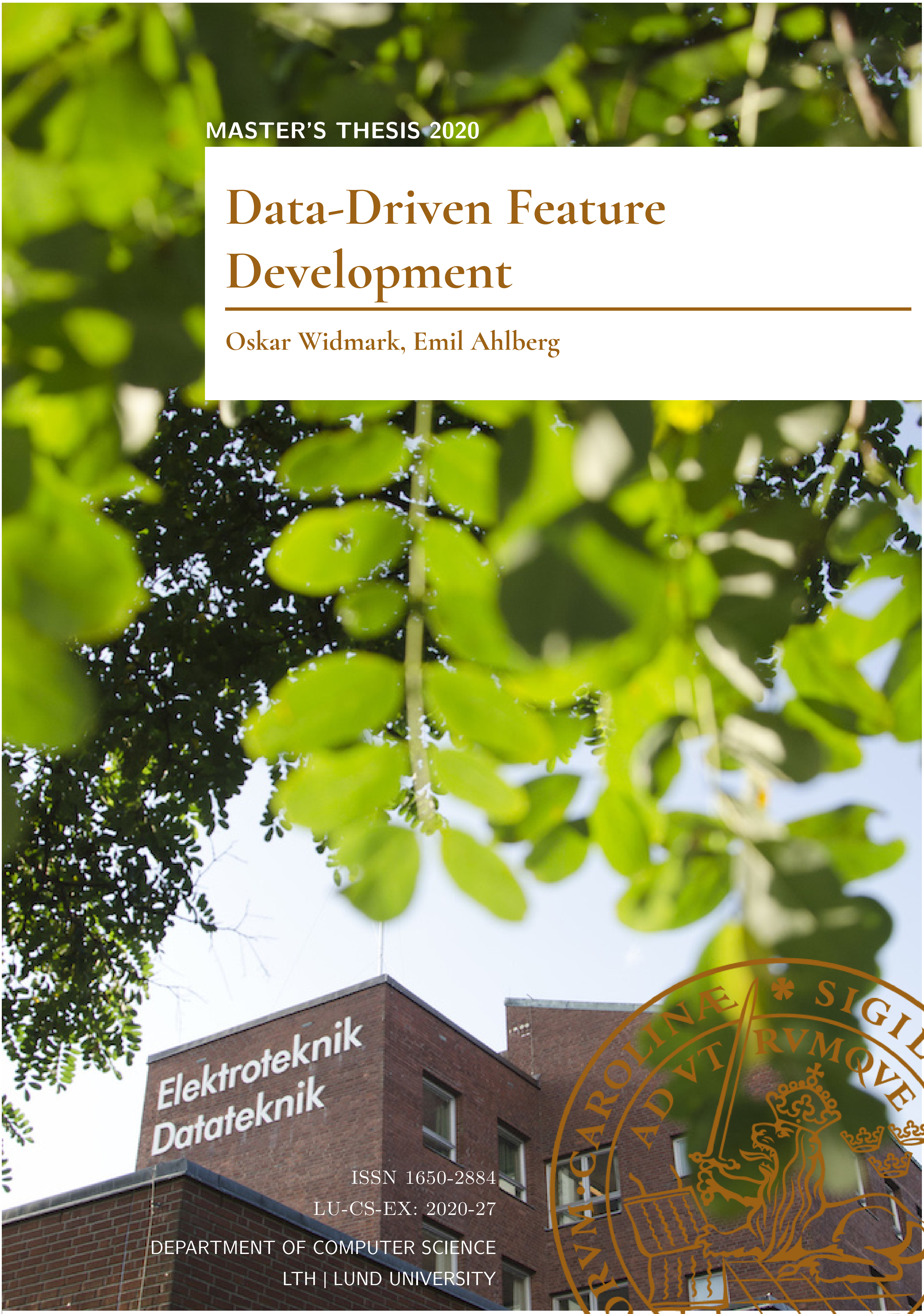


MASTER'S THESIS 2020

Data-Driven Feature Development

Oskar Widmark, Emil Ahlberg

ISSN 1650-2884
LU-CS-EX: 2020-27

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-27

Data-Driven Feature Development

Oskar Widmark, Emil Ahlberg

Data-Driven Feature Development

Oskar Widmark
tpi14owi@student.lu.se

Emil Ahlberg
har12eah@student.lu.se

June 22, 2020

Master's thesis work carried out at Axis Communications AB.

Supervisors: Ulf Asklund, ulf.asklund@cs.lth.se
Markus Andersson, markus.andersson@axis.com

Examiner: Martin Höst, martin.host@cs.lth.se

Abstract

Data-driven decision making in feature development is the continuous utilization of data in the development process, intended to focus development resources efficiently. This thesis aims to do two things: to find evaluation criteria that can be used to evaluate features of a mobile application, and to explore the next step in a data-driven development strategy for an organization by conducting an A/B test as a proof-of-concept implementation.

A model intended to capture user value and satisfaction solely by processing collected usage data is constructed using evaluation criteria which show different patterns in user behavior. To validate this model, user opinions are collected through surveys. A relative ranking of application features is obtained from the model, which corresponds to the ranking obtained from the survey data. Using a larger evaluation data set is suggested to further validate the model.

The proof-of-concept A/B-test is implemented and deployed to live users. Two implementations of a feature are compared to a control group using the aforementioned evaluation criteria. When the results are interpreted, the organization procures additional knowledge of the application and its user base, as the two implementations perform better than the control group. We conclude that the A/B-testing framework is a good way of experimenting with new ideas in a data-driven manner.

Keywords: data-driven decision-making, mobile software development, feature development

Acknowledgements

We would like to thank AXIS Communications for providing the thesis opportunity. During the thesis work the Mobile Applications department provided a great work environment and was very accommodating towards us. The teams at Mobile Applications have been very helpful and have provided support whenever needed.

Special thanks are extended to:

- Markus Andersson, our supervisor at Axis, who has provided great advice, much-needed help, and organizational knowledge throughout the thesis work.

- Ulf Asklund, our supervisor at LTH, who through our meetings on a weekly basis has been invaluable for maintaining a structure of the thesis as a whole and has always provided helpful counsel when needed.

Contents

Chapter 1
Introduction

1.1 Purpose and goal

Software development is becoming more and more customer-oriented, where instead of planning giant projects upfront, companies want to decrease their time to market and ideally deliver timely improvements to their product end-users. In competitive market environments, resources must be spent on what matters to the customer. One immediate consequence this has to an organization is that new knowledge needs must be met; how does the end-user think of, use, and perceive the product? The software industry is shifting towards shorter development cycles, where agile strategies are employed in order to deliver value to the end-user in an often continuous deployment manner. This provides an even greater incentive for companies to collect and interpret customer feedback as quickly and accurately as possible. Direct communication with the end-user is an obvious way of doing this and can be done through either surveys or talking to lead customers, which should represent the user base as a whole. This process can however be slow and/or inaccurate. Fortunately, the advent of usage and performance data collection has led to a quantitative way of evaluating user experience. The incorporation of usage data into the decision-making process is called **Data-Driven Decision-Making** and will be the focus of this thesis.

1.2 Research questions

The research questions of this master thesis were formulated during the first couple weeks of the thesis work. Since the main interest from AXIS Mobile Applications' perspective was to investigate how they could evaluate their features post-deployment data, the scope was set. They were then written as one general question about how post-deployment data can be used in feature evaluations, one specific question about feature evaluation criteria, and one development process-oriented question, since AXIS would want to use potential findings in

practice. The research questions can be viewed below.
(RQ1): How can post-deployment data be used to evaluate the level of success of feature implementations?
(RQ2): What types of evaluation criteria could be used to evaluate features released to the end-users?
(RQ3): How can the evaluation of implemented features be used in current software development processes?

1.3 AXIS Communications

The Swedish company AXIS Communications is a manufacturer of network-oriented security solutions, such as network cameras for surveillance, with various software solutions targeting several different types of users. The different mobile application solutions of the company enable the end-user to use features such as remote viewing surveillance cameras, receive different notifications, and remotely manage their physical access control by means of door station calls routed to mobile clients. The Mobile Applications department maintains and develops the AXIS mobile applications, which is the department where the thesis work will take place. This department develops and maintains several mobile applications for different systems, for both the iOS and Android platforms.

1.4 Contributions

Most practical work has been done in tandem, making use of pair programming and code reviewing, and no parts of this can be fully attributed to one of us. In some phases, we focused a bit more on different parts. During the iOS development, E. Ahlberg acted as the main programmer. The reverse can be said about the SQL coding parts, where O. Widmark played a key role.

Report-wise, the theory and mapping phase was reworked by both of us multiple times. In the methodological sections, as well as in the result and discussion, work was split into Widmark focusing more on the evaluation criteria, while Ahlberg focused on the A/B-test.

1.5 Ethical considerations

This thesis handles data that originates from users of the applications we are investigating, which can raise concerns about privacy. The usage data collected at AXIS Mobile Applications contain no personal identifiers and are in accordance with GDPR. Therefore, we conclude that our usage of this data will not break privacy since no additional personal data will be collected. This also applies to the in-app surveys, whose data is collected in the same way as all other usage data. The same can be said about the results from the online surveys. The responses are anonymous and cannot be traced back to the users who took them.

Chapter 2
Theory

2.1 Software development processes

2.1.1 Agile development

Agile development is a software development approach aiming to tackle the problems faced in traditional software development. Traditionally, the software is developed in sequential phases, where typically a phase must be finished before moving on to the next step. For example, the requirements must be set in stone before beginning to design the product. This model is known as the waterfall model and refers to how software was developed before agile methodologies were a thing [28]. This approach leads to a rigid development process, where real user feedback is delayed, possibly until the project is completed. While the product has gone through rigorous planning and has well-defined requirements, uncertainty, changing requirements or different interpretations can be detrimental to effective software development [28]. These problems are handled well by the Agile development model. The values defined in the Agile Software Development Manifesto [5], which are in contrast to the structure of the waterfall model, are the following:

> *Individuals and interactions over processes and tools*
> *Working software over comprehensive documentation*
> *Customer collaboration over contract negotiation*
> *Responding to change over following a plan*

2.1.2 Stairway to Heaven

Holmström Olsson et al. [24] present an evolution path in software development strategies which is followed in most companies, aptly named **Stairway to Heaven**, where "heaven" is the best implementation of Agile development. This is by no means a linear path; some parts of the companies might be further ahead than others. The evolution path is defined by five steps as seen in Figure 2.1, where the aforementioned traditional development is step A.

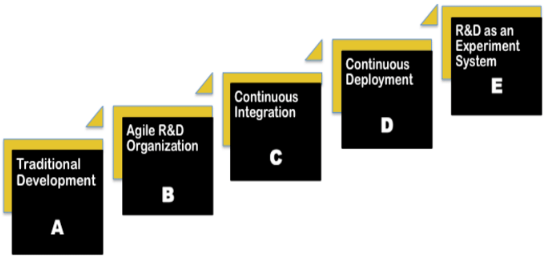


Figure 2.1: The "Stairway to Heaven" [24].

Step B concerns the first implementation of Agile development, where a company adopts agile practices in the development process. While improvements can be seen immediately, such as more flexibility concerning requirements and development structure, communication with the customer takes a long time, i.e. the feedback loop is long since meetings with customers need to be set up and software updates are sparse. Furthermore, product management and system verification are not yet agile. Planned releases that violate agile principles are still practiced.

Step C is the next step in the evolution: Continuous integration. Here, automated building and testing of continuous additions of the developers leads to a flexible development process, where ideas can easily be implemented and evaluated. If continuous integration is not applied, the work of different developers might diverge, which will lead to great difficulties trying to merge all the changes [11], draining time, and resources.

Step D is called "Continuous deployment", where the increased frequency in deployment leads to more frequent customer feedback on the current implementation. Being able to deploy updates at any time brings timely value to the end-users. This will result in more data, facilitating informed decision making about the product. Incremental changes are easier to debug, and quick fixes will lead to minimal downtime for users. However, this requires the end product to be able to receive updates quickly and install them without impacting critical performance.

In the final step E, deployment becomes more about experimenting with ideas and less about delivering a final product, using R&D as an experiment system. Here, usage data is collected in great amounts of not just what the customer thinks, but how the customer interacts with the product.

Implementing these steps works towards solving the difficulties connected to the **open-loop problem** [25]. The open-loop problem refers to the problem of slow and inefficient use of usage data in the development process. Ideally, a software organization wants quick and accurate feedback as input in the decision-making process, thereby closing the loop. Otherwise, product management that does not know what the customer values will conduct feature development driven by informed opinions (at best) from product management or developers, which could be biased. This can lead to features being developed that are not aligned with the customers' needs, requiring redeveloping and frustration in the development teams, which in turn wastes resources and thereby impeding progress.

2.1.3 Data-Driven Decision-Making

The way to decide which development activities to prioritize, using usage data as an input, is called **Data-Driven Decision-Making** (DDDM). According to Holmström Olsson and Bosch in their paper ”From Opinions to Data-Driven Software R&D” [25], the focus of an experiment system, the final step in the ”Stairway to Heaven”, should be to collect data from customer usage and feedback, and use it to further improve the product. However, the experience of developers and managers is not to be cast aside entirely, and decisions should be ”*informed* rather than *driven*” [22]. The usage of DDDM has been shown to increase productivity and profits for companies by up to 6% [8]. However, a bad implementation of DDDM could even lead to worse decisions than not using it. This could be due to ”bad data”, which is either poorly processed or unimportant, bad visualizations, or lack of understanding of either the data or the relationships contained in the data [18]. Therefore, focusing on collecting the right data and making a correct analysis is the central part of making DDDM work.

2.2 Customer orientation and feedback

Different information channels and mechanisms of acquiring customer feedback data for software companies have previously been summarized by Sauvola et al. [29]. A quite prevalent feedback mechanism that was identified was that of different manual channels, such as customer interviews, meetings, and lead customer testing. Meetings with and between internal stakeholders, sales personnel, product owners, and product management teams were another identified feedback method commonly utilized. Prioritizing and resourcing with regards to feature development was found to be challenging and very reliant on product manager assumptions and opinions, which were not validated until after the product was released. Also, none of the companies that utilized data collection in the post-deployment phase had a systematic way to leverage post-deployment data in their respective development process.

Suonsyrjä et al. have attempted to categorize the knowledge sources that organizations are using to understand their users into three main categories [32]. The *customer contact* knowledge source was defined as the information gained from face-to-face contact and interaction with customers, *user-centered information* constituted the second source comprised of feedback such as reviews about the product and support requests, while the third category consisted of the organizations *own information gathering*, such as post-deployment-data collection [32].

2.2.1 Different types of overall feedback data

Previous work has also tried to categorize distinct types of customer feedback data, coming from all knowledge sources, used in the industry [32]. The identified categories ranged from time themed data (time of day usage, on what parts in the product are the time spent), performance-related data (load, stress, and response time measures), amount-of-use metrics (of features and functionalities) and error-related events (exceptions and error messages). Different kinds of analyses performed using the listed feedback data types were then distinguished by the authors. Value analysis constituted one such category, entailing analysis of features and functionalities (in terms of most and least used) together with an attempted weighting of the importance of different use cases and functionalities. Another analysis category was user problem or pain point focused; this type of analysis aims to locate and understand weak or difficult areas of the software. Use paths were the third identified category, where important workflows and crucial paths can be analyzed. The two last categories were identified as the 'actual use' and 'user profiling' categories, where the former analysis type specifically considers feedback from the end-user (how the product actually is performing, and how it is used), while the latter refers to the analysis of different groups of users (on either personal attributes such as age or behavior, or technical attributes such as type of platform or specific product used). Finally, the authors identified use objectives for using the different categories of analysis on the distinct feedback metric types, as UX-, actual needs-, resourcing, and development options-related [32].

2.2.2 Post-deployment/user-interaction data

The collection of knowledge from customer contact and user-centered information is often slow, as presented in Section 2.1.1, leading to the open-loop problem. If **post-deployment data** can be used in place of these sources, the development process can be sped up and the loop can be closed.

Suonsyrjä et al. have in their article ”Post-Deployment Data: A Recipe for Satisfying Knowledge Needs in Software Development?” [32] previously studied possible ways that post-deployment data can be used by software development teams to better understand their end-users, continuing the work described in 2.2.1. The term post-deployment data refers to the data gathered after the commercial deployment of a product [32]. Post-deployment data can be categorized as, for example, performance data, quality data, usage data, or end-user feedback data. Data collected by tracking the users' interactions with an application at the user-interface level is called user-interaction data. Attempts to provide understanding about what objectives and utilization targets an organization might have and challenges that an organization might face when utilizing user-interaction data have been conducted [31]. The resulting analysis and use objectives have been summarized in Table 2.1.

Analysis Objectives	**Description**
Value Analysis	Find out how valuable a feature is. In addition to total usage, can also be based on who, from where and at what time the feature is used.
Pain Point Analysis	Identify possible problem areas for the users. Problems may relate to system performance or low usability. Use of timestamps may often be of use.
Use Paths	Find out the order of the use of features.
User Profiling	Ways of segmenting the total user population. Segmentation may occur on technical properties (platform used), or properties related to the user (novice or expert).
Use Objectives	**Description**
Informed Feature Development	Use insight from user-interaction data to support decision making. Identify features that may need improvements and guide new designs.
UX	Similar to *Informed Feature Development*, but specifically targeted at the UX.
Resourcing and Prioritizing	Having estimated the value of a specific feature, the priority of the backlog could be affected.
Requirements Validation	For example, by looking at the technical prerequisites of the user base, a minimal viable product or feature can be determined.

Table 2.1: Summary of analysis objectives and utilization targets identified for organizations collecting user-interaction data [31].

2.3 Creating evaluation criteria

The management of most organizations today would most likely say that they want to be, or already are, data-driven in their decision making [33]. For any type of organization, there are also most likely several measurements that would be beneficial to track, for example revenue, costs, and customer satisfaction [21]. Defining an overall evaluation criterion (OEC), to incorporate the weighted average of the different important business sub-metrics could thus be very valuable when it comes to making decisions on both the managerial and developer level. Such an OEC should be measurable in the short term (i.e. 2 weeks) but needs to capture long term value drivers for the organization [21]. Consequently, poorly constructed evaluation criteria can in some cases seem beneficial in the short term, while they perform terribly in the long term [21]. For example, hiking prices might drive short term revenue increases, while the long term outlook of the business actually may worsen. For a search engine, repeat visits (sessions per user) might thus be a much better measurement than the time a user spends with the service, which may increase due to inefficient querying. For example, an OEC for the Bing search engine is the sessions/user metric [20].

To be able to use usage data in a productive manner, it must be processed in ways that generate meaningful information for the developers and decision-makers. Generating adequate metrics from observed data constitutes a basis of analysis from which conclusions can be drawn. Rodden et al. identify a problem in that behavioral usage data is not used in a large scale way [27]. They present a framework used by Google called ”HEART” which provides a metric framework for web applications that breaks down the data into five categories: Happiness, Engagement, Adoption, Retention, and Task success. Three of these, engagement, adoption, and retention, are presented as new ways of evaluating user behavior from usage data. Engagement concerns the intensity and frequency users interact with an application. Adoption measures when new users start using the application. Finally, retention keeps track of users that keep using the application over time. Through having a great amount of post-deployment data, these metrics can be applied to produce meaningful results to be used in the decision-making process.

2.4 Testing hypotheses

In order to make data-driven decisions, an idea of what constitutes a successful implementation should be considered. Rodden et al. [27] suggest that first setting goals, then identifying what signals can indicate that a goal is fulfilled, and finally predicting how these signals can be seen in the measured metrics, in this case, the HEART framework metrics. This process can be seen as a more informal analogy to the statistical hypothesis testing, which is *the* standard framework of validating results in most sciences. An example of a model for using a similar method for testing hypotheses in data-driven decision making is the Build-Measure-Learn-loop, which is the core idea in the Lean startup model, an implementation of Agile development [26]. Here, the process can be broken down into the following four steps:

- **Plan**: Define a hypothesis of what to expect from the product, and make a plan of what to develop.

- **Build**: Create a minimal viable product (MVP), which is the least complex working version of the product, or develop an already existing product.
- **Measure**: Analyze how the product is holding up to the hypothesis.
- **Learn**: Depending on the results, there are two ways forward:
  - **Persevere**: If the hypothesis is correct, continue building the product the same way as before, i.e. jump to the **Build** step.
  - **Pivot**: If the hypothesis is refuted, change the course of the development by changing the hypothesis or the way to develop the product, i.e. jump to the **Plan** step.

Iterating this process until a satisfactory result is attained ensures that a minimal amount of development time is wasted.

2.5 Online controlled experiments

Internet-connected software running on clients (websites, mobile applications, and PCs) enables conducting online controlled experiments, or more colloquially called A/B-tests. The key motivation for running these experiments is the causal relationship that can be derived from a change in implementation [20]. Running an experiment in a control/treatment manner, potential improvements to the evaluation criteria can oftentimes with a high probability be associated with the treatment. This methodology has been adopted and embraced by several big tech companies, like Amazon, Facebook, Microsoft, Google, and LinkedIn [21].

Utilizing A/B tests allows for experimenting with ideas, which is the core concept of step E in Figure 2.1, discussed in Section 2.1.2. Normally, the test consists of two variants: Variant A, which is the default version of the tested feature, and variant B, which is the alternate version of the feature [21]. In terms of which users to include in the test, different sampling approaches can be used. The sample can consist of specifically targeted users, be completely randomized, or simply include the whole user base. The sampled users are evenly distributed between the intended test groups. The selection could be further augmented by ”triggering”, meaning only including users that reach a certain state, for example only including users that perform a specific action. This narrows down the sample to only include users that, for example, are concerned with the new feature. Generating data from observations of actual use (by logging events) is then critical to generate the metrics needed to evaluate the experiment, and should be added whenever it is possible and economically justifiable [21]. When the results of an experiment are to be interpreted, relevant metrics should be used. These metrics should preferably not only work in the short term, but long term business value should be captured by the criteria. Several pitfalls have previously been described when it comes to interpreting and evaluating the outcome of experiments [20, 9]. When introducing new features, primacy and novelty effects may be observed. Primacy effect refers to the short term decrease in efficiency that users may experience when the UX changes, while the novelty effect describes the initial increased usage of a new feature when it is investigated. Running an experiment for more than a week is one way to attempt to mitigate a possible initial effect bias.

2.6 The ”quality in use” concept

The International Organization for Standardization (ISO) consists of international expert groups, which produce reports on the current state-of-of-the art consensus on various topics [6]. The ISO/IEC 25022: Measurement of quality in use [1] includes measures to gauge effectiveness, efficiency, satisfaction, which constitutes parts of the *quality in use* ISO definition. These measures can be observed in Table 2.2. The core concepts included in the quality in use definition, for example satisfaction, might be a bit abstract. The measures of satisfaction are on the other hand more concrete and thus easier to try to evaluate.

Effectiveness	**Efficiency**	**Satisfaction**
Tasks completed	Task time	Overall satisfaction
Objectives achieved	Time efficiency	Satisfaction with features
Errors in a task	Cost-effectiveness	Discretionary usage
Tasks with errors	Productive time ratio	Feature utilisation
Task error intensity	Unnecessary actions	Proportion of users complaining
	Fatigue	User trust
		Proportion of user complaints about a particular feature
		User pleasure
		Physical comfort

Table 2.2: Summary of ISO/IEC 25022 Measurements of quality in use [1].

2.7 Online and in-product surveys

One way to measure the user experience of an application is to conduct a survey. Asking the survey respondent, for example the user of a mobile application, about the perceived satisfaction related to engaging with the app, or a feature of the app, generates user feedback data, which complements the post-deployment data collected automatically [27]. Online surveys constitute a cost-effective way to conduct surveys, when compared to traditional alternatives, such as phone surveys or paper surveys [37]. A disadvantage of the online survey method is the lower response rates when compared to other survey methods [30]. The online survey can be conducted using different survey mediums, either by sending users an email with the attached survey, or questions that can be asked in an in-product fashion. In-product surveys can be presented to the user when the user engages with the product or a feature of the product. For instance, LinkedIn utilizes both types of surveys in a complementary way, where email surveys are considered to be better suited for longer surveys [10]. The quality of a conducted survey can be affected by a number of factors, related to the design of the survey, discrepancies in the target group and the actual population, and errors due to non-responses [35]. Poorly formulated questions and confusing instructions may cause the respondent to misinterpret questions and thus unintentionally answer incorrectly, which suggests that designing the survey should be a conscious effort.

When rating quality, the Likert scale is a common way to gauge a respondent's opinion [4]. The Likert scale is usually a five-point bipolar response where ”1” is the least-valued and ”5” is the most-valued response (see Figure 2.2). The results from a Likert scale question are ordinal data, which can be ranked, but not be measured in a relative way. For example, the distance between 1 and 2 is not necessarily the same as the distance between 4 and 5. Therefore, descriptive statistics such as means and standard deviations are generally not valid, and usage of medians and ranges are preferred.

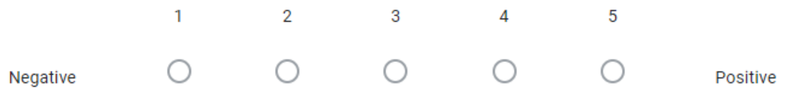


Figure 2.2: An example of a Likert scale.

2.8 Statistical modeling and tests

In order to empirically analyze and draw conclusions from generated post-deployment data, related to an A/B-test, valid statistical reasoning and modelling are needed.

The Poisson distribution plays a vital role in analyzing discrete events occurring over time, and it is used to empirically treat count data [19]. for example by using A/B-tests. Assuming that the count (i.e. the number of actions) generated by any user of either of the test groups is interchangeable with assuming that the count data of all users arrive with equal probability at any given point in time, i.e. the count data is generated by a Poisson process. This occurs in a memory-less fashion, meaning that past occurrences do not influence future ones [19]. This is a reasonable assumption as long as the users do not differ widely in terms of usage patterns (for example, the data set does not include bots).

Sampling count data over time can be used to estimate the intensity λ with which events occur in time, denoted $\hat{\lambda}$. A property of the Poisson distribution is that its variance is also the mean number of events per time unit [19]. When the observed intensity is not too small ($\lambda > 10$), the Poisson distribution approaches normality [7].

The Welch's t-test is a test for checking if two samples have the same mean, where the underlying assumption is that both samples are normally distributed. This test is an adaptation of the Student's t-test, with the modification that the variance of the different samples is allowed to vary [36]. This test can thus be utilized to test the null hypothesis in terms of sample averages, using two normally distributed samples. These samples could theoretically be acquired from observing count data generated by a Poisson process, which under mentioned circumstances approaches normality.

The χ^2 test constitutes one way of analyzing 2x2 contingency tables with a large number of samples, in regards to the confidence that the null hypothesis can be rejected with when looking at two different samples [2]. This test can thus, for example, be used to measure the statistical significance of results with binary outcomes. One example of this could be a frequency segmentation of the test groups, into categories that either displayed behavior X or behavior Y. This type of data can easily be used to construct a contingency table, upon which the test is applicable.

Chapter 3
Approach

Our thesis will focus on creating evaluation criteria measuring feature success. We will also propose a new way of conducting feature development called A/B testing.

The thesis work as well as the thesis itself was divided into three phases:

- **Investigation Phase**
  - Identify the client software, from which the usage data is collected
  - Get acquainted with the development process within AXIS Mobile Applications
  - Investigate how usage data is collected and used
  - Understand the business case for Mobile Applications
  - Compare the development process to theory and find out if something is missing
  - Map out which tools are utilized today in the development process
- **Development and Analysis Phase**
  - Evaluation criteria
  - A/B-test
- **Evaluation Phase**
  - Define what will be the foundation for evaluation of our results
  - Present our two methods of collecting evaluation data:
    - In-app survey
    - Online survey
  - Present how the evaluation will be conducted

Chapter 4

Investigation Phase

4.1 AXIS Mobile Applications

4.1.1 The client software

The applications (with related post-deployment data) that this thesis concerns are the mobile applications named AXIS Companion and AXIS Camera Station (ACS). The core use case for both of these applications is to enable secure remote video surveillance, streaming video data from recording devices. The applications are differentiated in terms of the intended end-user; Companion is regarded as a solution aimed at customers with small to medium-sized systems, while the ACS application is tailored towards customers in need of medium-sized to large system solutions.

Figure 4.1: The AXIS Companion and the AXIS Camera Station applications respectively.

4.1.2 Development processes

The Mobile Applications department at AXIS develops software utilizing agile strategies, inspired by the process frameworks Scrum and Kanban. Additionally, routines and tools are in place allowing automated testing, continuous integration, and continuous deployment. Releases are made on at least a weekly basis, and each developer can deploy code to production after it has gone through live testing. The development teams are developing features using the BML-loop. Features are implemented first as a Minimal Viable Feature (MVF), and then improved upon using both specifications and customer feedback. There exists a perceived weakness in the Learn part of the BML loop, as there is no formal follow-up for when feedback is collected for a specific feature.

Currently, the decision making concerning feature development is done in the "product manager sync" activity, which is a scheduled activity occurring once every other week. For each application, the meeting held includes the product owner, the engineering manager, product specialists, the agile team lead, and technical specialists based on the particular basis of needs. Here, the product owner communicates the interests of customers, while the agile team lead communicates information concerning the current development.

4.1.3 Current information channels

The AXIS mobile applications department receives user and customer feedback through a variety of information channels. The product owner aggregates information from discussions with various stakeholders, sales staff, technical specialists, and customer support, in order to establish the best possible understanding of what should be prioritized going forward. Sales staff have direct contact with customers, technical specialists interpret and convey the behavior of the deployed product looking at post-deployment data, and customer support communicates problems they have gotten reported. In addition to this, app reviews for both the iOS and Android platforms are collected. For trying new products and features, the use of lead customers, known as *pilot sites* or *pilot users*, are utilized. In-person meetings between AXIS R&D teams and the pilot users are thus another highly regarded information source for the company. Furthermore, collected post-deployment data is also used as an information channel, which is described further in Section 4.1.4.

4.1.4 Current status of data utilization

Since the introduction of post-deployment data collection, information extraction has been conducted at the developer level, where a dashboard of different metrics has been created using the Data Studio visualization tool (see Section 4.2.3). In most cases, the insight gained from data analysis is relayed bottom-up, meaning that developers and/or the agile team lead inform different stakeholders such as the product owner or other divisions of the organization about what has been observed.

The vast majority of analysis in place is related to the performance and stability category. Crashes, or app stability, are monitored using a web application. Error rates on certain tasks and requests made by the application are also tracked, and performance times such as load times for different core features are well known and monitored by each team within the department. For example, the Data Studio visualizations can highlight features which crash often or slow down the application, which need to be urgently taken care of. Additionally, there exists a visualization tool showing use paths, chronological sequences of events that users log when using the application.

General feature usage is also displayed, although in a simple way. Currently, it is visualized in two ways: through treemaps showing usage percentage per event, and bar graphs showing the most and least used features, in an attempt to distinguish between high and low-value features. The treemaps are used sparingly by one team, which uses them to decide which features need to be improved based on comparing actual usage to expected usage.

4.1.5 The mobile application business case

Establishing the business case for the company's mobile applications is very important for defining what usage patterns are preferable and what the overall goals should be. Both applications, Companion and ACS, are free to download and to use for both the Android and the iOS platforms. The prerequisite for utilizing the application is that the user owns at least one AXIS device, such as a camera, otherwise, the application offers no meaningful functionality.

Since the applications are free and without ads, the business goals for AXIS are different from most mobile applications. The applications are not directly monetized via the freemium model (free app, pay for premium functionality), so business cases such as ad revenue and different variants of sales conversions are not applicable to the department's particular circumstance. The mobile applications are instead offered as an additional way of interacting with the devices of the AXIS product portfolio, for example network cameras, and aim to increase the overall value of the entire product and solutions portfolio to the AXIS end-user. Trapping the user within the app or offering different incentives to increase total app engagement is not desirable according to AXIS, which on the other hand could be preferable in for example game applications monetized by ads. This puts the user value and satisfaction in focus. This, along with making sure that performance is good and error-rates are low, can be directly tied back to the "quality in use" concept described in Section 2.6, which AXIS uses as a basis for their goals.

4.1.6 Comparison to theory

In the **Stairway of Heaven**-model, Figure 2.1, the Mobile Applications department belongs fully to step D, **Continuous Deployment**, having a continuous deployment pipeline making deployment at-will possible. The next step **R&D as an Experiment System** on the evolution path, entails the introduction of experimentation with different ideas, in order to learn what the development should focus on. Reading the department's internal goal documents, it is apparent to us that the department is very cognizant of the need to shorten the feedback loop and implement this in their development process. Usage of performance and stability data to influence decision processes are already quite refined and indicate that AXIS values data in a way that can further push them towards using R&D as an Experimentation System. In Table 4.1, we explain how AXIS Mobile Applications fulfill the Analysis Objectives proposed by Suonsyrjä et al. (see Table 2.1).

Analysis Objectives	Summary
Value Analysis	Currently tracked as specific feature usage in relation to the total user base. The open loop problem is vocalized by management, but are they prioritizing the right things?
Pain Point Analysis	Error, performance, and stability data is in many cases automatically monitored and overall well-analyzed.
Use Paths	Tools to analyze this are well in place, but are not used much.
User Profiling	Segmentation on technical level (in terms of devices, mobile platform, etc) is performed. Users are not currently profiled based on usage behaviour.

Table 4.1: Our analysis of the fulfillment of Analysis Objectives by AXIS Mobile Applications.

4.1.7 What is missing?

The collection of data and tools for visualization are in place and is used for performance and error-related data. The successful implementation of a feature is of course partially defined by how well the application is running when using the feature. However, as mentioned in Section 4.1.5, user value and satisfaction is central in what constitutes feature success. Hence, measuring what users value and use may need further investigation and analysis of feature usage. The feature usage analysis currently only consists of usage percentages, we believe that there could be more ways to measure feature success.

Moreover, the perceived weakness in the Learn part of the BML loop with lacking hypothesis testing could be improved. This can be done by setting up hypotheses concerning feature usage, and by seeing how well these are fulfilled, deciding if persevering or pivoting is the best course of action. Since A/B-tests are not currently conducted, a proof-of-concept implementation and deployment of such a test would be interesting, in terms of what this framework could contribute to the department's learning. The test of a hypothesis concerning implementation variants of a feature and their subsequent performance based on user behavior would be interesting.

Use paths tools are in place ready for use but not fully utilized yet. Analyzing use paths could for example open opportunities for finding anomalies in how users navigate to certain features.

User profiling based on user behavior is also not done yet. The capability of differentiating between power users, the average user, and novel app users in terms of user behavior could potentially lead to new insights.

4.2 Tools

AXIS Mobile Applications use the following tools included in Google Analytics to collect and analyze event data from their applications.

4.2.1 Firebase

Firebase is a web-based development platform developed by Firebase Inc. acquired by Google 2014 [34]. It provides a number of cloud-based services for software development connected to authentication and the collection of application data. Integrating Firebase in an app enables automatic collection of user engagement and app usage data [17]. The data is captured in an event-wise manner; for example, when a user interacts with the application, the interaction is logged as an `user_engagement` event. By default, certain user properties will be added to the event, refer to Section 4.2.2 for a detailed overview of the data set. The event data is continuously uploaded to the Firebase cloud, which through the Firebase web application can be viewed and analyzed. Firebase also allows for integration with their cloud data warehouse BigQuery, which is utilized by the Mobile Applications department. The data collected by Firebase is thus also continuously funneled to BigQuery (more on BigQuery in Section 4.2.2).

4.2.2 BigQuery

Google BigQuery is a software-as-a-service that provides web access to Google Dremel, which is a distributed database querying system that can handle querying large datasets in seconds [23], and Google Storage, where the data is stored. Querying data is done using standard SQL.

Data set description

The data sets for the different apps at AXIS Mobile Applications all share certain characteristics. Automatic collection of events are identically across the applications, see Table 4.2.

Event Name	Trigger
app_update	When the app updates to a new version and launches again.
first_open	The first launch after an app installation/re-installation.
os_update	When the device OS system is updated.
session_start	After a period of inactivity, if user-app interaction exceeds a minimum time.
user_engagement	Triggers periodically, when the app is in the foreground.

Table 4.2: Sample of event types automatically collected by Firebase in Android and iOS apps, described by the complete automatically collected event list [12].

Events logged to Firebase are contextualized with several additional metric data fields, allowing for segmented analysis of the usage data. The data intended for analysis is the data exported to BigQuery. The Firebase-to-BigQuery export schema [13] of logged data describes the format and fields related to an analytic event. A sample of such information fields can be directly viewed in Table 4.3. Each event exported to BigQuery is further contextualized with parameters and user properties, where the details of a few interesting such fields are presented in Table 4.4.

Field Name	Description
app_info.version	The version name of the app.
device.mobile_model_name	The device model name.
device.operating_system	The operating system of the device.
platform	The platform of the app.
user_pseudo_id	A pseudonymous id for the user (app instance based).
geo.country	The country (IP-derived) from where the event was logged.
event_date	YYYYMMDD formatted date of when the event was triggered.
event_timestamp	The time of logging on the client, in UTC μs.
event_name	The name of the event.

Table 4.3: Sample of exported columns defined by the BigQuery Export schema [13], providing context to the reported usage of the applications.

Parameter Name	Description
firebase_screen_class	Denotes the screen class where the event occurred.
ga_session_id	A unique identifier identifying the session.
ga_session_number	Monotonically increasing ordinal number, specifying when the event occurred in relation to other events in the session.
User Property Name	
first_open_time	The time when the user first launched the app (in UTC).

Table 4.4: Sample of automatically logged event parameters and user properties tied to the analytic events exported to BigQuery [14].

The entries of the data set are logged on an event type format, meaning that data is logged when an event is triggered in the application. The event is contextualized with several different parameters, pertaining both to the context of the event and certain characteristics of the user. The events connected to the features of the application are named according to an internal name standard of the department. As an example, the video streaming event is named `ax_multiview_stream`, where `ax` marks the event as an AXIS custom event, `multiview` denotes an event group (namely events related to the multiview screen of the application), and `stream` refers to the actual action performed in the app. These types of events constitute the basis of the analysis since they hold the information about the users' engagement with the application features. The granularity of the data set also enables analysis to be conducted on various aggregation levels. For example, metrics can be derived on user-, session- and event levels.

4.2.3 Data Studio

Google Data Studio is included in Google Analytics as a tool to easily visualize data in many different ways with drag-and-drop functionality. It is specifically made with users with little experience in data analysis tools in mind. BigQuery is directly integrated into Data Studio, and queries can be specified and sent to receive data which fields can be plotted in various.

Data Studio is already an integral part of Mobile Applications, where it is used to visualize performance and errors. The data flow from the devices to Data Studio can be seen in Figure 4.2.

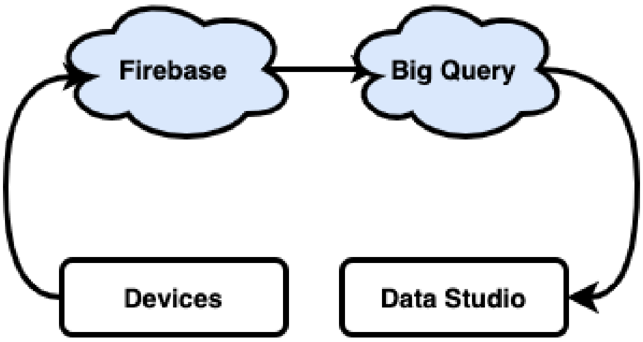


Figure 4.2: Data flow illustration of how the usage data is collected and subsequently visualized in the Data Studio tool.

4.3 Phase summary

Reaching the end of this phase, a couple of topics to pursue further have been identified. Investigating possible evaluation criteria used for feature assessments, and introducing the A/B-test methodology to the mobile applications department are from this point selected as the key tracks forward for the thesis work, with motivations provided below.

Since **A/B tests** aren't conducted, a proof-of-concept implementation, deployment, and subsequent hypothesis testing using collected data would be interesting to direct. A/B-testing is the main framework enabling the concept of experimenting with ideas, which according to discussed theory could provide the path forward in an attempt to close the open loop at the department.

Defining what constitutes favorable feature usage and constructing **evaluation criteria** on the feature level, attempting to measure how successful a feature is would be of interest to the department as well. To evaluate implementations of features on the mobile applications using post-deployment data, possible and desirable answers extracted from the usage data must first be considered and defined. What makes a feature successful? Furthermore, what questions would, when properly answered, provide actionable insights about a feature to the software development prioritization process? We pose the following three questions which we think that, when answered, could give valuable feedback and actionable insight:

- Do the users need the feature?
- Do the users find the feature?
- Do the users appreciate the feature, i.e. is it well implemented?

By utilizing available historic data for several different features of the applications, we will attempt to find usage patterns that will help answer the questions. For example, an unnecessary feature could potentially be characterized by users never using it, despite being aware of the feature's existence. This could indicate a low value of a feature, and focus could be put on either improving the feature, just leaving it as is, or removing it entirely. Another case could be users being unaware of the existence of a feature, and therefore never using it, which warrants other measures from the organization, such as making it more visible or informing about it. The recurring use of a feature could potentially indicate that the feature is both well implemented and needed, although the organization's domain knowledge might be needed from case to case to reach a conclusion.

Another identified topic, that from this point in the thesis is considered out-of-scope, is the segmentation of users on the behavioral level. To accomplish this, unsupervised machine learning could be tried out, producing user clusters on similar traits. The evaluation of the accurateness and potential use of this method would be a separate investigation, posing some difficulties in regards to how the assignment accurateness should be academically evaluated.

Finally, use paths tools are already in place, and we believe that our further efforts are best put into investigating A/B-tests and feature evaluation criteria, from this point forward.

Chapter 5

Development and Analysis Phase

After the investigation phase of the literature and AXIS mobile applications department, the next phase of the thesis work can be initiated. The main identified areas of potential improvement are researching and implementing evaluation criteria as a measurement of feature success as well as the conduction of online controlled experiments (A/B tests), using hypothesis testing. The work related to the two different tracks will be presented in two separate parts in this section.

5.1 Evaluation criteria

5.1.1 Introduction

The purpose of having evaluation criteria on the feature level for the mobile applications department, would as previously discussed be to try to gauge user value and satisfaction with a particular feature. This is the established premise, of what determines the overall success of a feature's implementation. Translating these goals to the metrics of the HEART framework, described in Section 2.3, the goal can be expressed as the maximization of H, Happiness. However, the way happiness data is collected in the HEART framework is by utilizing surveys, not by looking at raw usage data. Usage data is only used in the EAR metrics of the model, Engagement, Adoption, and Retention. These metrics can be calculated, using the already collected usage data at the department. An evaluation criterion for features should in this context then attempt to capture the happiness metric if possible, by only looking at data. Finding a way to accurately estimate happiness through usage data would constitute a way to solve the open-loop problem, since having such a metric would lessen the need for surveys, which are slow and not without overhead.

5.1.2 Defining the evaluation criteria

We propose using the same evaluation criteria for user behavior as defined in the HEART framework (see Section 2.3): Engagement, Adoption, and Retention. We will break down engagement in two parts: engagement and usage frequency. Retention is also captured in usage frequency.

We define **adoption** as a metric that captures how many users that actually use, or have adopted, a feature. It measures how many distinct users that have used the feature at least once for a given time period. For example, with a daily interval, the adoption each day is the number of distinct users that have used the feature for the first time during the time period, either for that day or cumulatively. Given a new feature, the adoption shows how successfully the feature is communicated to the user and answers the question: Do the users find the feature?

Adoption can also be measured during a later time period on a feature that has been released for some time. While still measuring real adoption somewhat, captures another aspect of feature success and answers the question: Do the users need the feature? The reasoning here is that users that have tried the feature before the given time period but had no need for it, will not use it again during the time period. This weakens the answer to the first question, but instead finds another way to measure feature success.

Engagement is a metric that measures in how many distinct sessions a feature has been used during a time period. For example, with a daily interval, the engagement each day is the number of distinct users that have used the feature that day. A high engagement should indicate that the feature is needed and possibly well-implemented.

Finally, we define **Usage frequency**, which groups distinct users by how many times they have used a feature in a distinct session over a certain time interval. Visualizing this results in a histogram where the x-axis represents the number of times a feature is used, and the y-axis represents the number of users. A high number of users using the feature many times indicates that the feature is needed and possibly well-implemented, resulting in a heavy-tailed distribution. We argue for that retention is caught by this metric, since a high repeated number of sessions where a feature is used by a user indicates that the user uses the feature over time, and is therefore retained.

5.1.3 Baselines

Some features are only relevant for a specific subset of users. For example, the slow-motion feature can only be reached from the playback state, which is only used when looking at recorded video clips. This is important to take into account since a feature located deep within the application could look unsuccessful when putting it in comparison to a shallow feature reached directly from the home screen. We solve this by looking at the state from which the feature can be reached. We call this concept baseline. With a baseline, we can get a percentage of users using a feature by dividing the number of users using the feature with the number of users that have been in the baseline state, thereby relativizing the data.

Furthermore, the concept is useful for features that have the same baseline in changing absolute numbers to percentages, creating a more familiar way to compare two metrics.

Moreover, a baseline can reduce the impact of external factors on the metrics. For example, a holiday could result in reduced usage, which would impact the absolute feature usage numbers negatively. However, since the absolute number of baseline usage also goes down, the negative impact can be nullified by using the baseline. Another example is the ever-increasing user base of an app. If the user base steadily increases, so will the usage of all features, and it will look like the feature is becoming more successful over time. With a baseline, the perceived positive effect is nullified, and the metrics can be viewed in a more objective way. An example of how baselines can eliminate noise can be seen in Figures 5.1 and 5.2.

We implement baselines in different ways for different metrics.

- **Adoption:** The metric is divided by the baseline adoption at the end of the time period, measuring daily adoption as a percentage of the final baseline adoption.

- **Engagement:** The metric is divided by the baseline engagement each day, measuring daily engagement as a percentage of the baseline engagement for that day.

- **Usage frequency:** This metrics is not impacted by baselines in the same way as the others since it is meant to show the distribution of the way the feature is used. Therefore, we divide the metric by the total number of distinct users using the feature during the time period, not impacting the distribution.

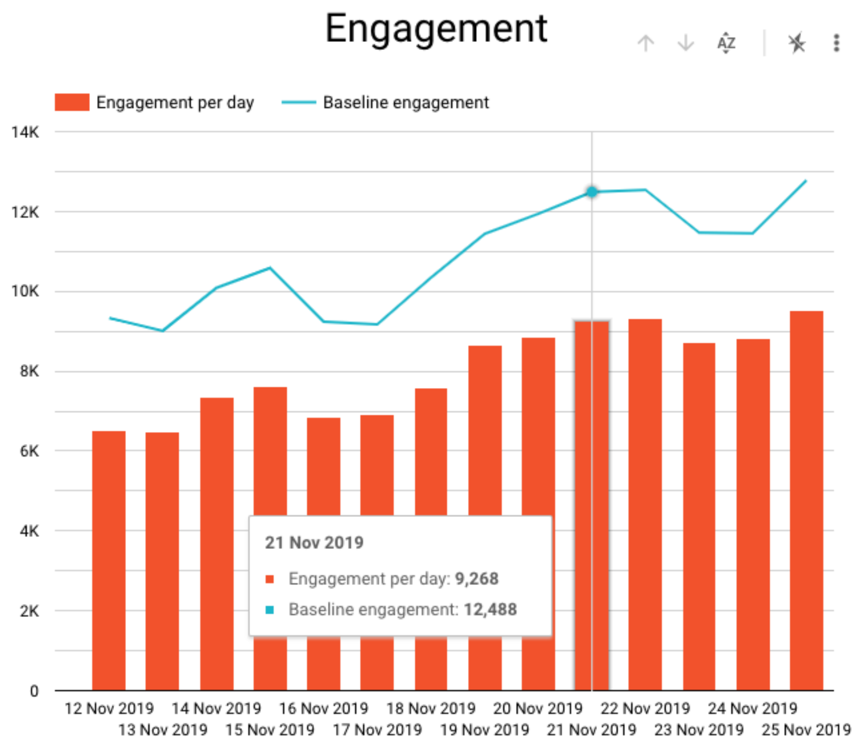


Figure 5.1: Absolute engagement (how many distinct sessions a feature has been used during a time period) for a feature X with the corresponding baseline engagement, showing how the noise for feature X can also be seen for its baseline.

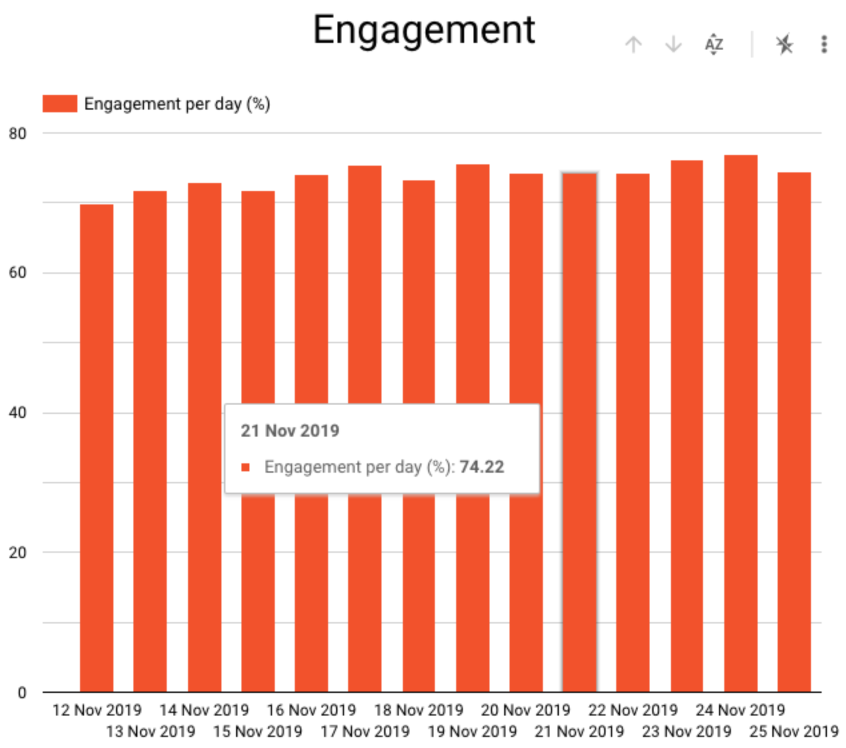


Figure 5.2: Relative engagement for feature X utilizing the baseline concept, showing how the noise in feature X is being cancelled out.

5.1.4 Platforms

The applications are developed separately for iOS and Android. Sometimes, features are not released simultaneously, which generates uneven comparisons for data across the platforms. This can manifest as sudden usage surges when a feature is released on the other platform. We introduce filtering based on either iOS or Android to be able to better sort out the results.

5.1.5 Practical Work

To present a visual representation of our evaluation criteria, we utilize the tools in Data Studio, using SQL queries to fetch the data from BigQuery. For a feature X, we first find out the name corresponding to the event logged when it is used. Using this as a parameter for the SQL query, we can fetch the relevant rows in the SQL tables. If a baseline is used, the rows containing the baseline event name will be fetched as well. Furthermore, we also input starting and end dates as parameters to get the time period we prefer, as well as the platform. We can then manipulate the data fetched to get the rows containing relevant data for each usage metric. Data Studio can then use the processed data to display it in the format we prefer.

We present the graphs as bar charts. For adoption and engagement, a breakdown with a daily granularity is shown over the given time period. Adoption is displayed both as the daily adoption as well as the cumulative adoption. Usage frequency is displayed in a bar chart as a histogram (histograms are not yet supported in Data Studio as far as we know). We are interested in capturing the overall usage pattern in terms of the number of sessions a user interacts with a feature over a time period. To this end we set a maximum bin number, putting power-users who use a feature in the same column (or bar). This limits the range of the X-axis, in order to prevent the graph from being unreadable. The numbers are presented in either absolute numbers, or in percentages if baselines are used. Leaning on the reasoning in Section 2.3, the evaluation criteria should be measurable while still capturing long term business value, hence 14-day time frames could be a reasonable time frame for the graphs.

5.1.6 Choosing the features

Using the metric visualizations available at this point, we looked at the overall patterns of several features. Both trying capturing user happiness with a feature using raw data, and then actually confirming this to be the case posed a real challenge. In an attempt to accomplish this, we chose four features from the application AXIS Companion to test our metric on. The reason for selecting this application was related to possible ways of evaluating our subsequent results in a later phase; lead customers, known as ”pilot sites”, were confirmed to be readily available, for us to ask questions to if needed.

The features chosen to analyze were the following, with accompanying motivation distilled from talks with developers, using their domain knowledge of the application:

- **Event list** - A new feature released a month before. For this feature, we were also provided an in-app survey (see Section 6.1.3).

- **Timeline** - A feature that is used a lot, and that the developers consider well-implemented.

- **Calendar** - A less-used feature, which the developers think is less valued.

- **HQ** - A feature that is well-used, but which the developers think the users have problems finding/using. However, the high usage frequency might be a symptom of users having to switch quality a lot, which might deteriorate their satisfaction.

Illustrations of the features along with an accompanying descriptive text (in Swedish) can be found in Appendix A.

These motivations are quite different, in terms of the impression that the developers have of the features. Using our visualization tool, we produce the graphs seen in Figures 5.3-5.6. Looking at these graphs, there seems to exits some inter-feature heterogeneity as well, from the usage frequency distribution pattern to the percentages of users and sessions interacting with the features. The feature patterns seem different enough to be able to make a meaningful comparison later.

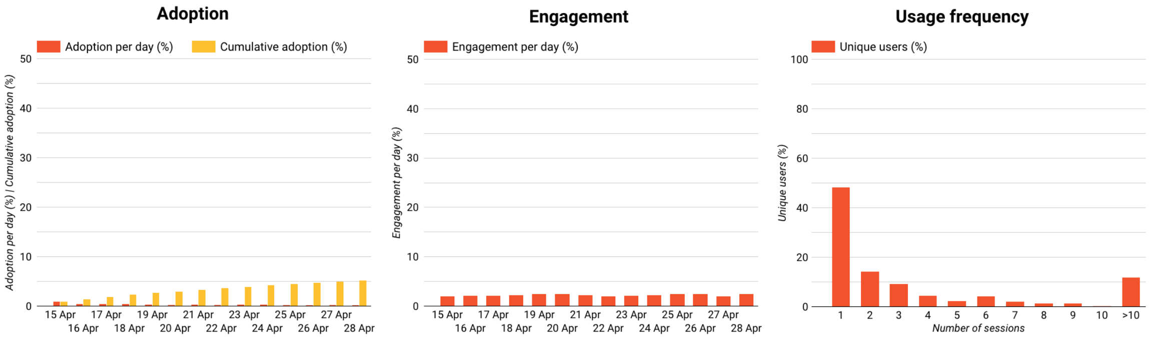


Figure 5.3: Data Studio evaluation criteria graphs for the event list feature.

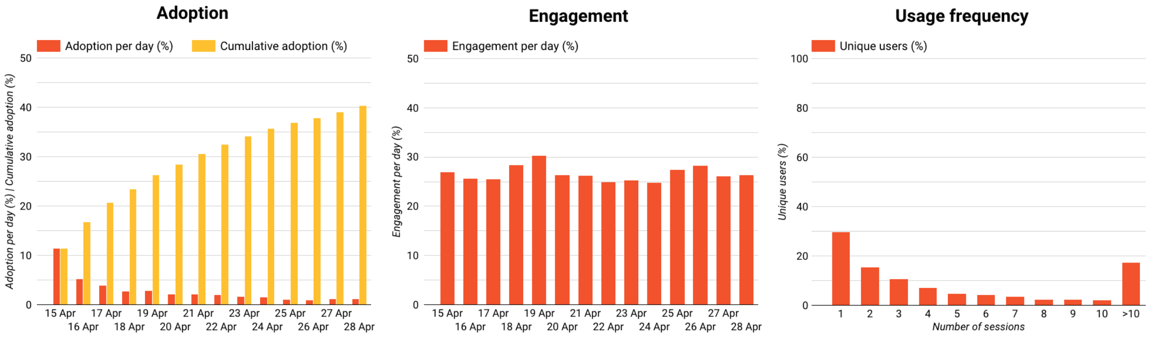


Figure 5.4: Data Studio evaluation criteria graphs for the timeline feature.

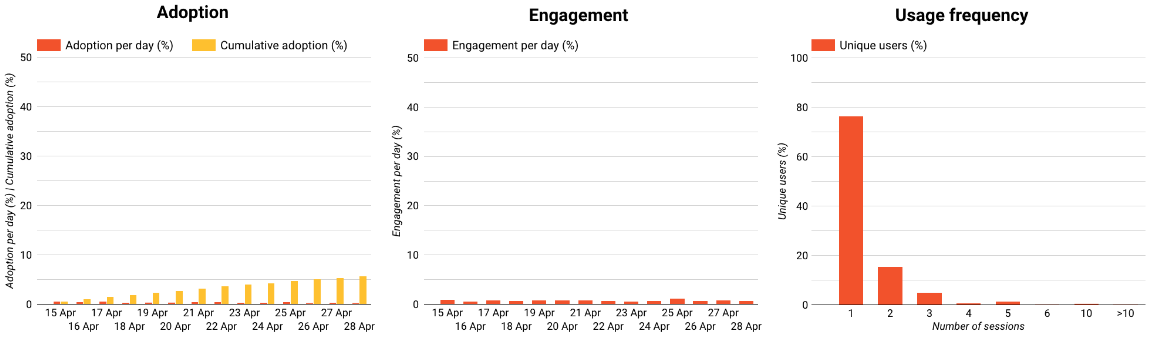


Figure 5.5: Data Studio evaluation criteria graphs for the calendar feature.

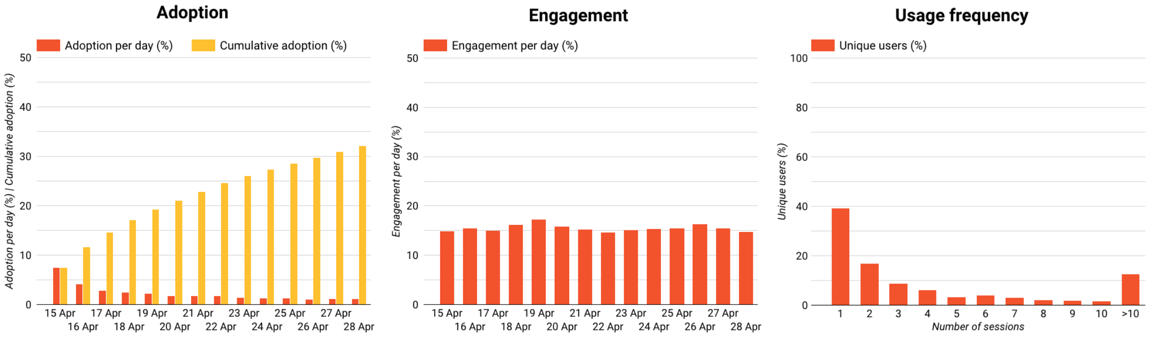


Figure 5.6: Data Studio evaluation criteria graphs for the HQ feature.

5.1.7 Our hypothesis

We propose that high relative values for all the metrics will indicate feature success, namely that the feature is valued and liked by the end-user. As mentioned before, high adoption indicates that users find and value the feature, while high engagement and usage frequency indicate that users value and/or appreciate the feature. However, due to external factors, we find that it is hard to put a value on a feature based on its metrics. Therefore, we propose comparing features instead. For example, if feature X has better metrics than feature Y, we say feature X is more successful than feature Y. For illustrative purposes, compare the overall appearance of Figure 5.4 and Figure 5.5.

5.2 A/B Testing

The implementation and deployment of an A/B-test are carried out within the scope of this thesis. The usage data of each user group will then be used as input to the analysis work related to the previously conceived hypothesis, in an attempt to proof-of-concept the conduction of A/B-testing to evaluate a defined hypothesis.

5.2.1 The idea

Following talks with our AXIS thesis supervisor, an idea eligible for testing was identified. The UX-lead had, in cooperation with developers of the ACS application, already created some early drafts of different ways to inform the end-user about new features. Evaluating different ideas comparatively to each other (and also to a control group), perfectly matches the prerequisites needed to conduct an A/B-test, as previously discussed in Section 2.5.

The feature that the end-user should be made aware of in this test is a feature released in December, about three months before the deployment of the A/B-test. The user base has not in any way been informed about the existence of the feature. The reason a tutorial is deemed necessary is that the feature usage is quite low, and the developers think the users have a hard time finding it.

The feature itself is a slow-motion playback of recorded video clips in the application, activated by long pressing in the video area. A screenshot of the playback screen of the appli-

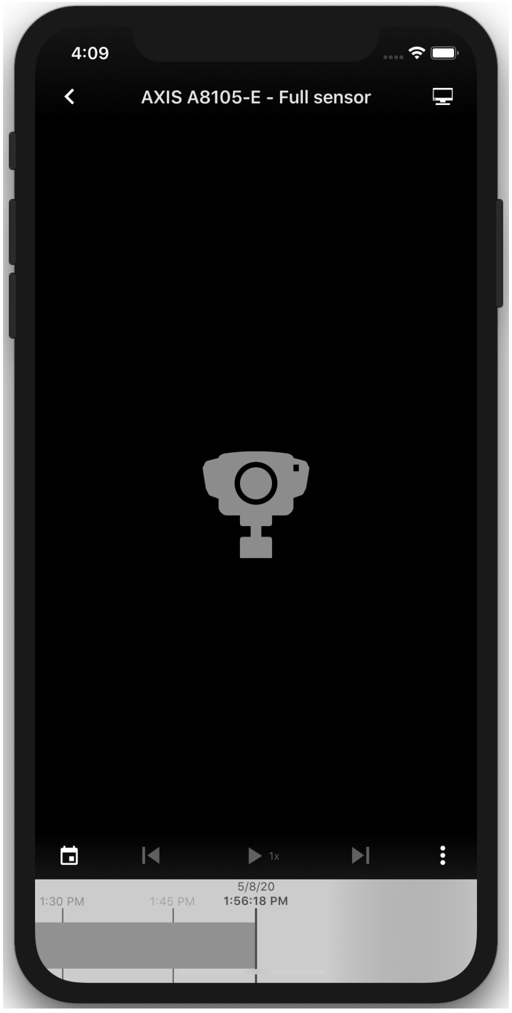


Figure 5.7: The playback screen of the iOS version of the AXIS Camera Station application. Long pressing in the video area triggers the slow-motion playback effect.

cation can be viewed in Figure 5.7 The application selected for conducting the experiment on is the iOS version of the ACS application. The UX-lead has finalized the drafts of the different ways to inform the user about the slow-motion feature before handing us the reins in terms of implementing everything needed to deploy the experiment.

5.2.2 A/B-testing framework

The analytics framework used by the mobile application department at AXIS, presented in Section 4.2, is the Google Analytics service on the Firebase platform. The capabilities of the platform were further investigated, and it was quickly discovered that it provides support for conducting A/B-tests. Firebase's **Remote Config** permits developers to set the values of predefined parameters directly in the Firebase web console, which the software application then fetches, providing the option to change the behavior of the application without having to deploy an update [16]. Targeting which users, or how many users receive what value for the defined parameter, constitutes a way of experimenting with controlled roll-outs of new functionality. As such, Firebase enables developers to conduct A/B-tests by using the Remote Config service [15]. Assigning users to groups, for example, A, B, and control, and thereafter sending each user a value for the Remote Config parameter corresponding to the group assignment, which in the application triggers the group-specific behavior, constitutes the integral parts of an A/B-test. A flow chart of how the assignment of test groups by Firebase works can be seen in Figure 5.8. Through logging analytic events tracking user behavior, evaluation of the A/B test can be analyzed in the Firebase web GUI, and the best feature (driving the most desired results) can be selected.

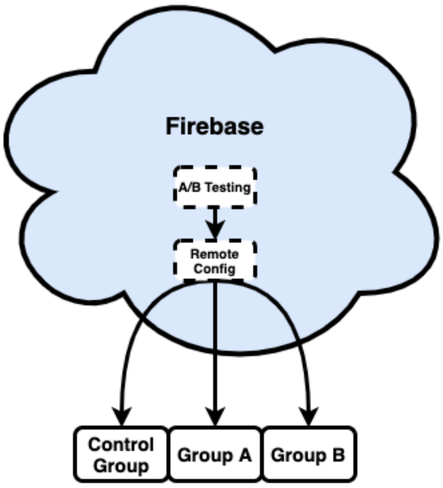


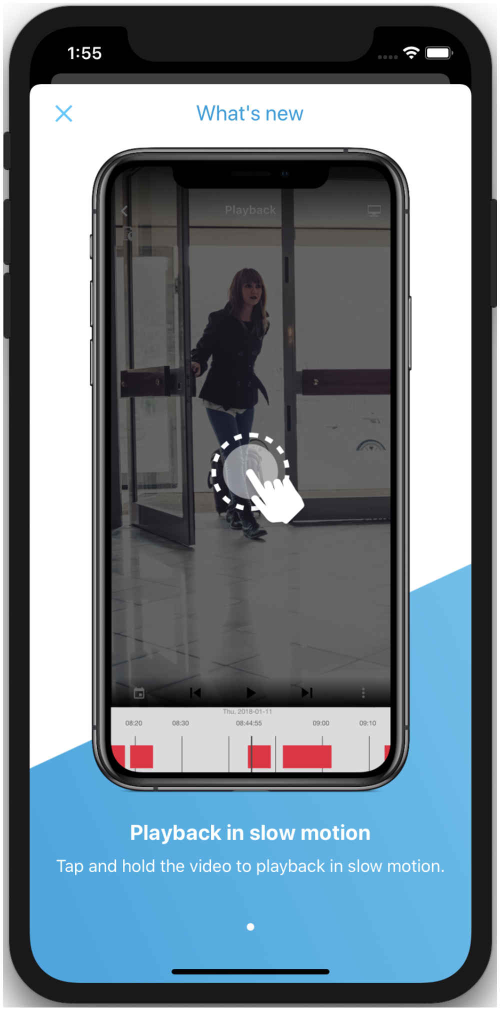
Figure 5.8: How A/B tests are distributed by Firebase.

5.2.3 The UX draft

As previously mentioned, the basic idea of the A/B-test is to evaluate which way the end-user should be informed about a new feature. The UX resources of the department were then devoted to finalizing a draft, ready for us to implement and deploy to production.

Variant A of the test is at the department know as *What's New*. This variant consists of a dialog listing the new features of the current version of the application, which is presented when the user starts the application. It is only supposed to be displayed one time; when a user dismisses the dialog, it should be regarded as read and thus not presented again to avoid cluttering the user experience with redundant pop-ups. The What's New dialog has been used in other applications of the department, but not in ACS, the target application of the test. The visual presentation logic has already been implemented on the application, but it has never been used. For reference, the UX draft of the What's New dialog can be viewed in Figure 5.9.

Variant B, the second way to inform the end-user about novel features, is an animated tutorial showcasing how to use the slow-motion feature. While the variant A dialog is displayed upon entering the application, variant B is displayed in adjacency to the slow-motion feature itself. When the user navigates to the video playback screen of the application for the first time, the animated tutorial launches. For visual reference, see Figure 5.10. To exit the animation, the user presses the 'Got it' button, in the upper left corner of the screen. Much



What's new

Playback in slow motion

Tap and hold the video to playback in slow motion.

Figure 5.9: The *What's New* dialog informing the end-user about a new feature, representing variant A of the A/B-test.

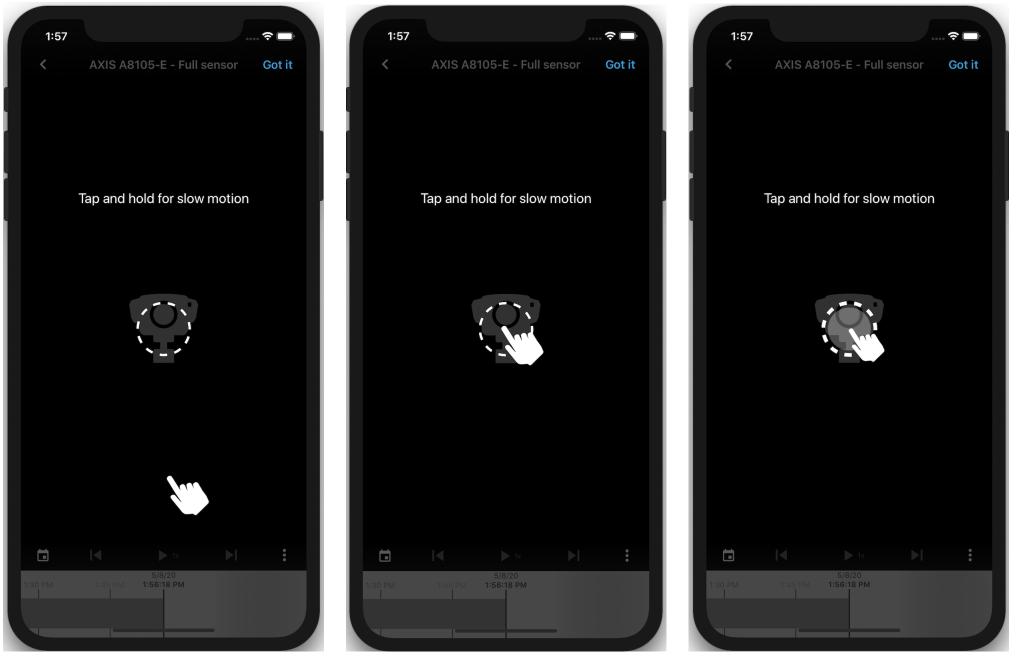


Figure 5.10: The *slow-motion tutorial* animation, visually showcasing and informing the end-user about the slow-motion feature. This animated tutorial represents variant B of the A/B-test.

like variant A, the information given by variant B is then considered to be consumed; thus, the tutorial is only displayed once per user.

5.2.4 Test design and hypothesis

Experiments introducing changes that may deteriorate the user experience may advantageously be conducted on a smaller population of the user base as a form of risk mitigation. This is not thought to be the case for our A/B-test, instead, it can be considered beneficial to include a high number of users, in order to potentially generate higher confidence conclusions based on the larger sample. A percentage (the exact percentage of total users is omitted for business-confidentiality purposes) of the user base of the iOS version of the ACS application will be divided evenly across variant A, B, and the control group (which will not receive any information about the feature).

The main goal of this A/B-test is to find out how information about a feature should be presented to the end-user. Since the information is a sort of a meta-feature, the event analyzed will be the `ax_playback_slowmotion`, which is logged when the slow-motion feature is activated. Given this circumstance, the **adoption** metric, as presented in Section 2.3 is believed to be the best metric available that the test should be evaluated on. Does A or B differ from the control group or each other, in terms of making the users try out the feature? As a secondary learning outcome, it would be interesting to learn if any of the test variants not only makes the users try out the feature to a greater extent but also makes the user continue to use the feature with higher intensity. The **engagement** metric, measuring distinct sessions engaging with a feature, will be used to evaluate this secondary learning goal.

The A variant, consisting of the *What's New* dialog, is theorized to potentially drive more users into the playback view of the application, based upon the fact that the dialog is presented at the top level of the applications view hierarchy, a view which all users visit. This could potentially incite users to navigate to the playback view to a greater extent.

The animated tutorial (variant B) is instead shown when the user on its own navigated into the playback view, positioned further down in the view hierarchy. When asked, most employees at the department propose that the animated tutorial could result in higher adoption rates, since presenting information about the feature in the vicinity of the feature itself could increase the chances of users actually attempting to use it.

5.2.5 iOS implementation

The implementation of the A/B-test on the iOS client will be carried out using XCode, an integrated development environment for developing iOS applications. The primary language used will be Swift, although some Objective-C code will also be written.

The Remote Config-functionality, fetching, and handling A/B-test group assignment of the individual app installations will be implemented first. Once this module successfully communicates with the Firebase platform, the A and B variants will be implemented. Since the visual framework needed to present variant A to the user has already been implemented (but never used), all we need to do is to handle presentation logic based on the applications test group assignment. Visual resources and tech writing is provided by the Mobile Applications UX-team and tech writers respectively, which we then will insert into the application. Variant A will then be considered as implemented. Variant B, the animated tutorial, will be implemented using the Lottie library for iOS [3]. Using this library, the animation designed by the UX-lead will be easily inserted into the application. Presentation logic based on the application's assigned test group (handled by the Remote Config module), will then be added.

For the scope of this master thesis, further technical implementation details of the A/B-test are omitted.

Chapter 6

Evaluation Phase

6.1 Evaluation criteria evaluation

Upon the end of the development and analysis phase, the task of evaluating the validity of the model and hypotheses remains. Leaning on the HEART framework, conducting surveys seems to be a good option for determining what the users actually think about the features in the previous phase. Two different types of survey methods were determined to be our next step forward, for the purpose of generating data to evaluate our evaluation criteria.

6.1.1 Online surveys

To be able to collect evaluation data, we create an online survey for pilot sites and beta testers. Lead customers describe real actual users that cooperate with the organization, while the beta testers consisted of internal axis employees (mobile applications department employees were excluded). Hence, the respondents were selected as a convenience sample. The survey concerns the four features described in Section 5.1.6. We first pose a few general questions to profile the respondents, and then we ask them to estimate their feature usage and their opinion about the selected features. If they do not use a feature, they are asked why not. The full results compiled from Google Forms (in Swedish) can be found in Appendix B and a flow chart describing the structure of the survey can be seen in Figure 6.1.

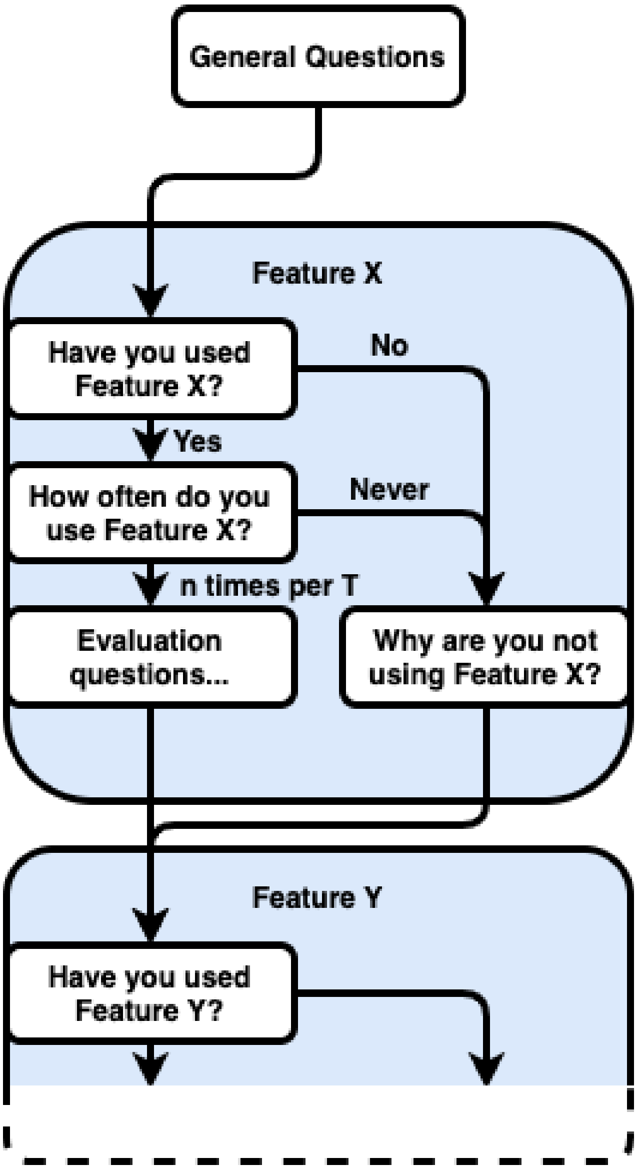


Figure 6.1: Flowchart describing the structure of the online survey.

Feature evaluation

The respondents are first presented with a picture of illustrating the feature with a descriptive text. From this, the respondent should be able to recognize the feature if they have used it. If they have, they are then asked how frequently they use the feature. Given that they use it sometimes, they are then asked to evaluate the feature. We pose two questions to capture user happiness, where we want to cover both the perceived importance of a feature, **value** and the experience of a feature, **satisfaction**:

We present them with a five-level Likert scale, where 1 corresponds to low value/negative experience, and 5 corresponds to great value/positive experience.

Given that they do not use the feature, we ask them why. We present the alternatives:

- I do not need it.
- I was not aware of its existence.
- I do not understand how it is used.
- I do not like it.

We believe that these reasons are exhaustive in terms of why a feature currently is not being used, and answers the questions posed in Section 4.3, but as a safety measure we also allow free-text reasons as answers.

6.1.2 Evaluating the results

Results

To be able to compare the features in a concrete way, we have to condense our evaluation criteria graphs into a single number for each usage metric, as seen below.

- **Adoption:** We divide the number of adopters of the feature at the end of the time period by the total number of distinct users.

$$\frac{distinct\ users}{total\ users} = adoption$$

- **Engagement:** We divide the number of sessions where the feature has been used by the total number of sessions.

$$\frac{feature\ used\ sessions}{total\ sessions} = engagement$$

- **Usage frequency:** We compute an average of the usage frequency, truncated at the maximum bin number. We then scale it by the inverse of the maximum bin number to get a number between zero and one.

$$\frac{1}{b_{max}} \sum_{k=0}^{\infty} min(k, b_{max}) \cdot u_k = usage\ frequency$$

b_{max} = the maximum bin number, u_k = the number of users that have used the feature k times.

To be able to compare the features as described in Section 5.1.7, we will use majority voting. We will order the features based on their value for each metric. The placement in this ordering will then be used for voting, and the features will be ranked for which their placement has the majority vote (see example in Table 6.1).

	Feature X	**Feature Y**	**Feature Z**	**Ordering**
Adoption	0.3	0.5	0.1	Y > X > Z
Engagement	0.7	0.3	0.1	X > Y > Z
Usage frequency	0.6	0.2	0.1	X > Y > Z
Majority vote for placement	1	2	3	X > Y > Z

Table 6.1: Example of majority voting

Survey

Using the survey results, we compute usage metric numbers in a similar way as we did for our evaluation criteria results.

- **Adoption:** We divide the number that answered that they used a feature by the total number of respondents.
- **Engagement:** We divide the sum of what the respondents answered that they use the feature each month, by the sum of what the respondents answered that they use the application each month.
- **Usage frequency:** We compute the average of what the respondents answered that they use the feature each month. We then scale it by the inverse of the maximum bin number.

These three numbers will be used to show how well the respondents of the survey represent the user base as a whole.

We also compute numbers for value, satisfaction, total value, and total satisfaction for each feature. For simplicity's sake, we calculate means for the Likert scales.

- **Value:** The average of the answers to the question ”What value does feature X have to you?”
- **Satisfaction:** The average of the answers to the question ”What is your general experience of feature X?”

Looking at the measured value/satisfaction only captures part of the value/satisfaction aspect for the particular feature. We argue that it also could be important to account for the value/satisfaction, or rather the lack of it, the feature brings to users that do not use it. For this purpose, we also look at total value and total satisfaction, by extrapolating the answers to why respondents don't use a feature to the value and satisfaction categories.

- I do not need it. $\Longrightarrow$ **Low value.**
- I was not aware of its existence. $\Longrightarrow$ The feature does not communicate its existence. $\Longrightarrow$ Full potential value is not reached. $\Longrightarrow$ **Low value.**
- I do not understand how it is used. $\Longrightarrow$ The overall feature experience is confusing. $\Longrightarrow$ **Low satisfaction.**
- I do not like it. $\Longrightarrow$ **Low satisfaction.**

With this extrapolation, we can create the two final numbers.

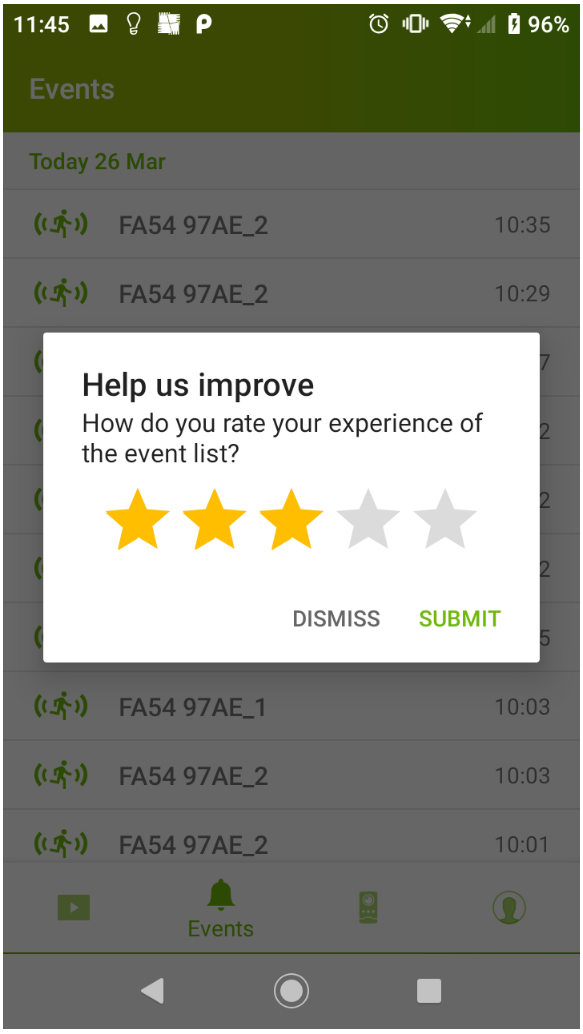
- **Total value:** The average of the answers to the question ”What value does feature X have to you?”, also including the ”Low value” answers as having a score of 0.

- **Total satisfaction:** We compute the average of what the users answered that they use the feature each month, also including the ”Low satisfaction” answers as having a score of 0.

These four opinion numbers will be used to support or reject the hypotheses we draw from our results.

6.1.3 In-app survey

Another way to find out what the users actually think of features is to ask them directly in the application. To investigate a feature using this strategy, an in-app survey will be deployed for the event list feature. The in-app survey dialog can be seen in Figure 6.2. The dialog is presented to the user when the event list has been used five times. This should make sure that the user understands what the event list is used for and that the user is not just accidentally using the feature. The user can then input a grade between one to five stars (a Likert scale), indicating the experience the user had when using the feature, or dismiss the survey.



Today 26 Mar

FA54 97AE_2 10:35

FA54 97AE_2 10:29

Help us improve

How do you rate your experience of the event list?

FA54 97AE_1 10:03

FA54 97AE_2 10:03

FA54 97AE_2 10:01

Figure 6.2: The in-app survey dialog, asking users to rate their experience using the event list feature.

Finding usage patterns in responses

Given a high number of responses and a diverse opinion of the feature, we will be able to segment users based on opinion. We will then be able to produce graphs and results and see if there is a significant difference in usage patterns between high raters (4-5 stars) and low raters (1-2 stars). Furthermore, we will also produce results for those who dismiss the survey. We will employ the same methods for comparison as in Section 6.1.2. The time period chosen for results will take place after the final survey response has been collected, to see if respondents continue using the feature.

6.2 A/B test evaluation

We will not have access to any qualitative evaluation data concerning user value or satisfaction for neither the A/B test or the slow-motion feature itself. We instead make a relative evaluation regarding the different implementations of the way to inform about this feature.

This evaluation will aim to indicate the best implementation based on the selected evaluation criteria.

6.2.1 Statistical tests

The statistical reasoning and basis for this part can be reviewed in Section 2.8, where applicable statistical concepts are discussed. We aim to empirically evaluate the conducted A/B-test using two types of data for the A, B, and control test groups. The statistical power of any eventual test group deviations will be calculated and presented.

The first type of test uses session-based feature interaction count data as a basis, which in the case of our metric will correspond to cumulative **engagement**. The outcome of the stochastic count variable X is retrieved from the usage data set, which can be expressed as observing the outcome of a Poisson process:

$$X \sim P(\lambda; t)$$

where λ denotes the intensity with which events occur, observed during time period t. We obtain an estimate of λ, denoted $\hat{\lambda}$, according to

$$\hat{\lambda} = \frac{1}{t} \sum_{i=0}^{t} x_i$$

where x_i represents the observed count outcome at the time i.

Assuming a large enough $\hat{\lambda}$, X approaches normality (see Section 2.8),

$$X \sim N\left(\hat{\lambda}, \sqrt{\hat{\lambda}}\right)$$

a two-tailed Welch's t-test can be used to check the statistical significance of the sample's mean deviation. This is achieved by constructing a small Python script (see Appendix C).

The second type of test is based upon the **adoption** metric. This type of data is binary (non-adopter or adopter) for each test group, and an outcome can be expressed as observing the outcome of a binomial distribution. Since the comparisons are made pairwise, and the data is binary, contingency tables can be produced for each pair, and the statistical significance of the outcomes can be computed using the χ^2 test. The area of interest, from the evaluation perspective, is the frequency distribution of these values across different groups. Higher adoption of the feature in one group, which is statistically significant, would indicate that a test variant is superior in making the users try out a feature.

Three 2x2 contingency tables are produced, using the observed adoption data for respective groups. Another Python script is used in order to implement this test (see Appendix C).

Chapter 7

Results

7.1 Evaluation criteria results

7.1.1 Relevant online survey results

For the online survey, 22 users responded out of the 70 potential respondents. The processed answers from the online survey are shown in Tables 7.1 and 7.2, obtained as described in Section 6.1.2.

In Table 7.1, the adoption of the features is displayed, as well as the reason for non-adoption. Important to note here is that all non-adopters did not adopt due to low value (with the reasoning explained in Section 6.1.2). This results in the total satisfaction score being the same as the satisfaction score.

Table 7.2 shows the monthly frequency and session count for the different features, as well as the total session count for the application as a whole. These are used for computing the engagement and usage frequency metrics for the survey.

The resulting numbers for usage data are shown in Table 7.3, and for respondent opinion numbers in Table 7.4. For a more detailed summary of the survey answers, a print-out of the Google Forms summary can be found in Appendix B (in Swedish).

	Event list	**Timeline**	**Calendar**	**HQ**
Adopters	12	22	13	13
Non-adopters (Low value)	10	0	9	9
Non-adopters (Low satisfaction)	0	0	0	0

Table 7.1: Online survey respondent adoption of the different features, presented alongside an aggregated of non-adoption reasons.

	Event list	**Timeline**	**Calendar**	**HQ**	**App**
1-5 (avg: 3)	6	5	10	5	
5-10 (avg: 7.5)	2	6	0	3	
>10 (avg: 10)	4	10	3	5	
Total Monthly Sessions	63	153	57	87.5	238

Table 7.2: Respondent usage frequency per month for each feature as well as total monthly sessions for the online survey.

	Event list	**Timeline**	**Calendar**	**HQ**
Usage frequency	0.304167	0.370325	0.23085	0.336538
Engagement	0.264706	0.642857	0.239496	0.367647
Adoption	0.545	1	0.591	0.591

Table 7.3: Computed evaluation criteria numbers for the usage metrics based on the response data of the online survey.

	Event list	**Timeline**	**Calendar**	**HQ**
Value	3.92308	4.63636	3.38462	4.38462
Satisfaction	4.07692	4.09091	3.92308	4.23077
Total value	2.31818	4.63636	2	2.5909

Table 7.4: Computed numbers for the opinion metrics base on the response data of the online survey.

7.1.2 Post-deployment data evaluation criteria

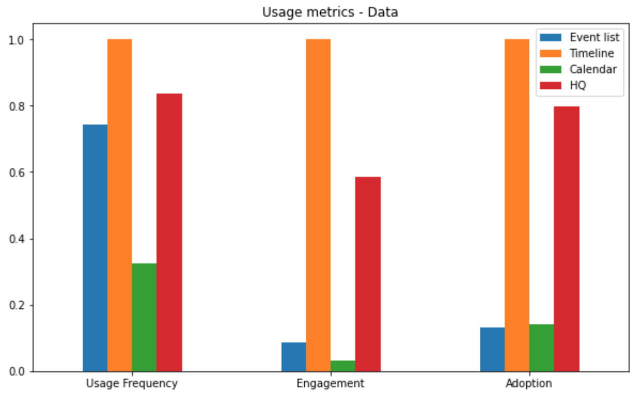
Due to the limited response rate of Android users, post-deployment data was only collected from iOS users. The computed evaluation criteria numbers can be viewed in Table 7.5, based on the usage data presented in Figures 5.3-5.6.

	Event list	Timeline	Calendar	HQ
Usage frequency	0.32	0.43	0.14	0.36
Engagement	0.0229	0.2676	0.0081	0.1562
Adoption	0.0526	0.4031	0.0569	0.321

Table 7.5: Computed evaluation criteria numbers for the usage metrics, based on collected post-deployment data.

7.1.3 Ordering by majority vote

The majority vote ordering of the features, together with normalized graphs for all results are presented in Figures 7.1, 7.2 and 7.3. The means of the normalized metrics can be viewed in conjunction with a summary of the majority vote ordering in Figure 7.4. We abbreviate the feature names with their first letter, i.e event list becomes "E".

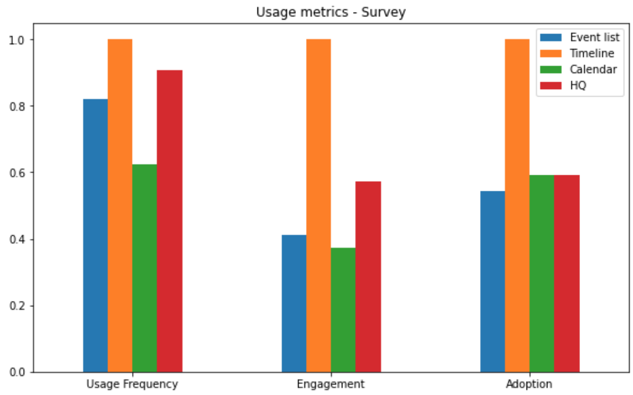


(a) Normalized usage metrics for post-deployment usage data.

	Ordering
Usage frequency	T > H > E > C
Engagement	T > H > E > C
Adoption	T > H > C > E
Results	T > H > E > C

(b) Relative feature ordering obtained by majority voting, using usage metrics for post-deployment usage data.

Figure 7.1: Relative feature ordering based on normalized usage metrics for post-deployment data.

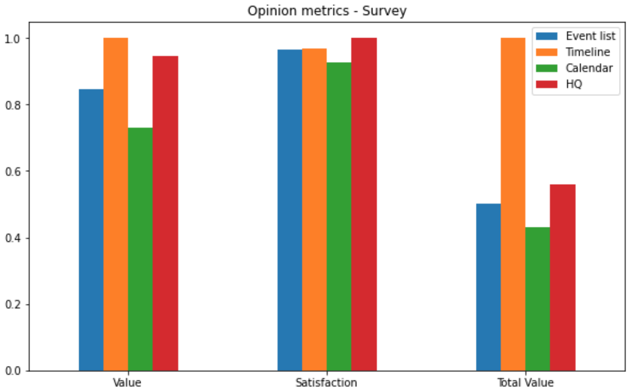


(a) Normalized usage metrics for online survey data.

	Ordering
Usage frequency	T > H > E > C
Engagement	T > H > E > C
Adoption	T > C = H > E
Results	T > H > E > C

(b) Relative feature ordering obtained by majority voting, using usage metrics for the online survey data.

Figure 7.2: Ordering based on normalized usage metrics for the online survey data

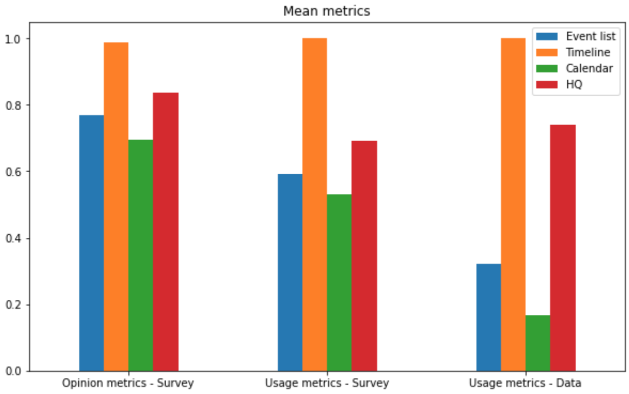


(a) Normalized opinion metrics for the online survey data.

	Ordering
Value	T > H > E > C
Satisfaction	H > T > E > C
Total value	T > H > E > C
Results	T > H > E > C

(b) Relative feature ordering obtained by majority voting, using opinion metrics for the online survey data.

Figure 7.3: Feature ordering based on normalized opinion metrics of the online survey answers.



(a) Mean normalized metrics

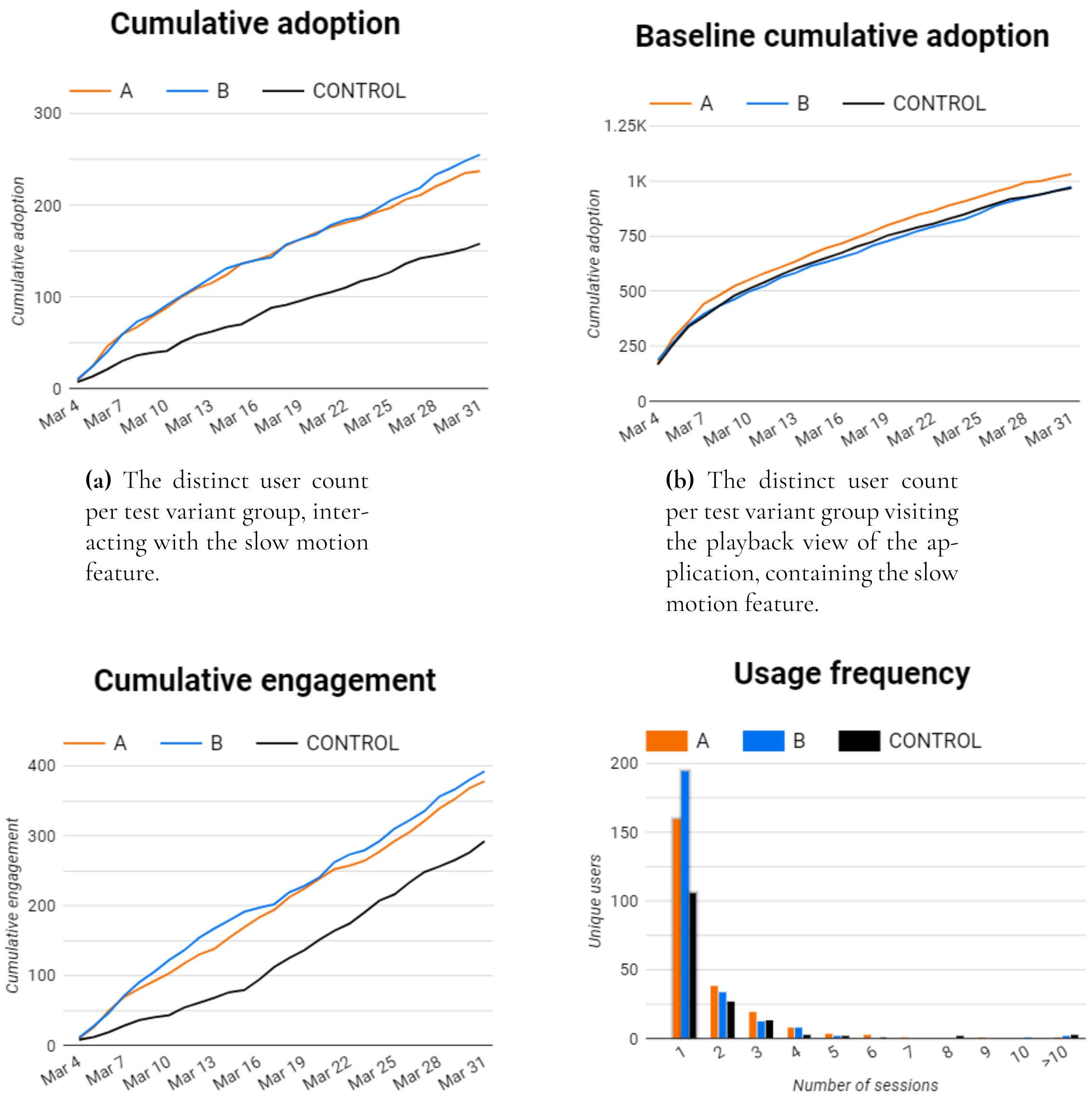
	Ordering
Opinion	T > H > E > C
Usage - Survey	T > H > E > C
Usage - Data	T > H > E > C

(b) Final feature ordering for the post-deployment feature usage data, the online survey usage data and the online survey opion metrics.

Figure 7.4: Usage and opinion metrics in contrast to the final ordering for each metric type, based on post-deployment and online survey data.

7.2 A/B-test

This section presents the outcome of the A/B-test itself in terms of our evaluation criteria, shown in Figure 7.5. Relevant data for calculating the statistical power of the test variants can be viewed in Table 7.6, for the χ^2 test and Welch's t-test respectively. Statistical tests were conducted for both the 14 day and 28 day time periods, counted from the release of the test to the end-user.



(a) The distinct user count per test variant group, interacting with the slow motion feature.

(b) The distinct user count per test variant group visiting the playback view of the application, containing the slow motion feature.

(c) Accumulated sessions of the application, containing at least one slow motion feature interaction.

(d) Displays the number-of-session distribution for the users, in terms of sessions including the slow motion feature interaction.

Figure 7.5: The results of the A/B-test from the 28 days following the release of the test, in terms of the evaluation criteria.

χ^2 test data			
	A	**B**	**C**
Adopters$_{14}$	146	143	88
Non-Adopters$_{14}$	597	531	614
Total$_{14}$	743	674	702
Adopters$_{28}$	237	255	158
Non-Adopters$_{28}$	794	719	810
Total$_{28}$	1031	974	968

Welch's t-test data			
	A	**B**	**C**
λ_{14}	13.86	14.43	8.0
λ_{28}	13.5	14.0	10.42

Table 7.6: Resulting data extracted for the 14 and 28 day time periods following the release of the test, constituting the input values for the χ^2 test and Welch's t-test. Note that the experiment was conducted on a subset of the total user base of the application.

	A-B	A-C	B-C
Welch$_{14}$	0.69	$8.9 \cdot 10^{-5}$	$3.4 \cdot 10^{-5}$
χ^2_{14}	0.51	$3.2 \cdot 10^{-4}$	$2.3 \cdot 10^{-5}$
Welch$_{28}$	0.62	$1.6 \cdot 10^{-3}$	$3.4 \cdot 10^{-4}$
χ^2_{28}	0.11	$2.3 \cdot 10^{-4}$	$1.5 \cdot 10^{-7}$

Table 7.7: The resulting p-values obtained from the statistical tests performed on the different test variants, for the 14 day and 28 time period data.

7.3 In-app survey segmentation

The results for the in-app survey segmentation can be viewed below. The response count for the different user groups (based on happiness) is shown in Table 7.8. The relevant evaluation criteria for the data obtained are presented in graphs in Figure 7.6. The evaluation criteria numbers are then presented in Table 7.9 and displayed in comparison in a graph in Figure 7.7. The time period chosen for the evaluation criteria is the two weeks after the final answer of the survey had been collected; 22 April - 5 May.

	Count
Happy (4-5 stars)	23
Neutral (3 stars)	1
Unhappy (1-2 stars)	0
Dismiss	55

Table 7.8: In-app survey response counts between April 1 and April 21

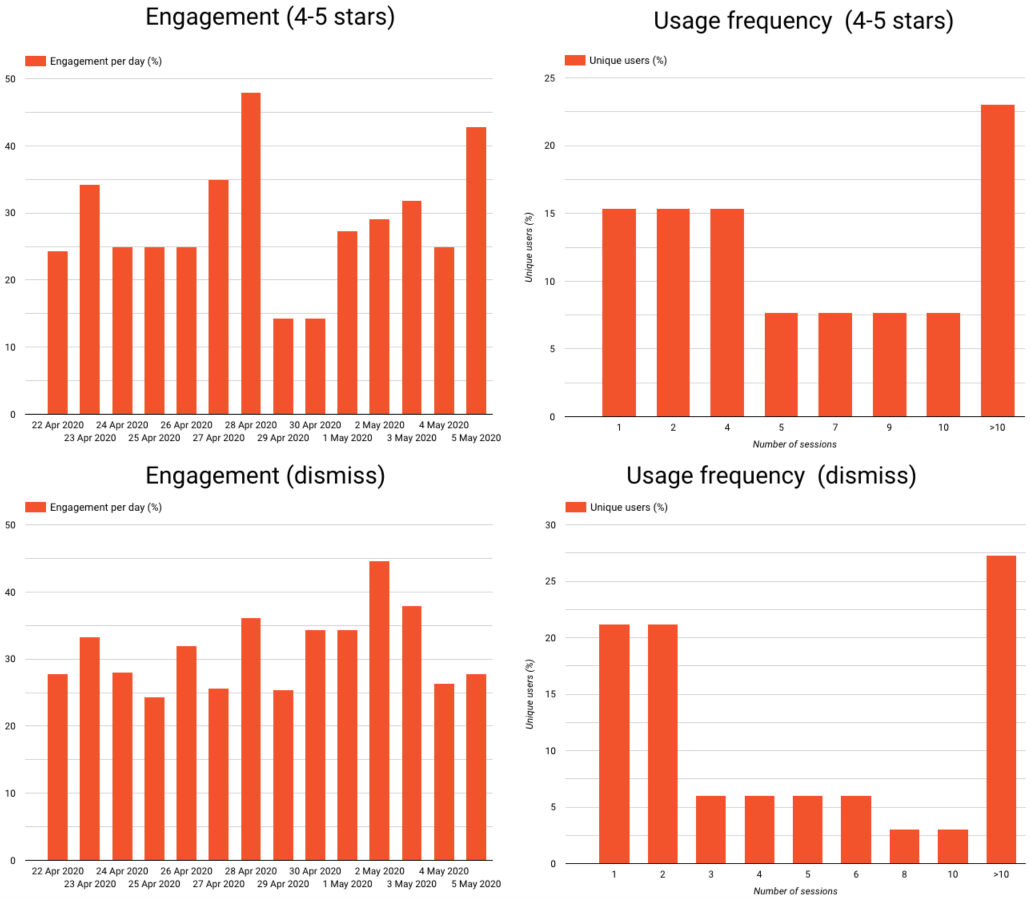


Figure 7.6: Data Studio evaluation criteria graphs for relative engagement and relative usage frequency, segmented on happy users and dismissers.

	Happy	**Dismiss**
Usage frequency	0.577	0.5
Engagement	0.281	0.301

Table 7.9: Computed evaluation criteria numbers for the usage metrics, based on collected post-deployment data.

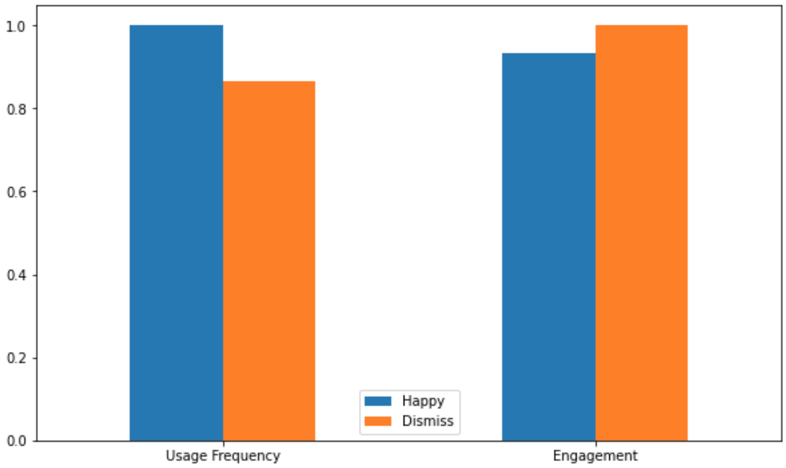


Figure 7.7: Normalized usage metrics for post-deployment usage data.

Chapter 8
Discussion

8.1 Evaluation criteria

If post-deployment data can be used to find feature success, it will result in a faster feedback loop, where developers can quickly find which features need further development without having to resort to customer contact, which solves the ”open-loop”-problem. This also shifts the development process to become fully data-driven, pushing developers to make informed decisions based on data and not opinion. Our ranking system attempts to find a link between Happiness and Engagement, Adoption, and Retention in the HEART-framework, where maximizing happiness will maximize feature success, as presented in Section 5.1.1.

The final results for the evaluation criteria, seen in Figure 7.4, show that a pattern emerges when condensing the evaluation criteria graphs into numbers and comparing them to the survey results. Through majority voting for each type of metric, we get the same relative ordering: Timeline > HQ > Event list > Calendar. This means that the timeline is ordered as the best feature, HQ the second-best, etc. This ordering confirms the opinions of the developers concerning the timeline and the calendar, which are ordered as the best and the worst respectively (see Section 5.1.6). Since the ordering is the same for both the usage metrics and the opinion metrics, we draw the conclusion that our usage metrics could indicate user value and satisfaction. For example, if feature X has a majority of better evaluation criteria numbers than feature Y, feature X gives more user value and satisfaction than feature Y. Consequently, in line with the business goals of AXIS, feature X is then more successful than feature Y.

However, there are many factors impacting our results that make this conclusion quite weak (see Section 8.1.3 for discussion).

8.1.1 Detailed comparisons

Comparing Figure 7.1 (usage metrics - post-deployment data) and Figure 7.2 (usage metrics - survey), we find very similar patterns for the metrics, although the intra-metric variance for the post-deployment data is much bigger. The most interesting observation here is the discrepancy between the usage frequency and the other evaluation criteria numbers for the event list, where the usage frequency is relatively high. The event list is a new feature, and has, as expected, low adoption and engagement. We propose that this indicates that the users that have started to use the event list appreciate it, since a high usage frequency score indicates a high relative number of users that use the feature often. Another interesting observation is that adoption for HQ is comparatively low for the metrics from the survey, tying with calendar, partially confirming the suspicion of the developers that HQ is hard to find (see Section 5.1.6). Note that this is not the case for the post-deployment data.

The opinion metrics for value and total value, seen in Figure 7.3, show a clear pattern similar to the two previous usage metrics. Satisfaction has very little variance due to most respondents rating their satisfaction as either 4 or 5.

8.1.2 Assumptions and generalizations

The responses from the online surveys are mostly from iOS users (19 out of 22 respondents). Since the responses from Android users are few and we want as many respondents as possible for evaluation purposes, we group them together with the iOS users as we assume that they use the application in the same manner. This means that we only consider the iOS version of the application and only present usage metrics for post-deployment data for iOS. We also make some generalizations concerning some of the free-text answers for some of the questions. For example, a user that answered ”I got the impression that HQ was automatically on” on the question ”Why do you not use HQ?” was categorized as ”I didn’t know it existed”, and was interpreted as a ”low-value” answer.

We also use the means of the survey responses for the opinion metrics. As mentioned in Section 2.7, Likert scales are generally not supposed to be condensed into mean values, since they do not represent relative data points, and the correct way would be to use medians and ranges instead. We forego best practice to instead be better able to differentiate the opinion metrics between the different features, giving fewer ties in the ordering.

8.1.3 Limitations and problems

There are several elements that impact our results negatively. The most glaring issue is using the online survey as a representative sample of the opinions of the whole user base. First of all, the respondents are very few. Having only 22 respondents, we cannot say that we have anything near a robust sample. Furthermore, these users are a very specific subset of the user base, most of them beta-testers employed at AXIS, which for example could provide them with useful internal information, and having very specific user cases, not representative of the average user. The low variance of the satisfaction answers makes it hard to draw any sort of conclusion based on user satisfaction, and the ranking represents primarily user value. Also, the way we order the result does not take into consideration the great intra-metric difference between the features and could oversimplify findings. For example, the adoption difference between the calendar and the event list is much smaller than the one between the calendar and HQ.

8.2 In-app survey

Being able to label satisfied users in the collected post-deployment data would be very beneficial in terms of finding feature usage patterns that correspond to feature success. Having access to these labels, segmentation on satisfied and dissatisfied users could be done, potentially giving valuable information input to the organization.

The segmentation for the in-app survey is done by grouping happy users and users that dismiss the survey. The evaluation criteria graphs are condensed into evaluation criteria numbers for the metrics to investigate if any patterns can be found. Due to problems using baselines for adoption, we excluded it for this comparison. However, the metrics found were very similar, no patterns could be detected, and no ranking could be done.

8.2.1 Limitations

The number of responses is lower than expected. Hoping for a large response number giving us a good sample of the event list users, only 24 users respond to the survey, and 55 users dismiss it. Furthermore, there are no users rating their experience as bad (1-2 stars). That is why we compare dismissers to happy users; it is the only comparison we can make. We attribute the bias to happy users to the decision to have the survey only be displayed to users that have used the feature more than five times. This filters users that have used the feature a few times and not been happy with it. Also, due to implementation details, the counting of feature usage only started after the in-app survey was deployed. This removes early adopters that stopped using the feature from the response pool. Of course, the positive responses could be because of a good implementation of the feature.

8.3 A/B-test

In order to augment and improve the Learn-step of the BML loop, helping AXIS become more data-driven in their mobile application feature development, a proof-of-concept implementation and deployment of an experiment was conducted. The experiment itself randomized a subset of the real end-users on the iOS version of the ACS application, into three equally sized groups. The users of the different groups received different information about the existence of the slow-motion feature, for viewing recorded video clips, see Figure 5.9 and Figure 4.1, representing the A and B variant respectively (the control group received no information).

8.3.1 Main result

The learning outcome, based on the results, is that the users should be informed about new features, but the better way of informing users could not be distinguished in this test. In terms of getting the users to try the feature once, as well as producing higher engagement, a significant difference was found across the test group data. Using both the engagement and adoption metrics as evaluation criteria produced statistically significant results, $p < 0.05$, strongly indicating that the A and B variants were preferable to the control group, yielding both higher adoption rates as well as engagement intensity, see Figure 7.5a, Figure 7.5c and

Table 7.7. However, A and B could not confidently be said to differ in terms of the adoption and engagement metrics. The department's informal "hunches" regarding the superiority of the A and B variants, see Section 5.2.4, could not be confirmed by observing the collected data. The different locations of the information presentations were thought to have side-effects, the dialog of variant A presented at a shallow navigation layer of the application could potentially drive a higher proportion of its test group population into the playback view. While this effect can be seen in Figure 7.5(b), we investigated further, and found that variant A had a higher number of users assigned to it for some reason, canceling out any higher adoption for the baseline. Thus, this hunch is proven to be false. Neither did the adoption rates of the A and B variants differ in a statistically significant way.

As was presented in Section 2.5, internet-connected software running on clients is suitable for conducting experiments like A/B-tests. The proof-of-concept implementation and deployment of the A/B-test of this thesis is another example of this. The department can learn new things about their users and their application using empirical tests, pushing the organization even further towards in its pursuit of data-driven decision making.

8.3.2 The case for A/B-tests

Inference of causal relationships between software implementations and their estimated respective performance observed using collected post-deployment data is not always a simple matter, but A/B tests can mitigate most of the confounding factors. The thesis work has in large parts consisted of trying to figure out what conclusions one can draw from analyzing feature usage data in particular. When and if a pattern or a deviation from prior patterns is detected, what can be said about the underlying cause that manifests itself in the data? A wide array of factors can influence the feature usage data that is collected, sometimes to the degree that the noise might be indistinguishable from actual information. Patterns may differ on the temporal level; usage fluctuates depending on the time of day and day of the week. Holidays, COVID-19 lockdowns, weather conditions, network infrastructure, power outages, differing technical specifications (for example, device and operating system used) are all other possible variables to consider, which influences how the end-users usage pattern manifests itself. Power users and new users probably interact with the application in different ways, producing yet another variable to factor into the analysis. The A/B-test methodology tries to mitigate these external factors by assigning users to their respective test groups using randomization. The external factors' influence on the collected usage data should hopefully not differ across groups, which makes it more likely that the effect an intervention actually had can be established.

8.3.3 Method discussion

Taking a closer look at the resulting slow-motion feature engagement metric, Figure 7.5c, an interesting observation can be made. Looking at the slope of the test variants, before and after March 15, a mean reversion of the groups can be observed, where both the A and B variants result in a steeper engagement curve initially. This appearance can be interpreted in one of two ways, in terms of what caused it. The first interpretation relies solely on the fact that the users of the A and B test groups initially were given information about the feature, which in turn resulted in a higher intensity of sessions interacting with the slow-motion feature.

However, a measurement error might have been introduced when the test was first released. Old Firebase framework dependencies in the iOS app build that was released to the app store resulted in initial erroneous logging of the test group assignments for the individual users. Instead of logging which group a user belonged to, a null value was found in its place. A fix for this was released a few days later, which enabled a retroactive mapping of user groups. However, this might have introduced a smaller measurement error, since some users never could be mapped this way.

In any way, the engagement metric slope for the different test variants seems to experience a mean reversion; 14 days after the intervention was deployed no meaningful difference in engagement rates can be spoken of. This means that while users in the A and B groups to a greater extent tested the feature than control group users, they did not continue to frequently use the feature. Bluntly put, the slow-motion feature's value to the users could be limited since they did not continue to use it. This could be another interesting finding obtained from the A/B-test.

The statistical modeling for the engagement metric was based upon the assumption of the session count data being modeled as a Poisson process. An underlying assumption for a process having a Poisson distribution is that the intensity remains constant, it should be time-invariant. This may in practice not be entirely true, as we observed shifts in intensity manifested as mean reversions, after about 14 days after the test was released.

A final note about the statistical significance levels of the $\boldsymbol{Welch}_{14}$ test related to the control group should be made, found in Table 7.7. This value was obtained using the 14 day intensity $\lambda_C = 8.0$, as given in Table 7.6. In the theory chapter, Section 2.8, this value was required to be greater than 10 in order for the Poisson distribution to be normally approximated. This makes the underlying model less accurate, as the resulting Poisson distribution is not "as normally distributed" as we want it to be. However, the 28-day intensity values fulfill all prerequisites.

8.4 Further work

8.4.1 Evaluation criteria

Given that the conclusion for the evaluation criteria holds, a ranking representing user value could be produced using post-deployment data. This could be applied to all features, giving rise to a ranking system where the low ranking features could be identified and further investigated why they rank so poorly.

It is not easy to directly measure user value and satisfaction, using only post-deployment data as input. Several external factors could result in the wrong conclusions drawn. For example, an important feature not working 75% of the time will result in high usage, and our ranking will indicate that it is properly implemented when that is not being the case. Additionally, while the features chosen for the results could be compared well without using a baseline, it is important to consider using one when making comparisons between shallowly and deeply located features, in terms of their placement in the applications view hierarchy.

To be able to validate our model and conclusions, which would motivate its integration into the development process, we propose deploying more surveys directed at the actual users of the application, either as more extensive in-app surveys or as online surveys. Being certain that the model in fact captures user value and satisfaction, would enable quick and cheap feedback, closing the loop.

Nevertheless, the analysis of post-deployment usage data using adoption, engagement, and retention is important in itself. It shows usage patterns for different features and could be used to identify anomalies in behavior. It is a cornerstone in the analysis in A/B-tests, where the minimization of external factors impacting the results will result in robust metrics for deciding which variant is the best. It can also be an important factor in a hypothesis-driven decision making process. It could be used to answer questions, which in turn will produce hypotheses, such as ”What percentage of adoption do we expect after two weeks?”, ”What should be the average daily engagement when the feature is properly implemented?” and ”What kind of usage frequency pattern will this feature produce?”. Also, comparisons with other features using historical data could also be done, answering questions such as ”Should feature X have better usage patterns than feature Y given the same time period after release?”.

Furthermore, in-app surveys could be directed at users with a certain usage pattern. For example, users that stop using a feature X after some number of times could be targeted with a survey asking ”Why did you stop using feature X?”, or high-frequency users could be asked: ”Is feature X working properly?”. This information can then be used in the development process to find if strange patterns for a certain feature point to a problem, or to show reasons for usage patterns for certain segments in the user base.

All in all, while a link between feature success and our evaluation criteria would be a golden opportunity to make feature development fully data-driven, we cannot currently make a case for the existence of such a link. Due to all the external factors impacting different features in various ways, we find it hard to argue for a catch-all way to measure user value and satisfaction from any analysis of usage data. It might work for a case-to-case basis, but the loss in automation will lessen the positive impact in the development process, since decisions continuously need to be made in what ways external factors affect the analysis. For measuring user happiness, it is probably better to depend on the state-of-the-art way of asking questions to users, preferably by utilizing online surveys and in-app surveys. Another promising alternative as well is data mining app reviews.

8.4.2 A/B-test

A/B tests conducted by a smaller development organization might differ from how tech giants like Google or Microsoft utilize the methodology. Development resources must be channeled toward development activities generating the greatest user value, and conducting extensive A/B-tests does not come without overhead in development time and resources. The most resource-effective way would probably be to conduct A/B tests, where one of the variants was seen as the control. This would minimize development overhead, but delay the potentially ”new” version to part of the user base. Another strategy could be to deploy risky changes to smaller sets of the total user base, closely monitoring key metrics to prevent quality deterioration.

The way forward concerning the continued conduction of A/B-tests is highly connected to the general process of evaluating features and the success of feature implementations. The A/B-test methodology aims at finding superior implementation variants and selecting the most appropriate one going forward. The evaluation metric then quickly becomes key in making the better decision on which variant to proceed with. In our case, the adoption metric was easily identified as the key metric to evaluate the test variants. However, the information feature can be regarded as somewhat of a meta-feature, informing about the real functionality provided by the slow-motion feature. It would have been interesting to test two different implementations of a feature itself. Perhaps the HQ feature could have been tested, where the different test groups were assigned different video stream quality values. Choosing a good metric to evaluate this test might not have been as obvious, giving us more of an analysis.

To be able to utilize the A/B test fully, it is important to embrace the hypothesis-driven decision making presented in for example the BML-loop. With A/B-testing, it is very easy to see what differences there are after releasing the variant and reducing opacity in the development process.

The test was conducted on the iOS version of the ACS application. Results from an Android version of the application would have been interesting to compare with as well, which is something that could be attempted in the future.

8.5 Revisiting the research questions

To summarize our findings and discuss our answers to the previously formulated research questions, each research question is discussed separately in the bullet list below.

- **RQ1**: How can post-deployment data be used to evaluate the level of success of feature implementations?

  Several ways of using post-deployment data for evaluating feature success were investigated. We identified several use- and analysis objectives of post-deployment data already in place at the Mobile Applications department, such as the usage of data in tracking app stability and performance. Value-, pain point-, use path-, and user profiling analysis were identified as different ways of utilizing the collected data. The thesis was focused on the value analysis area. Novel methods to the department were then investigated and introduced: the A/B-test and the feature usage evaluation criteria.

- **RQ2**: What types of evaluation criteria could be used to evaluate features released to the end-users?

  The quality in use definition described in theory lists various measures of the effectiveness, efficiency, and satisfaction concepts that features potentially could be evaluated on. The circumstance under which the thesis work was carried out resulted in a focus on the satisfaction concept of this definition. The reason for this focus pertained to the department's current data utilization being heavily geared towards the effectiveness and efficiency measures already, making the satisfaction concept most valuable to investigate. For the department, we found that evaluation criteria capturing user value and satisfaction based on user-feature interaction data could be used as a new way of evaluating features. For this purpose, the evaluation criteria used for post-deployment data in the HEART framework: Engagement, Adoption, and Retention was investigated. Both the adoption and engagement criteria were also found to work in an A/B-test context, for testing hypotheses regarding different implementations of a feature.

- **RQ3**: How can the evaluation of implemented features be used in current software development processes?

  Using a model capturing user value and satisfaction solely using post-deployment data is incentivized in two ways; the high speed paired with the relatively low cost of obtaining continuous feedback. Depending on the organization's trust in the evaluation criteria, it could be used either as a quick feature screening tool, identifying potential candidate features requiring further investigation, or it could be used as the standard tool that feature implementations are bench-marked against. Further investigation methods could entail consulting feature experts with the specific domain knowledge for additional insight, or conducting surveys on parts of the user base to obtain user opinions. A/B-testing as a way of evaluating feature variants is a good tool, and it can become a central part of the BML-loop by bringing forth an easy way of posing relative hypotheses, utilizing the evaluation criteria we developed. The way A/B-tests can be used in practice can be tailored according to context, feature improvements could be tested against the old feature, different variants could be tested simultaneously, to evaluate the continuous improvements and additions made to an application.

Chapter 9
Conclusion

Satisfying organizational knowledge needs for effective feature development is an important aspect to consider, especially for customer-oriented software companies. Knowing how the end-user interacts with and think of an application can provide a basis for decision making and planning activities related to further development. To this end, customer data needs to be collected and analyzed, results must then subsequently be interpreted, in order to close the feature feedback loop. Using post-deployment data as an information channel constitutes a quick way of learning how well the application performs in production. Collecting performance-, error-, or usage related data enables for rapid feedback, making quick pivoting of development efforts possible. The informed feature development process consequently benefits from ways of evaluating features according to the specific organization's business goals. For this purpose, consciously defining evaluation criteria capturing long term value drivers for an organization that still are measurable in the short term becomes a crucial activity.

For mobile applications not directly monetized in the application itself, alternative value drivers must be identified. Providing an exemplary user experience and features that are of high value to the users can constitute one such organizational business value target. In light of this, being able to capture user happiness and the knowledge of what features the users value by processing collected usage data becomes very attractive. One of the original motivations for the thesis work was to create such feature evaluation criteria. Capturing end-user satisfaction with a feature by utilizing collected usage data can be attempted, but is not without complications in practice. Noise and overall stochasticity of collected data, produced by a wide array of underlying causes, pose real challenges in this regard. While it could be possible to construct such evaluation criteria, a surefire way used in industry to measure user satisfaction is to conduct surveys, asking users for their opinions directly.

A relative ranking of four features of a mobile application was obtained in terms of user opinions collected using surveys. Using only post-deployment data, the same ranking could be found. This indicates that by only processing post-deployment data, users' opinions about the feature could be captured to some extent. To further investigate and validate this finding for our context, further surveys should be conducted in order to expand the evaluation data basis beyond what was accumulated during this thesis. Surveys are on their own also a potential way of evaluating features for this category of business targets. However, the specific target audience of the user base should be considered; can the survey effort be directed intelligently to specific parts of the user base?

Informed feature development can also be administered by experimenting with new ideas, deciding whether to pivot or persevere based on what can be observed in the collected post-deployment data. Introducing a novel software development strategy to the department in the form of the A/B-test methodology, new things about a mobile application and its user base was learned. By conducting a proof-of-concept A/B-test, including A and B variant implementations, hypothesis formulation and, live deployment to production and, subsequent analysis of post-deployment data, this methodology was proven effective in its capabilities to test experiment with new ideas, empirically drawing conclusions based on data. The utilization of A/B-tests helps establish a causal relationship about implementations and their effect, making the data that the decision making is to be made upon more reliable. Conducting A/B-test in the smaller development team setting in a large scale way can possibly pose challenges on its own since development resources are limited and need to be directed efficiently. The best way forward to leverage this strategy into the development process should be investigated further or be determined on a case-to-case basis.

References

[1] ISO/IEC JTC 1/SC 7. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement of quality in use. Standard ISO/IEC FDIS 25022:2016, International Organization for Standardization, 2016.

[2] Alan Agresti. *Categorical data analysis*, volume 482 of *A Wiley-Interscience publication*. John Wiley & Sons, New York [u.a.], 2 edition, 2003.

[3] Airbnb. iOS Lottie Docs. `http://airbnb.io/lottie/#/ios`. [Online; accessed 24-April-2020].

[4] I Elaine Allen and Christopher A Seaman. Likert scales and data analyses. *Quality progress*, 40(7):64–65, 2007.

[5] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development, 2001.

[6] Nigel Bevan, Jim Carter, Jonathan Earthy, Thomas Geis, and Susan Harker. New iso standards for usability, usability reports and usability measures. In *HCI 2016: Human-Computer Interaction. Theory, Design, Development and Practice*, volume 9731, pages 268–278, 07 2016.

[7] George E. P. Box, J. Stuart Hunter, and William Gordon Hunter. *Statistics for experimenters : design, innovation and discovery*. Wiley series in probability and statistics. Wiley-Interscience, 2005.

[8] Erik Brynjolfsson, Lorin M. Hitt, and Heekyung H. Kim. Strength in numbers: How does data-driven decisionmaking affect firm performance? *SSRN eLibrary*, 2011.

[9] Pavel Dmitriev, Somit Gupta, Dong Woo Kim, and Garnet Vaz. A dirty dozen: twelve common metric interpretation pitfalls in online controlled experiments. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1427–1436, 2017.

[10] Weitao Duan, Qian Wang, Rogier Verhulst, and Ya Xu. Scalable online survey framework: from sampling to analysis. *arXiv preprint arXiv:1906.10082*, 2019.

[11] Paul M. Duvall, Steve Matyas, and Andrew Glover. *Continuous integration : improving software quality and reducing risk.* Addison-Wesley signature series. Addison-Wesley, 2007.

[12] Firebase Help. Automatically collected events. `https://support.google.com/firebase/answer/6317485`, 2020. [Online; accessed 13-March-2020].

[13] Firebase Help. BigQuery Export schema. `https://support.google.com/firebase/answer/7029846?hl=en`, 2020. [Online; accessed 13-March-2020].

[14] Firebase Help. Event and parameter details (Google Analytics for Firebase). `https://support.google.com/firebase/answer/7061705?hl=en&ref_topic=7029512`, 2020. [Online; accessed 13-March-2020].

[15] Google Developers. Firebase A/B Testing. `https://firebase.google.com/docs/ab-testing`, 2020. [Online; accessed 13-March-2020].

[16] Google Developers. Firebase Remote Config. `https://firebase.google.com/docs/remote-config`, 2020. [Online; accessed 13-March-2020].

[17] Google Developers. Google Analytics. `https://firebase.google.com/docs/analytics/`, 2020. [Online; accessed 13-March-2020].

[18] Marijn Janssen, Haiko van der Voort, and Agung Wahyudi. Factors influencing big data decision-making quality. *Journal of Business Research*, 70:338 – 345, 2017.

[19] N.L. Johnson, A.W. Kemp, and S. Kotz. *Univariate Discrete Distributions*. Wiley Series in Probability and Statistics. Wiley, 2005.

[20] Ron Kohavi, Alex Deng, Brian Frasca, Roger Longbotham, Toby Walker, and Ya Xu. Trustworthy online controlled experiments: Five puzzling outcomes explained. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 786–794, 2012.

[21] Ron Kohavi and Roger Longbotham. *Online Controlled Experiments and A/B Testing*, pages 922–929. Springer US, Boston, MA, 2017.

[22] Christoph Matthies and Guenter Hesse. Towards using data to inform decisions in agile software development: Views of available data. *Proceedings of the 14th International Conference on Software Technologies*, 2019.

[23] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. Dremel: Interactive analysis of web-scale datasets. *Proc. VLDB Endow.*, 3(1–2):330–339, September 2010.

[24] H.H. Olsson, H. Alahyari, and J. Bosch. Climbing the "stairway to heaven" – a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. *2012 38th Euromicro Conference on Software Engineering and Advanced Applications, Software Engineering and Advanced Applications (SEAA), 2012*

38th EUROMICRO Conference on, Software Engineering and Advanced Applications, Euromicro Conference on, pages 392 – 399, 2012.

[25] H.H. Olsson and J. Bosch. From opinions to data-driven software r&d: A multi-case study on how to close the 'open loop' problem. *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on, Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, pages 9 – 16, 2014.

[26] Eric Ries. *The lean startup: How constant innovation creates radically successful businesses*. Portfolio Penguin, London; New York, 2011.

[27] Kerry Rodden, Hilary Hutchinson, and Xin Fu. Measuring the user experience on a large scale: User-centered metrics for web applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 2395–2398, New York, NY, USA, 2010. Association for Computing Machinery.

[28] Walker W. Royce. Managing the development of large software systems: concepts and techniques. *Proc. IEEE WESTCON, Los Angeles*, pages 1–9, August 1970. Reprinted in *Proceedings* of the Ninth International Conference on Software Engineering, March 1987, pp. 328–338.

[29] T. Sauvola, L.E. Lwakatare, T. Karvonen, P. Kuvaja, H.H. Olsson, J. Bosch, and M. Oivo. Towards customer-centric software development: A multiple-case study. *2015 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2015.

[30] Martha Sinclair, Joanne Otoole, Manori Malawaraarachchi, and Karin Leder. Comparison of response rates and cost-effectiveness for a community-based survey: Postal, internet and telephone modes with generic or personalised recruitment approaches. *BMC medical research methodology*, 12:132, 08 2012.

[31] S. Suonsyrja, O. Sievi-Korte, K. Systa, T. Kilamo, T. Mikkonen, T. Bures, and L. Angelis. Objectives and challenges of the utilization of user-interaction data in software development. *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2018.

[32] Sampo Suonsyrja, Laura Hokkanen, Henri Terho, Kari Systa, and Tommi Mikkonen. Post-deployment data: A recipe for satisfying knowledge needs in software development?. *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2016 Joint Conference of the International Workshop on, IWSM-MENSURA*, pages 139 – 147, 2016.

[33] Richard Berntsson Svensson, Robert Feldt, and Richard Torkar. The unfulfilled potential of data-driven decision making in agile software development. In Philippe Kruchten, Steven Fraser, and François Coallier, editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 69–85, Cham, 2019. Springer International Publishing.

[34] James Tamplin. Firebase is joining google! https://techcrunch.com/2014/10/21/google-acquires-firebase-to-help-developers-build-better-realtime-apps/, 10 2014. [Online; accessed 13-March-2020].

[35] Vera Toepoel. *Doing Surveys Online*. SAGE Publications, Inc., 55 City Road, London, 1 edition, Jun 2016.

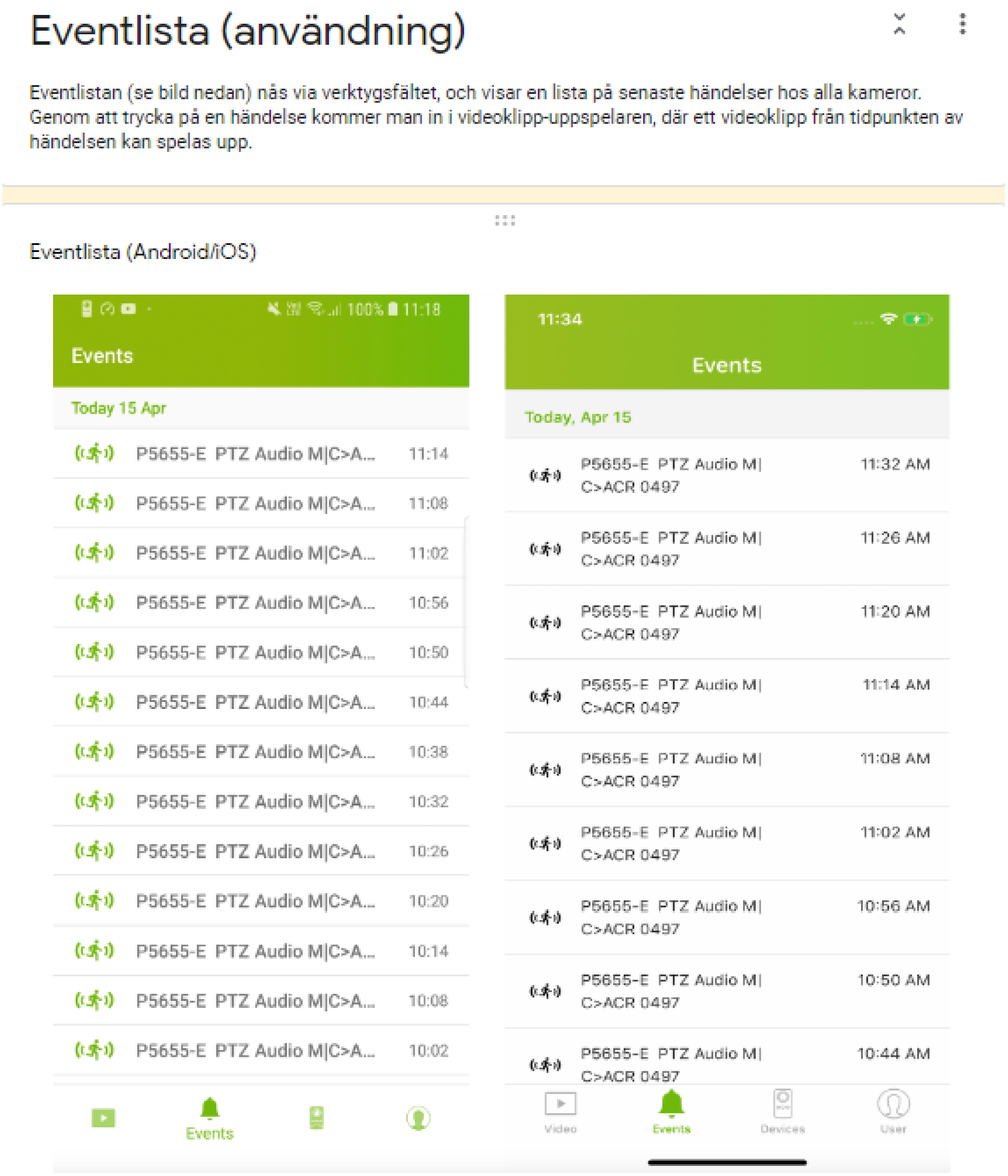
[36] B. L. Welch. The generalization of 'Student's' problem when several different population variances are involved. *Biometrika*, 34(1-2):28–35, 01 1947.

[37] Kevin B. Wright. Researching Internet-Based Populations: Advantages and Disadvantages of Online Survey Research, Online Questionnaire Authoring Software Packages, and Web Survey Services. *Journal of Computer-Mediated Communication*, 10(3), 07 2017. JCMC1034.

Appendices

Appendix A

Online survey - Feature descriptions



Eventlista (användning)

Eventlistan (se bild nedan) nås via verktygsfältet, och visar en lista på senaste händelser hos alla kameror. Genom att trycka på en händelse kommer man in i videoklipp-uppspelaren, där ett videoklipp från tidpunkten av händelsen kan spelas upp.

Eventlista (Android/iOS)

100% 11:18

Events

Today 15 Apr

P5655-E PTZ Audio M|C>A...	11:14
P5655-E PTZ Audio M|C>A...	11:08
P5655-E PTZ Audio M|C>A...	11:02
P5655-E PTZ Audio M|C>A...	10:56
P5655-E PTZ Audio M|C>A...	10:50
P5655-E PTZ Audio M|C>A...	10:44
P5655-E PTZ Audio M|C>A...	10:38
P5655-E PTZ Audio M|C>A...	10:32
P5655-E PTZ Audio M|C>A...	10:26
P5655-E PTZ Audio M|C>A...	10:20
P5655-E PTZ Audio M|C>A...	10:14
P5655-E PTZ Audio M|C>A...	10:08
P5655-E PTZ Audio M|C>A...	10:02

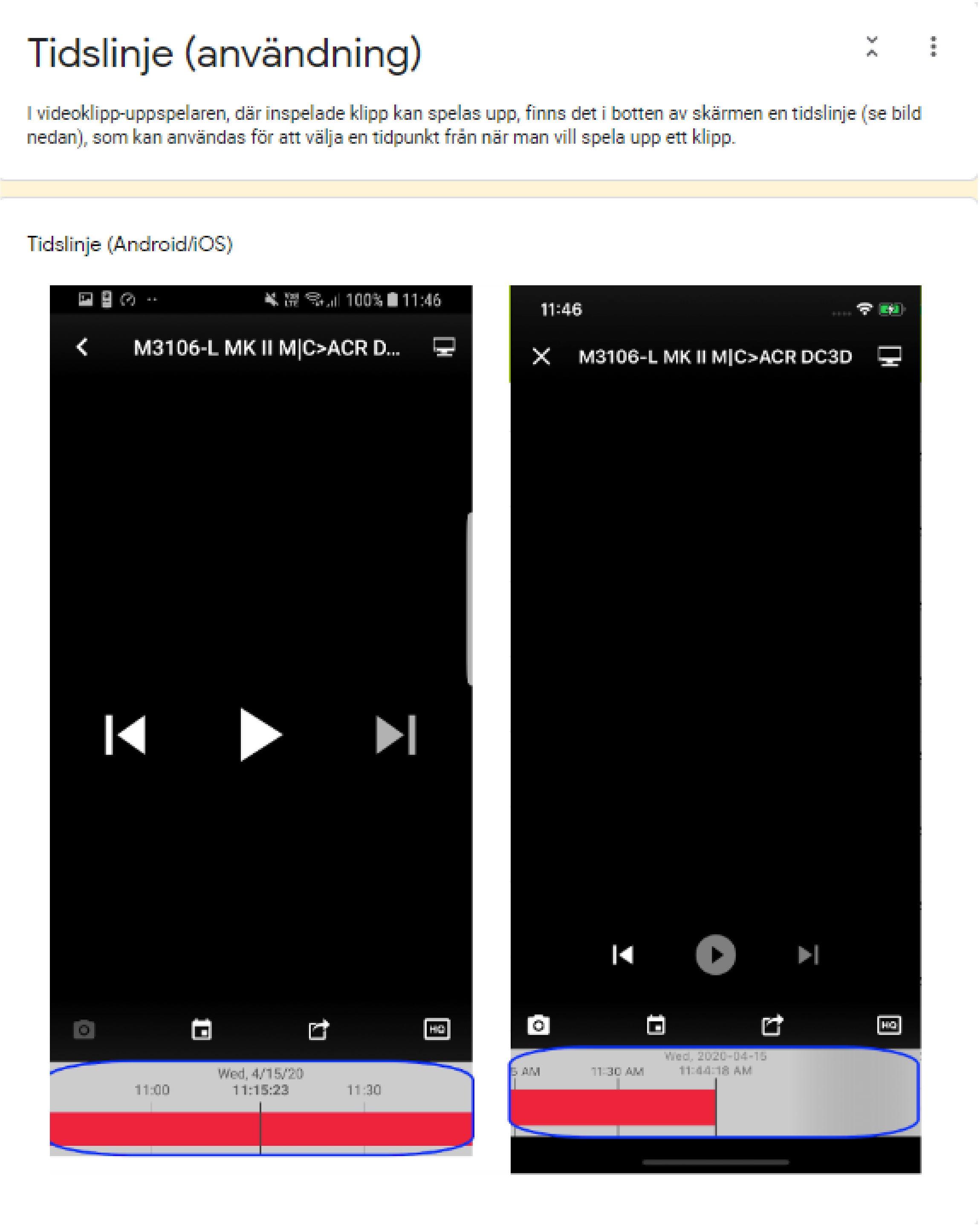
11:34

Events

Today, Apr 15

P5655-E PTZ Audio M| C>ACR 0497	11:32 AM
P5655-E PTZ Audio M| C>ACR 0497	11:26 AM
P5655-E PTZ Audio M| C>ACR 0497	11:20 AM
P5655-E PTZ Audio M| C>ACR 0497	11:14 AM
P5655-E PTZ Audio M| C>ACR 0497	11:08 AM
P5655-E PTZ Audio M| C>ACR 0497	11:02 AM
P5655-E PTZ Audio M| C>ACR 0497	10:56 AM
P5655-E PTZ Audio M| C>ACR 0497	10:50 AM
P5655-E PTZ Audio M| C>ACR 0497	10:44 AM

Figure A.1: Screenshots and description of the event list feature of the AXIS Companion application.

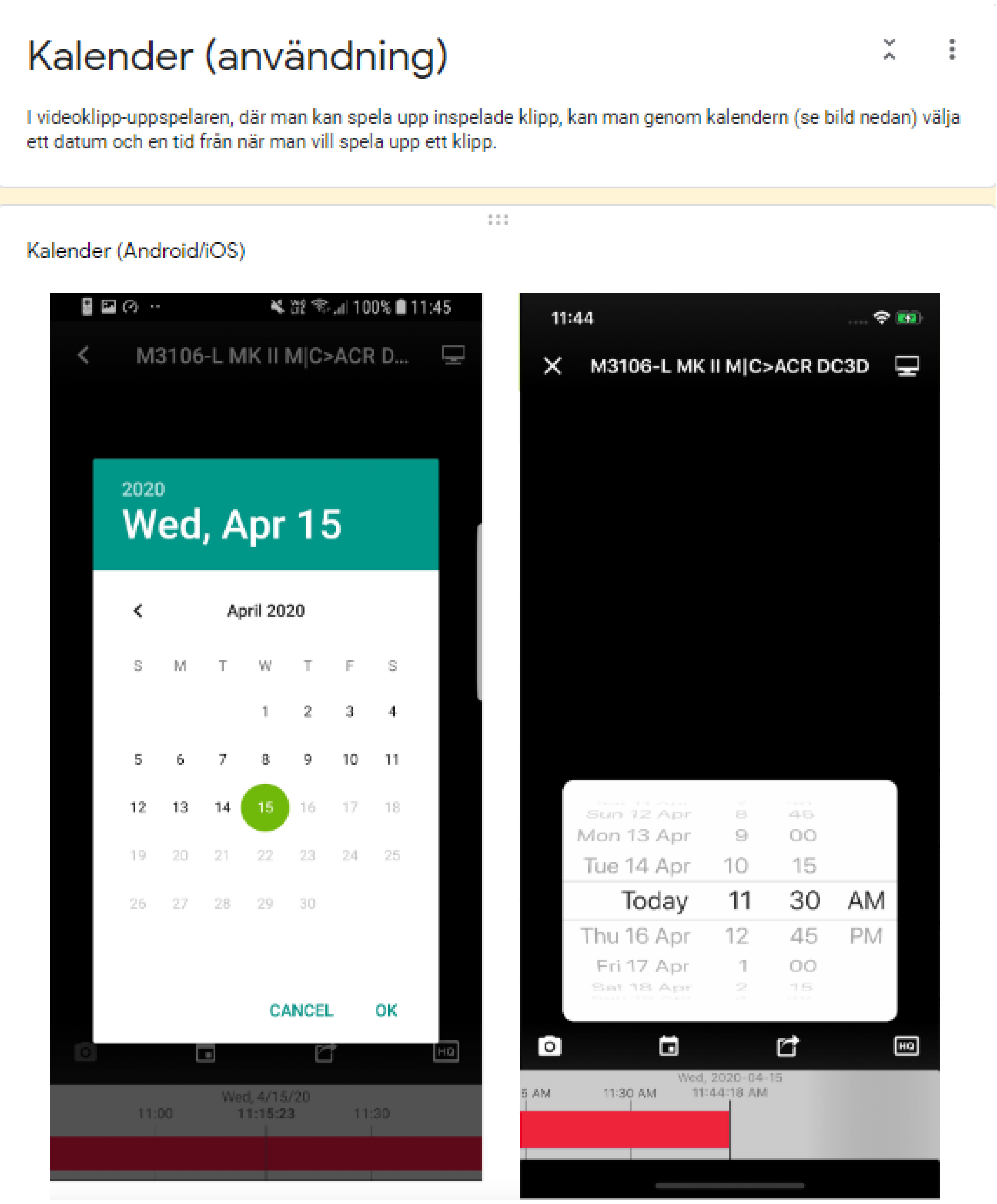


Tidslinje (användning)

I videoklipp-uppspelaren, där inspelade klipp kan spelas upp, finns det i botten av skärmen en tidslinje (se bild nedan), som kan användas för att välja en tidpunkt från när man vill spela upp ett klipp.

Tidslinje (Android/iOS)

Figure A.2: Screenshots and description of the timeline feature of the AXIS Companion application.

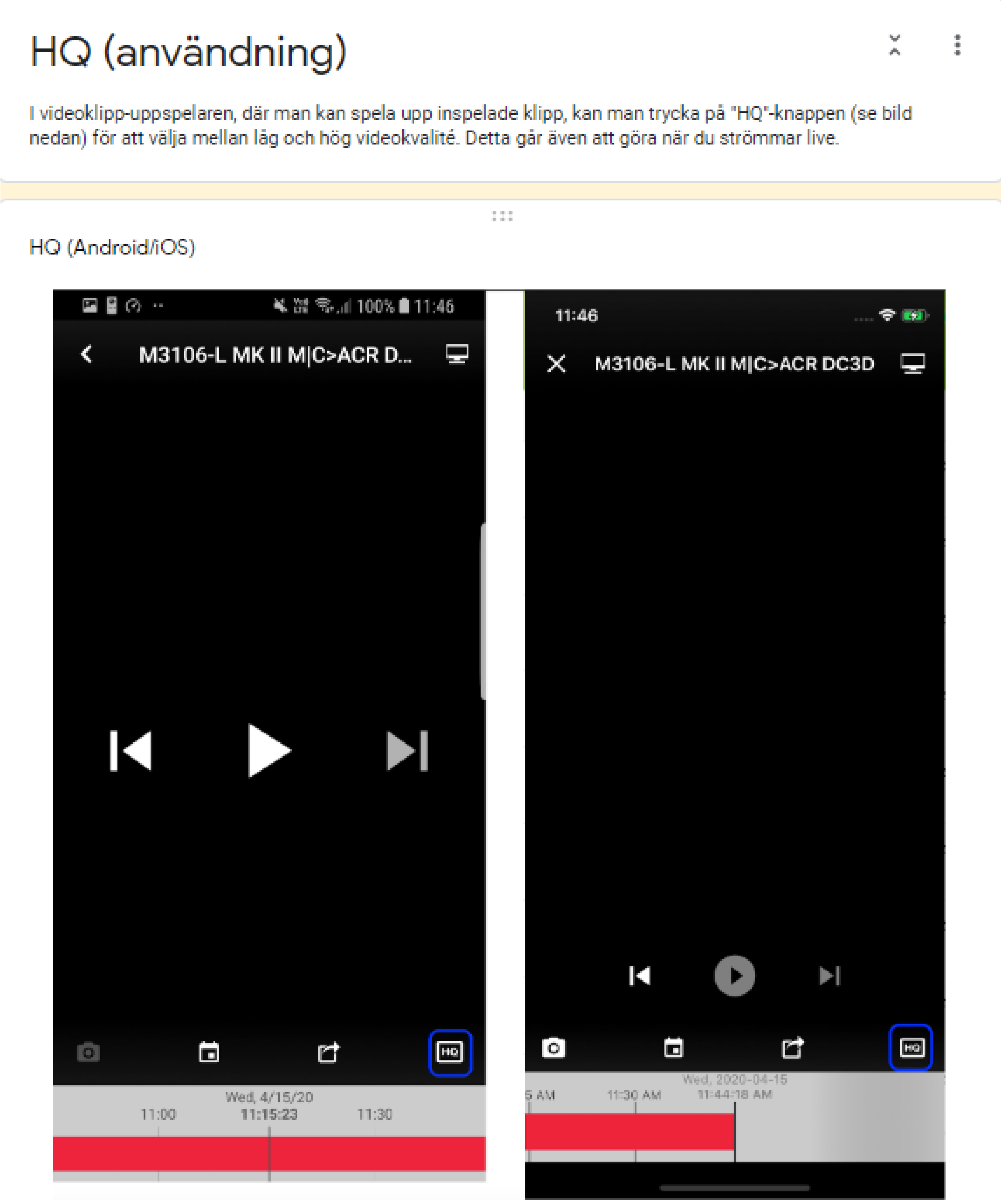


Kalender (användning)

I videoklipp-uppspelaren, där man kan spela upp inspelade klipp, kan man genom kalendern (se bild nedan) välja ett datum och en tid från när man vill spela upp ett klipp.

Kalender (Android/iOS)

Figure A.3: Screenshots and description of the playback calendar feature of the AXIS Companion application.



HQ (användning)

I videoklipp-uppspelaren, där man kan spela upp inspelade klipp, kan man trycka på "HQ"-knappen (se bild nedan) för att välja mellan låg och hög videokvalité. Detta går även att göra när du strömmar live.

HQ (Android/iOS)

Figure A.4: Screenshots and description of the video quality selection (HQ) feature of the AXIS Companion application.

Appendix B

Online survey - Questions and answers

Undersökning av Axis Companions mobilapp-användning

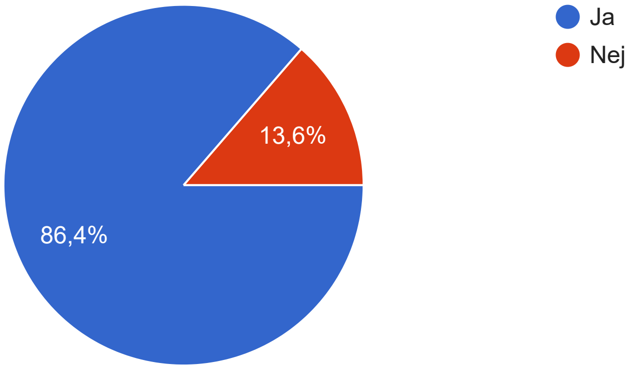
22 svar

Publicera analyser

Generella Frågor

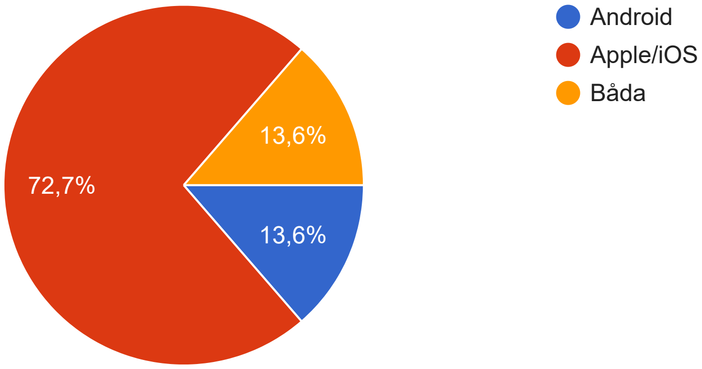
Är du anställd på Axis?

22 svar



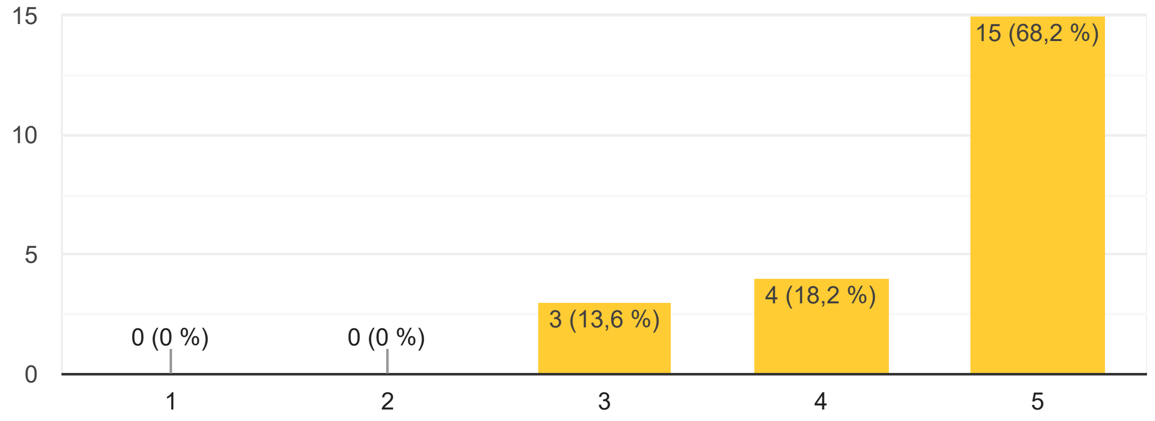
Vilken plattform använder du Companion på?

22 svar



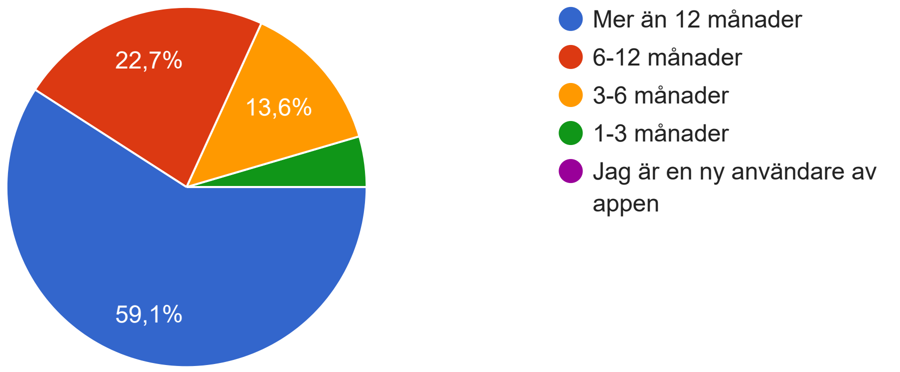
Hur van vid att använda mobilappar skulle du säga att du är?

22 svar



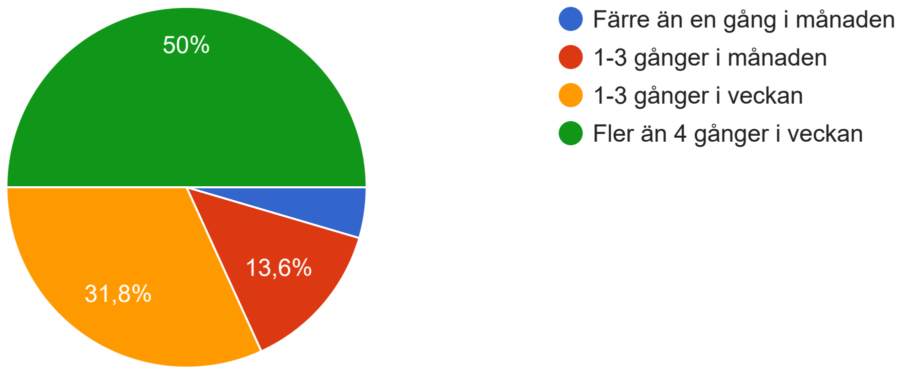
Hur länge har du använt mobilappen för Companion?

22 svar



Hur ofta använder du mobilappen för Companion?

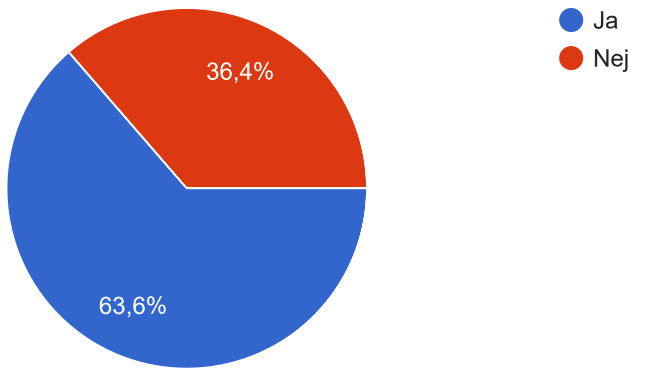
22 svar



Eventlista (användning)

Har du använt eventlistan?

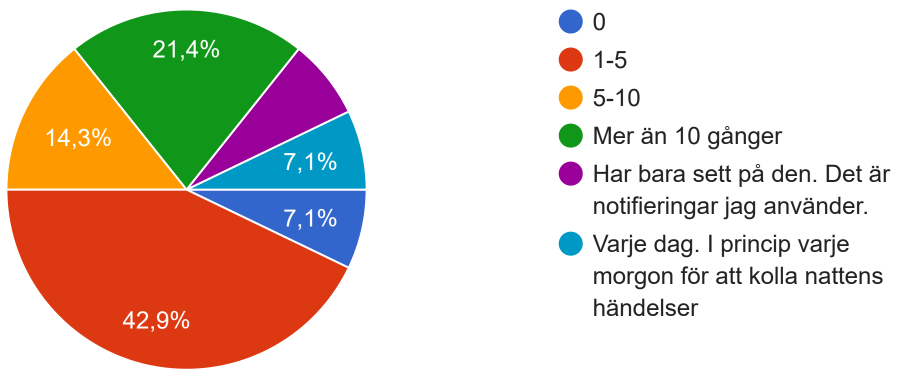
22 svar



Eventlista (frekvens)

Ungefär hur ofta använder du eventlistan per månad?

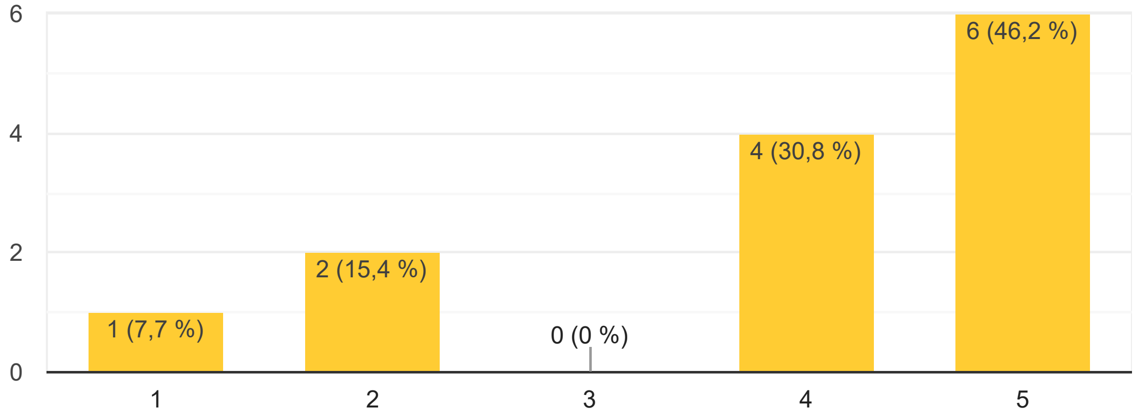
14 svar



Eventlista (utvärdering)

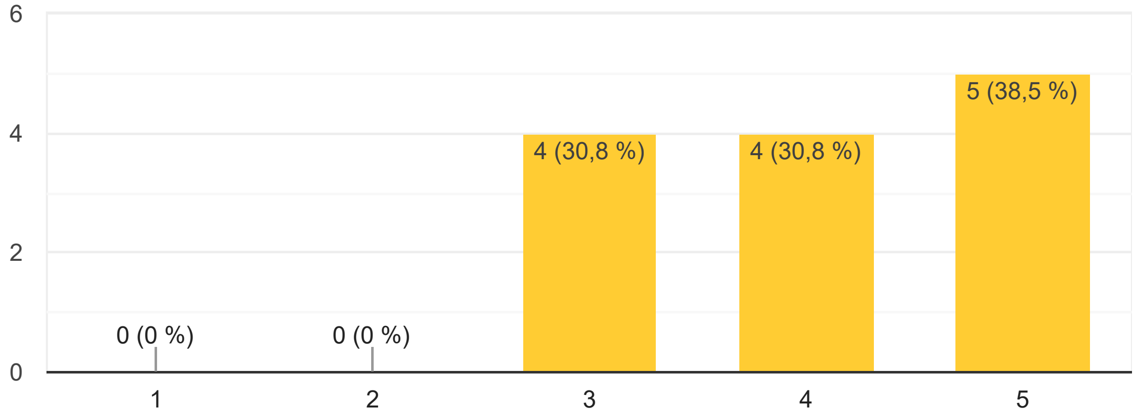
Vilket värde har eventlistan för din användning av Companion? (omformulera)

13 svar



Vad är din generella upplevelse av eventlistan?

13 svar



Övriga synpunkter på eventlistan? (valfritt)

5 svar

Kanske är en bug, men när jag trycker på ett event börjar videon spela ca 15sek innan eventet triggas.

Förenklar att hitta i videon, men borde kunna användas för fler typer av event än VMD

Ett smidigare och snabbare sätt att hitta till händelser/inspelningar än att bläddra bland recordings.

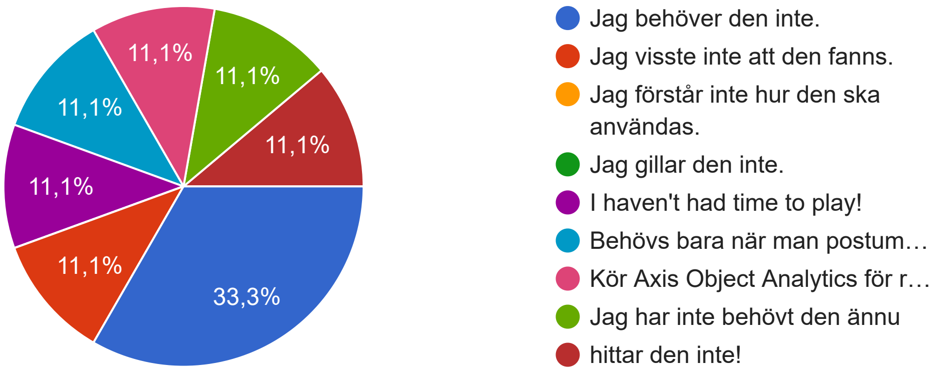
Man skulle kunna sortera den efter kamera, inte bara kronologisk

Använder det framförallt när jag är på semester.

Eventlista (ej använd)

Varför använder du inte eventlistan?

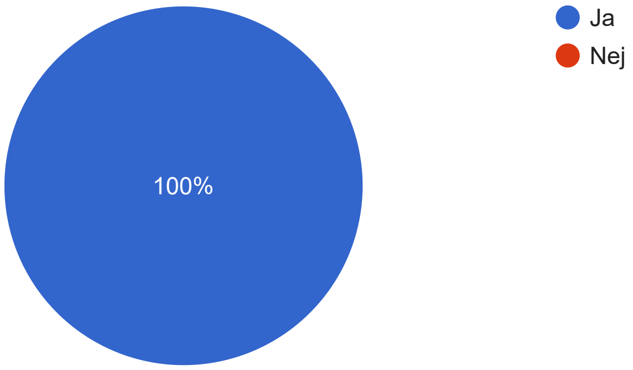
9 svar



Tidslinje (användning)

Har du använt tidslinjen?

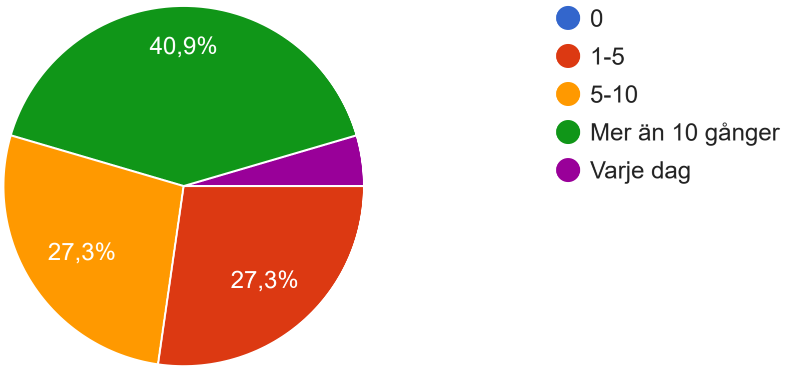
22 svar



Tidslinje (frekvens)

Ungefär hur ofta använder du tidslinjen per månad?

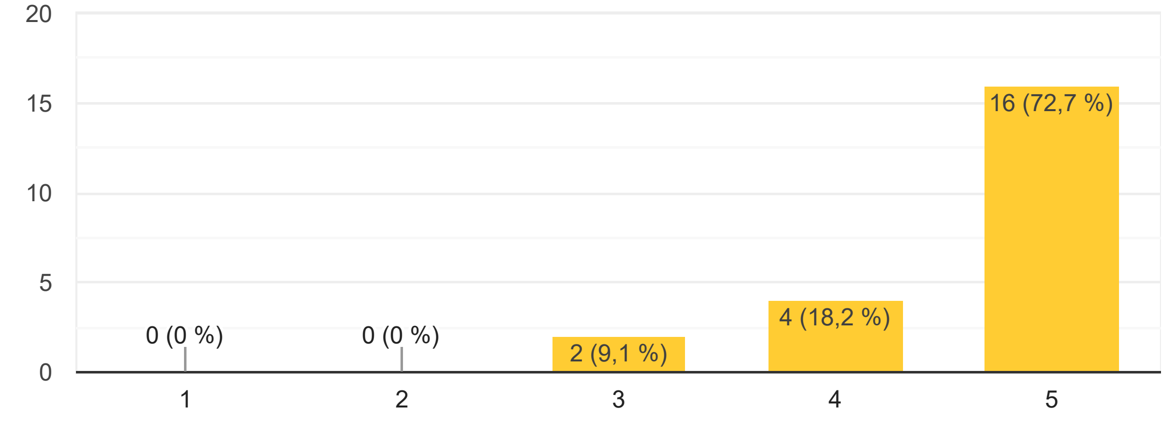
22 svar



Tidslinje (utvärdering)

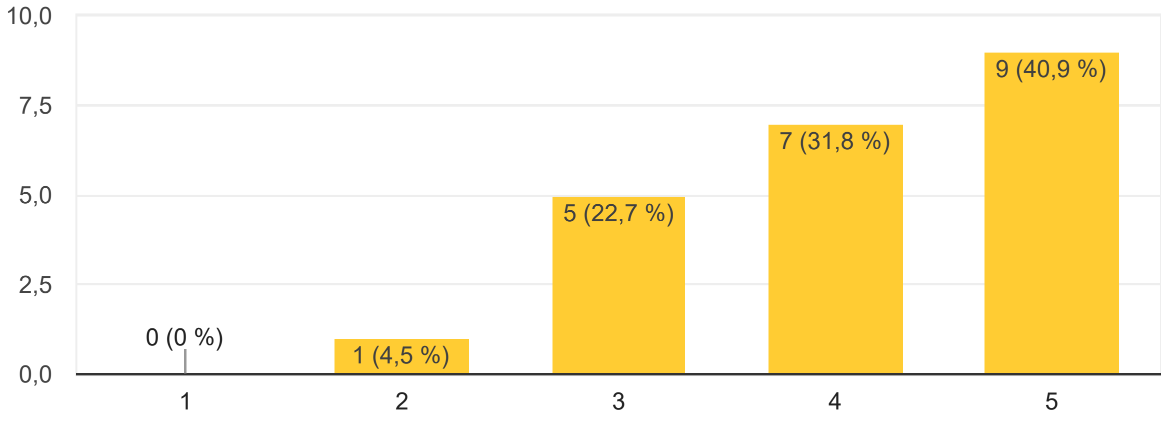
Vilket värde har tidslinjen för dig?

22 svar



Vad är din generella upplevelse av tidslinjen?

22 svar



Övriga synpunkter på tidslinjen? (valfritt)

8 svar

Kanske en bug, men jag tycker att när jag väljer en tidspunkt att spela så hoppar videon bakåt i tiden allt från 1-20sek och börjar spela därifrån.

Jag har vid upprepade tillfällen haft problem med att appen inte går till den tid som jag klickar på i tidslinjen. Buggen är rapporterad, men problemet finns fortfarande kvar.

It often seems to bug out when scrubbing between multiple recordings

Enkel att använda. Lite långsamt att accessa video

Blir lite buggig att starta klipp ibland då jag spelar in video via ett annat event än det VMD event som companion skapar på din kamera.

Skulle vilja ha fler färger beroende på vad som triggat event (Continouos, Motion triggered VMD, andra triggade event från ACAPS (AOA), etc

Superanvändbart

Tidslinje (ej använd)

Varför använder du inte tidslinjen?

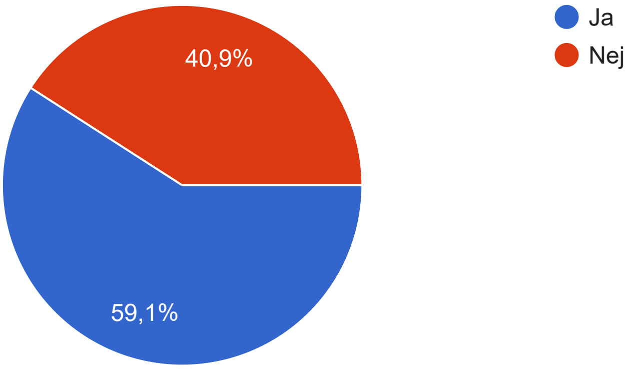
0 svar

Det finns ännu inga svar på den här frågan.

Kalender (användning)

Har du använt kalendern?

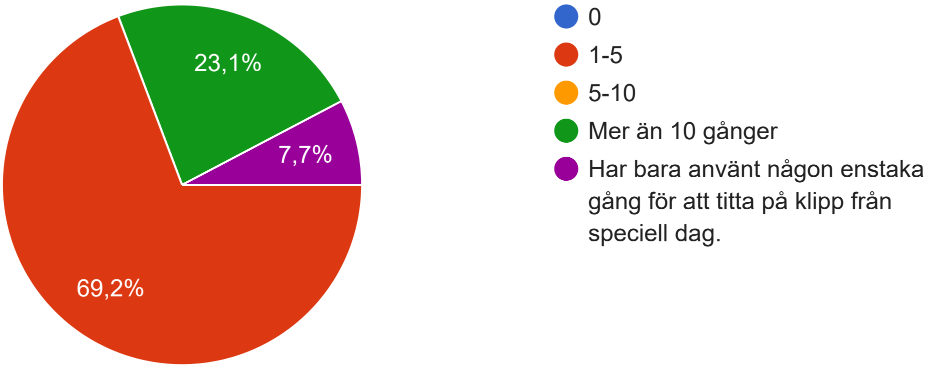
22 svar



Kalender (frekvens)

Ungefär hur ofta använder du kalendern per månad?

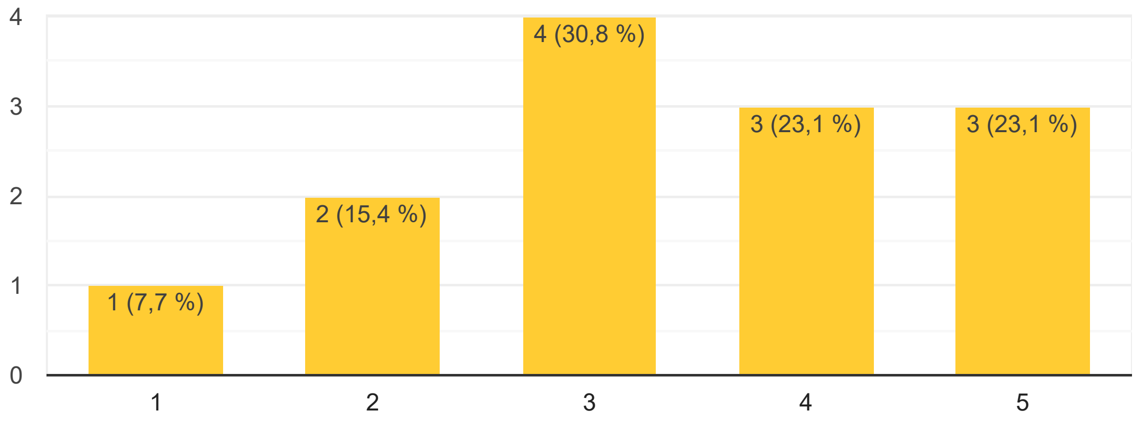
13 svar



Kalender (utvärdering)

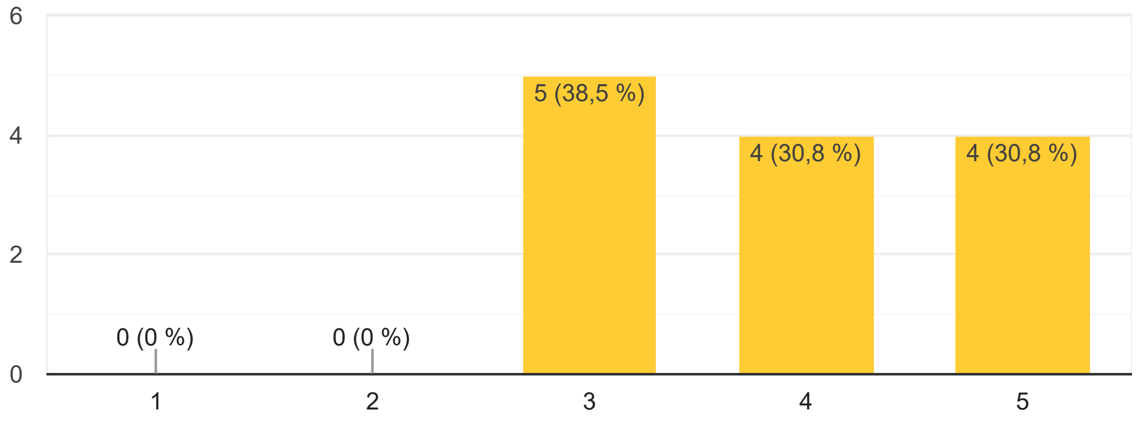
Vilket värde har kalendern för dig?

13 svar



Vad är din generella upplevelse av kalendern?

13 svar



Övriga synpunkter på kalendern? (valfritt)

2 svar

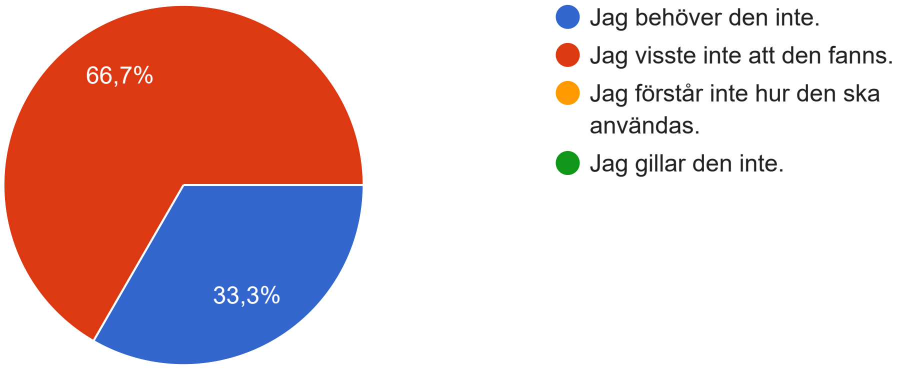
Inga generella förbättringsförslag

Egentligen är det en värdefull funktionalitet när det väl behövs, jag har bara inte haft behovet så ofta.

Kalender (ej använd)

Varför använder du inte kalendern?

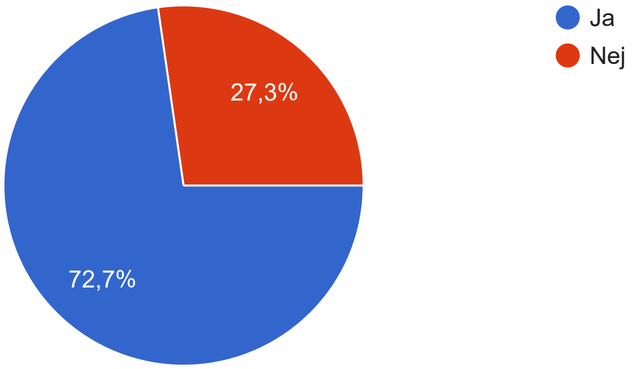
9 svar



HQ (användning)

Har du använt HQ?

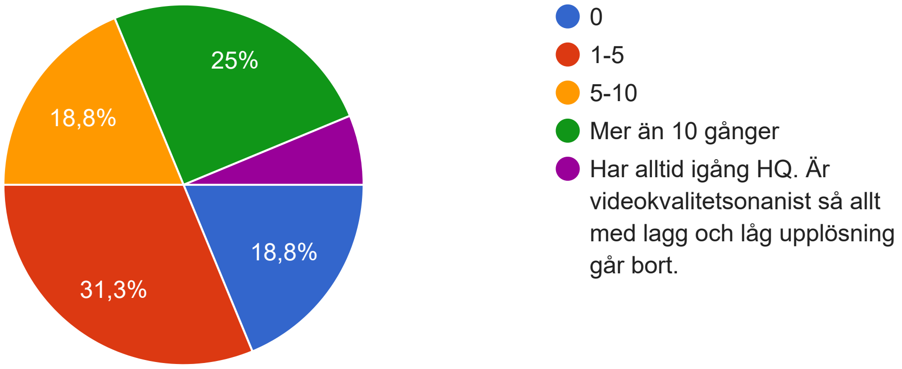
22 svar



HQ (frekvens)

Ungefär hur ofta använder du HQ per månad?

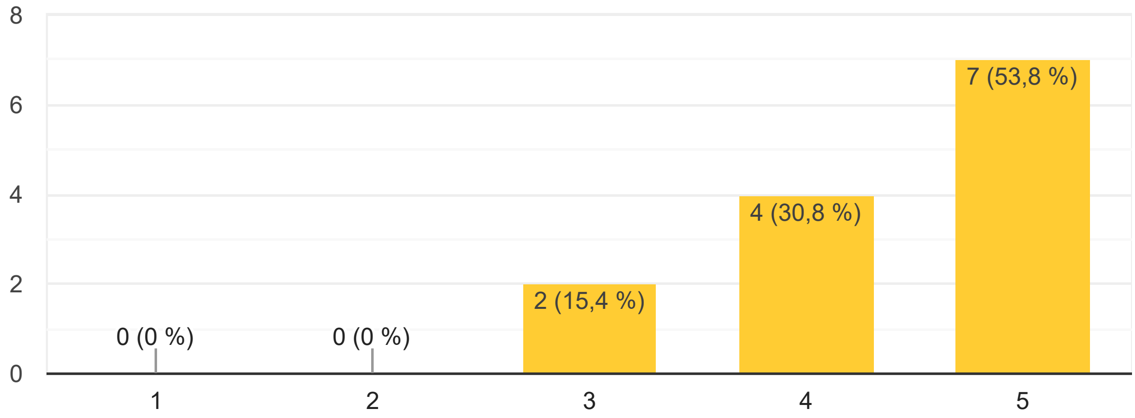
16 svar



HQ (utvärdering)

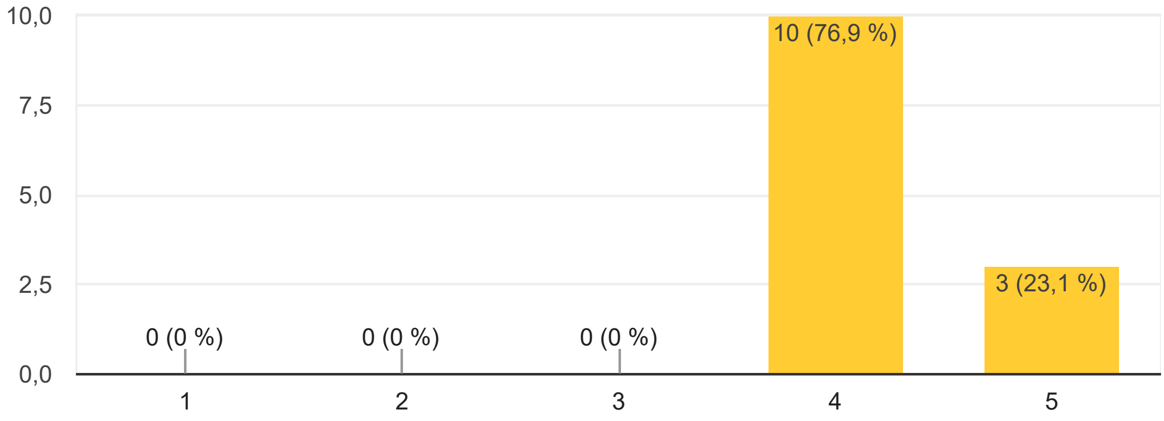
Vilket värde har HQ för dig?

13 svar



Vad är din generella upplevelse av HQ?

13 svar



Övriga synpunkter på HQ? (valfritt)

5 svar

tar ibland lång tid att "ladda om" videon från låg till hög kvalite (men det kan man inte lasta appen för)

Jag tror funktionen kan vara svår att hitta för användare.

Använder sällan low quality stream

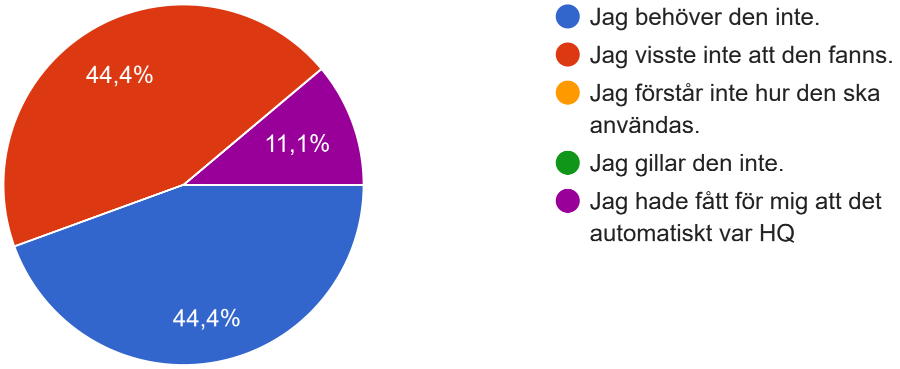
Har problem att leverera baserat på uppkopplingshastighet. Dvs ska du kolla en 4k inspelning via mobilnät och du har överbelastat nät eller dålig täckning så laggar filmerna rätt friskt. Borde bli bättre på att buffra innan filmerna spelas upp.

Lätt att glömma att man har HQ disablad. Kanske ha olika färger för icke HQ och HQ?

HQ (ej använd)

Varför använder du inte HQ?

9 svar



Appendix C

Python script for Welch’s t-test and the χ^2 test

```
import numpy as np
from math import sqrt
import scipy as sp
import scipy.stats
import itertools

###############################################
# Welch’s t−test for cumulative engagement

def welchParams(la, lb, na, nb):
    va, vb = na − 1, nb − 1
    t = (la − lb) / sqrt(la / na + lb / nb)
    v = (la / na + lb / nb)**2 / \
        (la**2 / (na**2 * va) + lb**2 / (nb**2 * vb))
    return t, v

# 14 days
lambda_a = 13.86
lambda_b = 14.43
lambda_c = 8.0
n = 14

# 28 days
# lambda_a = 13.5
# lambda_b = 14.0
# lambda_c = 10.42
# n = 28

for (label, lambda_1, lambda_2) in [(’A/B’, lambda_a, lambda_b),
                                     (’A/C’, lambda_a, lambda_c),
                                     (’B/C’, lambda_b, lambda_c)]:
```

```
    t, v = welchParams(lambda_1, lambda_2, n, n)
    p = 2 * sp.stats.t.cdf(-abs(t), v)

    print()
    print('Welch\'s t-test for:', label)
    #print('t:', t, 'v:', v)
    print('p-value:', p)
    print(str('-'*10))

##############################################


##############################################
# Chi2 test for cumulative adoption

# 14 days
a = [146, 597]
b = [143, 531]
c = [88, 614]


# 28 days
a = [237, 794]
b = [255, 719]
c = [158, 810]

ab = [a, b]
ac = [a, c]
bc = [b, c]

for (label, m) in [('A/B', ab), ('A/C', ac), ('B/C', bc)]:
    _, p, _, _ = sp.stats.chi2_contingency(m)

    print()
    print('Chi2 test for:', label)
    print('p-value:', p)
    print(str('-'*10))

##############################################
```

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-06-11

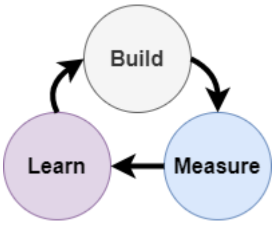
EXAMENSARBETE Data-Driven Feature Development
Using user-interaction data for feature evaluation
STUDENTER Emil Ahlberg, Oskar Widmark
HANDLEDARE Ulf Asklund (LTH), Markus Andersson (AXIS Communications)
EXAMINATOR Martin Höst (LTH)

Datadriven featureutveckling

POPULÄRVETENSKAPLIG SAMMANFATTNING **Emil Ahlberg, Oskar Widmark**

Mobilappar bör utvecklas för att möta användarnas behov och önskemål. Insamlad användningsdata kan utnyttjas för att lära sig mer om sin användare, vilket kan vägas in i framtida utvecklingsprioriteringar. Arbetet utforskar A/B-test som metod samt framtagande av utvärderingskriterier för att indikera användarvärde i detta syfte.

Att använda insamlad användningsdata för att förbättra en app är en mycket tilltalande strategi i en värld där fler och fler processer blir datadrivna. A/B-test är ett koncept där olika varianter av en feature slumpvis tilldelas olika appanvändare vilket utgör ett bra sätt att experimentera med nya idéer. Genom att testa olika varianter på sin användarbas kan en organisations hypoteser kring olika implementationsförslag utvärderas ordentligt. Ett enkelt experiment skulle kunna undersöka antal klick på en knapp som användare generarar då knappen för olika användargruppen ges olika färg. Genom att titta på ett antal generella utvärderingskriterier, till exempel hur många distinkta användare som faktiskt använder en knapp eller hur ofta en knapp används, kan en knapp-feature bedömas utifrån insamlad data.



Vi genomför ett A/B-test för en mobilapp för att undersöka effekten av att informera användare om nya features på två olika sätt jämfört med att inte gå ut med någon information alls.

Resultatet för A/B-testet visar tydligt att fler användare provar en feature då information ges. Vilken variant av informationsdelning som är att föredra kunde inte med urskiljas på ett statistiskt signifikant sätt.

För appar vilka främst har nöjda användare som mål är det värdefullt om målets uppfyllandegrad kan fångas i liknande utvärderingskriterier. Features skulle isåfall kunna utvärderas snabbt och resurseffektivt. I vårt arbete rankar vi användarens sammanvägda behov och upplevelse av en feature med enbart användningsdata. Vi samlar även in användares åsikter kring utvalda appfeatures för att svara på om kundnöjdhet är en egenskap som kan bedömas utifrån insamlad data.

Utifrån inkomna svar på enkäter gällande användares åsikter kring olika features erhölls samma featurerankning som vi genererat utifrån att bara titta på användningsdata. Detta indikerar att kundnöjdhet kan uppskattas genom att titta på användningsdata, men större utvärderingsunderlag bör samlas in för att ytterligare styrka denna slutsats.

Analys av användningsdata parat med A/B-test är ett bra sätt att lära sig mer om en app och dess användare. Om kundnöjdhet verkligen fångas av vår modell kan apputveckling i framtiden göras mer datadrivet.