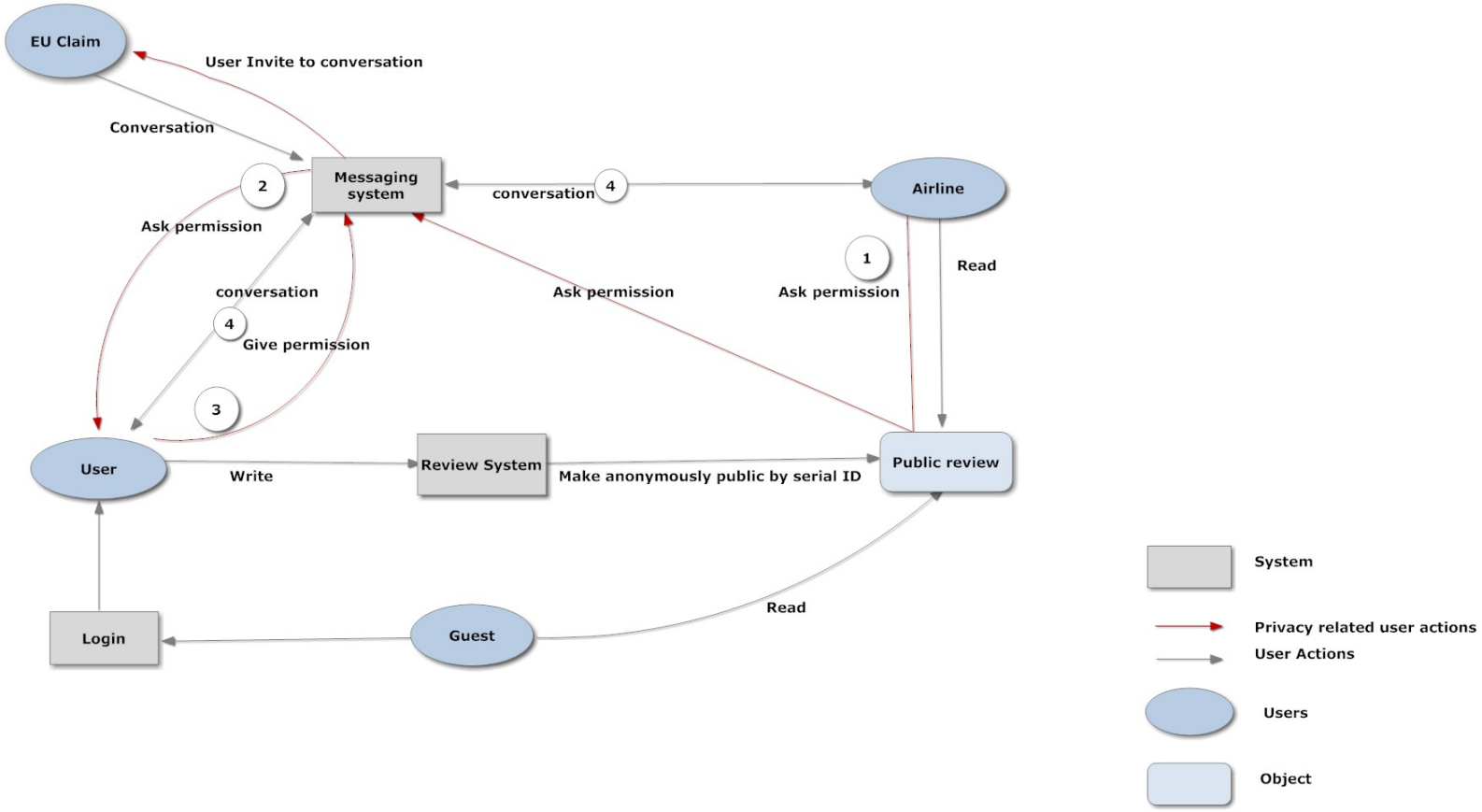

Figure 1: Privacy viewpoint

The privacy viewpoint sown in figure 1 shows how the privacy of the user is garanteed 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.

2.2 Data Flow Viewpoint

- Related stakeholders: KLM, Initiator
- Related Concerns: Data Integrity, Performance, Scalability

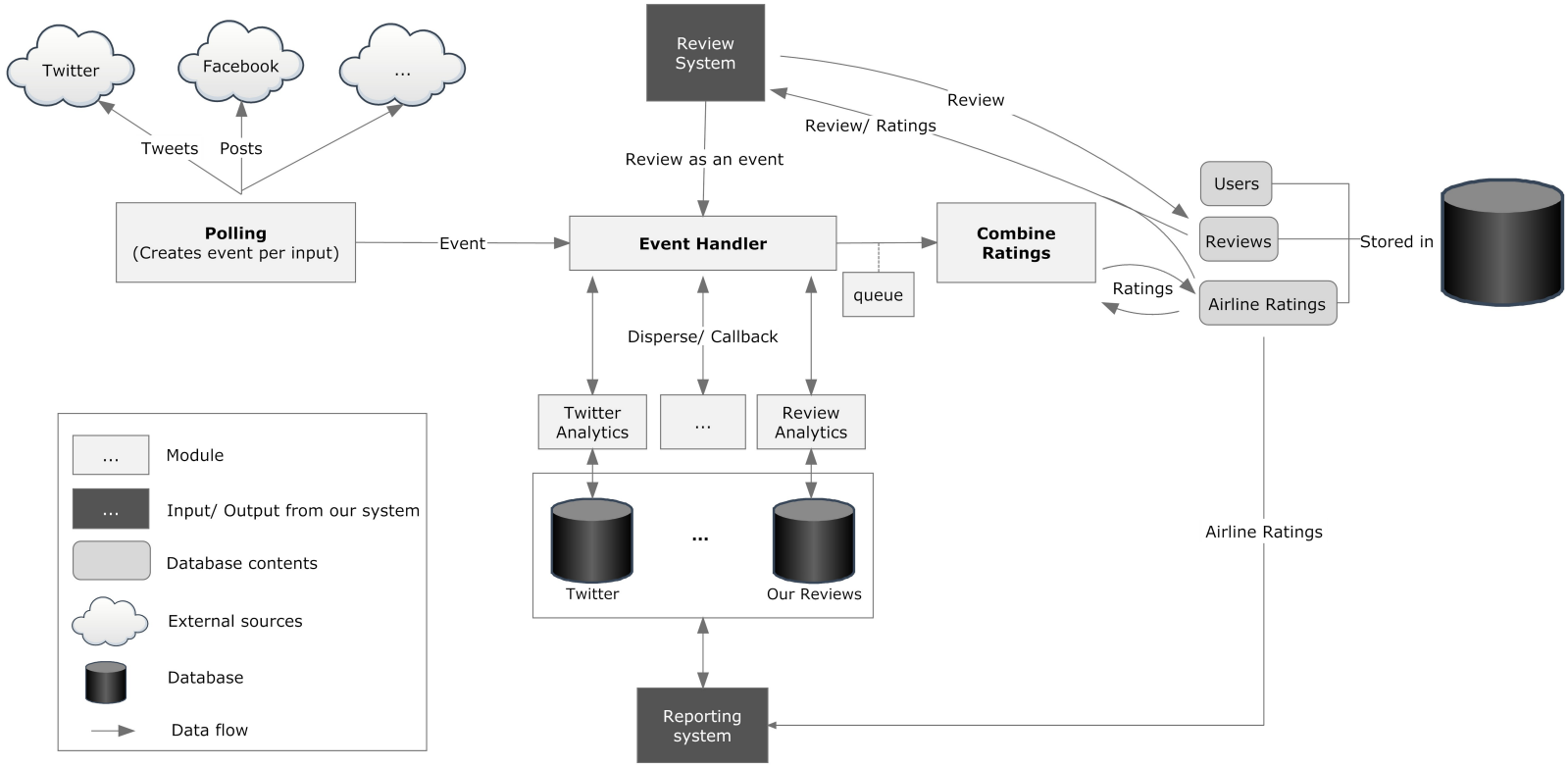

Figure 2: Data Flow Viewpoint

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 what is going on in there is of no importance in this picture. 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.

3 Description of our Solution

A Domain Knowledge - Design Decision

In this section we will present the domain specific problems related to airline reputation management system. 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 we searched through websites relevant to airline companies. We tried to understand what is important to 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 us 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:

- Analytics concerning its progress through graphs and diagrams.
- Latest reviews or bad reviews from our site.
- Crossreference 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 analyzed first in order to be usefull. There are multiple options available when analyzing 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 usefull, 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

A.4.1 Import questions

This section lays out the domain related questions with data storage. 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 we define 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 analyzed 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 we take one specific company (KLM) and multiply the values found for this company with 600:

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 scalablity.

How many databases? If we use more than one database we need to decide how much we can split up the data while still keeping it 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 seperate 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 analyzed 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 we can balance the load and only write away data after it has been analyzed, 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 persistet 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 separatly.

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 continously 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 therefor 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 powerfull 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 durin 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 canceled the service from the airline company to accomodate 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.

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.