Strategies, Benefits and Challenges of App Store-inspired Requirements Elicitation

Alessio Ferrari
ISTI-CNR
alessio.ferrari@isti.cnr.it

Paola Spoletini
Kennesaw State University, USA
pspoleti@kennesaw.edu

***Abstract*—App store-inspired elicitation is the practice of exploring competitors' apps, to get inspiration for requirements. This activity is common among developers, but little insight is available on its practical use, advantages and possible issues. This paper aims to empirically analyse this technique in a realistic scenario, in which it is used to extend the requirements of a product that were initially captured by means of more traditional requirements elicitation interviews. Considering this scenario, we conduct an experimental simulation with 58 analysts and collect qualitative data. We perform thematic analysis of the data to identify strategies, benefits, and challenges of app store-inspired elicitation, as well as differences with respect to interviews in the considered elicitation setting. Our results show that: (1) specific guidelines and procedures are required to better conduct app store-inspired elicitation; (2) current search features made available by app stores are not suitable for this practice, and more tool support is required to help analysts in the retrieval and evaluation of competing products; (3) while interviews focus on the *why* dimension of requirements engineering (i.e., goals), app store-inspired elicitation focuses on *how* (i.e., solutions), offering indications for implementation and improved usability. Our study provides a framework for researchers to address existing challenges and suggests possible benefits to fostering app store-inspired elicitation among practitioners.**

I. Introduction

Requirements can be elicited from stakeholders through several techniques, including interviews, focus groups, workshops, and questionnaires [1]. In the last decade, together with the increasing growth of the market of mobile applications (*apps*), app stores and user reviews offered a new means to gather requirements directly from users [2], [3]. Automated solutions have been developed to classify reviews [4], mining non-functional requirements [5], trace reviews with other artifacts [6]–[9], and other tasks [3], [10]. Information from app stores is also used by developers in their daily practice, to prioritise features based on user feedback [11], and even to look into competing apps to better understand and exploit the existing market. In particular, the recent survey by Al-Subaihin *et al.* [11], involving 186 developers from 36 countries, shows that *The majority of surveyed developers use it [the app store] to explore apps related to their application domain to gain an understanding of the expected user experience and anticipate features*. In this paper, we refer to this practice with the name *app store-inspired elicitation* (ASE). To support this activity, researchers have recently developed specific tools to extract features from competing products, and enable comparison and feature recommendation [12]–[14]. However, despite the common use of ASE, limited information is available on its use in practice, and in particular on what are the specific strategies adopted by requirements analysts and developers to select inspirational apps. Furthermore, knowing what are the advantages and difficulties of ASE, also with respect to other elicitation techniques, could help to better understand when and how to use it.

In this paper, we empirically study ASE in an experimental simulation, in which we evaluate the natural behaviour of analysts in a contrived setting [15]. Specifically, we recruit 58 analysts, and we set-up a realistic elicitation scenario consisting of two phases. In the first phase, the analysts perform a classical elicitation process with *interview-based elicitation* (IBE), in which they verbally and synchronously communicate with a customer to define an initial set of requirements. In the second phase, they use ASE to extend the requirements following the ideas inspired by similar apps. The analysts report their rationale for choosing inspirational apps as well as reflections on the different process phases. We perform thematic analysis of the qualitative data, and identify strategies, benefits and challenges of ASE, as well as differences with respect to IBE in the specific two-phase elicitation setting.

According to our analysis, different strategies are used for the selection of inspirational apps. These can be driven by well-planned searches based on possibly required features, but also lazily based on what is returned by search engines, according to simple queries concerning the app domain. The main emerging benefits concern the possibility of performing *hands-on* evaluation of competitors' products, thus informing the implementation. Challenges include the difficulty of comparing products in the app store, the uninformative content of app reviews, the risk to mimic other apps, and the absence of a structured process to support ASE. The identified core difference with interviews is that IBE focuses on goals, and ASE focuses on solutions. While the former helps to better interpret needs and foster stakeholders' relationship, the latter supports feasibility assessment, prevention of usability issues, and generalisation of an app for a wider market.

This is the first work that investigates strategies, benefits, and challenges of ASE, a widely common but not highly investigated practice, and provides evidence for them. We are also among the first ones who consider ASE in conjunction with traditional interviews: these are typically treated as in-

dependent silos by software engineering research, while they should be considered as complementary practices. Our findings mainly provide evidence-based knowledge and contribute to *theory* in software engineering. Our results can be useful to researchers, as our list of strategies and challenges provide motivations to develop further knowledge, techniques, and tools around ASE. The identified benefits can also foster an informed adoption of ASE by practitioners, as well as an appropriate combination with more traditional elicitation practices.

II. Related Work

Software engineering supported by app store mining is a widely studied topic. The survey by Martin *et al.* [3] gives a comprehensive overview of the different tasks considered in the literature, while the systematic reviews by Genc-Nayebi and Abran [2], and by Dabrowksi *et al.* [10] provide insights on app review mining. Our work focuses on *requirements elicitation* using the app store, and in the following, we briefly point to relevant papers in this field.

The majority of the studies aim to support requirements elicitation by filtering user feedback, typically in the form of app reviews, e.g., by automatically classifying bug reports and feature requests [4], [16] or by clustering the feedback to support release planning [17], [18], and extraction of non-functional requirements [5]. More recently, given the insufficiency of the app store in fully supporting app development [19], studies have focused on linking app reviews with other requirements-relevant artifacts, such as issue tracking systems [7], bug reports [8], and tweets [9]. These works leverage user feedback as a primary data source.

More closely related works to ours are those leveraging *app descriptions* of competing products, besides reviews, to exploit the app market. For example, Jiang *et al.* [12] automatically recommend new features, by extracting existing functionalities from similar product descriptions and API names. Similarly, Liu *et al.* [13] extract features from competing apps to facilitate app comparison, while Dalpiaz and Parente [14] use reviews for the same goal. Existing tools also support feature extraction from app descriptions. These include SAFE [20], and the solution by Harman *et al.* [21], also extended in later studies to find the correlation between features and ratings [22], [23]. The goal of product comparison and feature recommendation was also addressed in an earlier work by Dumitru *et al.* [24], but considering software descriptions from *Softpedia.com*. In the field of software product line engineering, researchers have addressed the problem of market analysis with similar approaches [25], [26].

Besides studies on the *automation* of ASE, some empirical works exist which aim to investigate software engineering issues in mobile app development by means of surveys with practitioners [27]–[30]. Among them, the only one that explicitly studies the role of app stores in requirements elicitation is the one by Al-Subaihin *et al.* [11]. This highlights that ASE is a common practice, but it does not give indications on how this activity is performed or its associated issues.

Contribution. With respect to related works, this is the first one that: (1) provides a list of strategies, benefits, and challenges of ASE and compares it to IBE; (2) deeply investigates ASE in practice, instead of focusing on its automation, or its general adoption by practitioners. Furthermore, while in a recent work [31] we analysed the results of combining IBE with ASE from a quantitative standpoint, in this paper we focus on qualitative insights, thus providing an orthogonal and complementary contribution.

III. Study Design

The proposed study can be classified as an *experimental simulation*, in which we want to evaluate the natural behaviour of analysts in a contrived settings [15], as done in other studies about interviews [32]–[35]. In an experimental simulation, one considers a realistic context and attempts to reproduce most of its characteristics in a fictional setting, so that the behaviors of subjects and other phenomena can be accurately observed. This is a compromise between the control over behaviors, which one can have with an experiment, and context-dependent insight, which one could obtain with a case study. We deem this strategy appropriate to evaluate the IBE and ASE *in context* while keeping a sufficient degree of control. Furthermore, this allows us to make a more in-depth analysis, e.g., with respect to surveys, and to consider human aspects, which have a limited role in tool proposals.

To perform our study, we recruited 58 participants, and we set up a two-phase process of requirements elicitation. In the first phase, *interview-based elicitation* (IBE), the analysts perform two interviews with a customer and then document the requirements. In the second phase, *app-store inspired elicitation* (ASE), the analysts extend the initial requirements based on similar products retrieved from the app stores. We systematically collect and code analysts' reflections, to produce a list of strategies, benefits, and challenges in relation to ASE. These are also compared with those observed by the analysts in relation to IBE, to identify differences.

It should be remarked that different results may be obtained if the two steps were reordered (first ASE, then IBE), and our results apply only to the ordering scenario presented in the paper. This is an *intentional* decision since our goal is to observe interviews and app store analysis in the realistic case in which an app is developed traditionally for a specific customer, and is then generalized for a wider public. The enactment of the alternative scenario, in which the analyst first develops a market-oriented app, and then a customized one, is not considered and could lead to different conclusions.

A. Study Participants

As requirements analysts, we recruited 58 graduate students enrolled in the first or second semester of the Master in Software Engineering at Kennesaw State University, GA, USA. At the time of the experiment, they were all taking a course on Requirements Engineering, in which they have been introduced to elicitation techniques and user stories. 80% of the students have already some professional experience. In

particular, 48.33% have *intense* professional experience, i.e., have covered a variety of roles including software engineers, developers, consultants, and defense contractors. The other 31.67% have a less significant professional experience in the field and, after an undergraduate degree in a computing-related discipline, have worked either in close fields or have just research or teaching assistant experience. The remaining 20% of the students are "career changers", i.e., students who have an undergraduate degree in a non-computing related discipline. They usually have some working experience in their field, and have transitioned to computing through a certificate in which they have learned programming, algorithms, computing foundations, and software engineering. The cohort of participants thus includes both professionals and novices.

B. Research Questions (RQs)

The following RQs are addressed in our study.

- **RQ1:** *What are the strategies adopted for the selection of inspirational apps to support ASE?* The question aims to categorise the typical strategies used by analysts to select similar apps. To answer the question, we collect the comments provided by the analysts to motivate their selection, and we analyse the data to produce a classification of typical strategies.
- **RQ2:** *What are the benefits and challenges of ASE?* The question aims to understand what are the advantages of using similar apps to take inspiration for novel requirements, and what are the difficulties encountered by the analysts. To answer the question, we collect the comments of the analysts in this regard.
- **RQ3:** *What are the differences between ASE and IBE?* This question aims to provide a comparison between the two elicitation styles in the considered realistic setting in which IBE is performed first and ASE is performed afterward. To answer the question, we categorise the benefits and challenges of interviews, and we critically analyse the differences with the categorisation in RQ2.

The steps of the data analysis and collection procedure (approved by the Institutional Review Board of Kennesaw State University, GA, USA) are described in the following, and depicted in Fig. 1. Tasks are depicted with single margin, while double margin indicates data. Tasks 1—8 concern activities for data collection, while 9—14 are for data analysis. All the steps in the figure have numbers that correspond to their descriptions in the following sections.

C. Data Collection

Tasks 1—5 concern IBE, including all the activities that go from initial customer ideas to documented requirements. These steps are based on the design of the experiment on elicitation interviews by Debnath *et al.* [32]. Tasks 6—8 concern ASE. For IBE, we do not report all the steps in the figure, since this paper mainly focuses on ASE.

1. Preparation Analysts are given a brief description of an app to develop and are asked to prepare questions for a customer that they will interview to elicit the products' requirements. The product is an app for the management of summer camps. A fictional customer is required to study a set of about 50 user stories, which are regarded as the initial customer ideas for the experiment. The user stories are taken from the dataset by Dalpiaz [36], file `g21.badcamp.txt`. The use of the same customer for all the interviews is in line with similar experiments, such as [34] and [37].

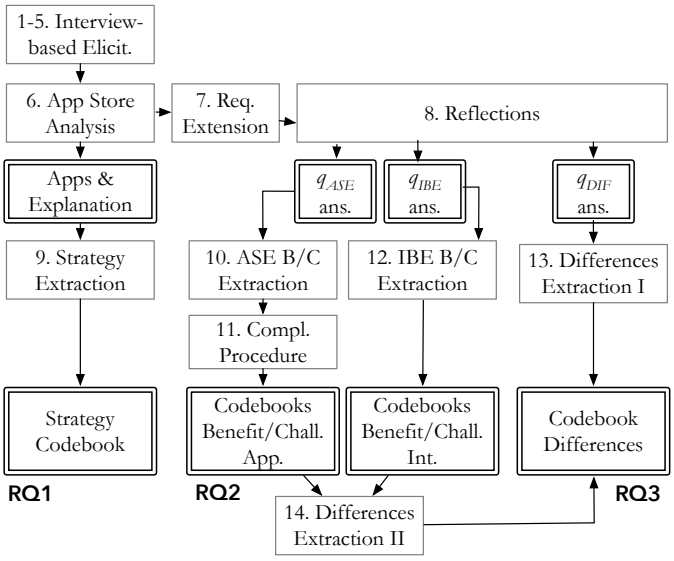

Fig. 1. Data Collection and Analysis Procedures.

2. Interview Each analyst performs a 15 minutes interview with the customer, possibly asking additional questions with respect to the ones that they prepared. The customer answers based on the set of user stories that describe the product—which are not shown to the analysts, to increase realism. The analysts are required to record their interviews, and take notes.

3. Requirements Analysis Based on the recording and their notes, the analysts have to: (a) perform an initial analysis of the requirements, and based on this analysis (b) produce additional questions for the customer to be asked in a follow-up interview.

4. Follow-up Interview The analysts perform a follow-up interview with the customer, which also lasts 15 minutes, to ask the additional questions prepared.

5. Requirements Documentation After the second interview, they are required to write down from 50 to 60 user stories for the system. We constrain the number of user stories between 50 and 60 to be consistent with the number of user stories in the original set. About 50 is also the typical number in the dataset by Dalpiaz [36], which we deem representative of user story sets used for research purposes.

6. Analysis of App Stores To explore and get inspired by possible competing products available on the market, the analysts are asked to perform an informal market analysis based on the app stores, comprising the following steps:

- Select at least 5 mobile apps from Google Play or the Apple App Store that are in some way related to the developed product (e.g., other apps for summer camps, apps for trekking, or anything that they consider related).

- Try out the selected apps to have an idea of their features when compared with their product.
- Go through the app reviews to identify desired features, and additional requirements that may be appropriate also for their product.

No automated tool, except for the default app store search engines, was provided for the market analysis task, and the analysts were free to browse the app stores following their intuition. The goal was to avoid confounding factors, i.e., the usage of a tool, and also to elicit challenges that could be addressed through tools, including possibly existing ones. Based on the analysis of the app stores, the analysts are required to list the selected apps and their links, together with a brief description that 1. outlines the main features of the product, 2. explains in which way the product is related to the original one, and why they have chosen it.

7. Requirements Extension Based on the analysis of the app stores, the analysts are asked to add 20 user stories to their original list.

8. Reflections The analysts are asked to fill out a questionnaire to reflect on the experience. This includes three open-ended questions: q_{ASE}) What are the benefits and challenges of ASE?; q_{IBE}) What are the benefits and challenges of IBE?; q_{DIF}) What are the differences between IBE and ASE?

D. Data Analysis

Data analysis is performed by two researchers, by means of thematic analysis according to Braun and Clarke [38], within a *critical realist* framework, using a *theoretical* approach (i.e., RQ-driven), and following the guidelines by Saldaña for the coding activity [39]. In the following, we report the analysis performed for each RQ, with examples of how the themes were produced. For RQ2, we also complemented the analysis with a literature review, and for RQ3 we included a brainstorming activity to analyse the differences between elicitation styles. For the complete and more detailed protocol, the codebooks, and the coded data, we refer to the supplementary material [40].

9. RQ1: Strategy Extraction The input data are the brief descriptions of apps with motivations for the choice, cf. step 7. The data from the 58 analysts include 295 items, one for each app. These were coded in multiple iterations of open and closed coding by the researchers. Codes were cross-checked to come to a consolidated set of themes, which were then re-applied to the data. For example, a statement such as "*This app is related to my app because it is a resource for people who are interested in camping and other outdoor events and activities*" was associated with the theme *user profile similarity*. At the end of the activity, the researchers redacted a codebook including: strategy name; description; example cases.

10. RQ2: ASE Benefits/Challenges Extraction The input data are the reflections from step 8, answers to q_{ASE}. An iterative process of open and closed coding was followed also in this case. For example, the item containing the text "*getting hands-on experience with potential features and evaluating their implementation*" was coded among benefits as *hands-on evaluation*. Instead, "*User reviews are a majority either based on total hatred of a product or complete satisfaction. These two extremes do not lead to productive identification of what requirements are missing in a product*" was coded among challenges as *polarised reviews*. The researchers aggregated the codes into common categories (i.e., higher-level themes), and produced two codebooks, one for benefits, and one for challenges, with: category; benefit/challenge; description; examples.

11. RQ2: Complementary Procedure to complement benefits and challenges of ASE, we performed a lightweight systematic literature review (SLR) following the guidelines by Kitchenham [41]. Specifically, we queried the Scopus engine with the following string ((benefit* OR challenge*) AND ((app AND store*) OR (app AND review*)) on title, abstract, and keywords, selecting only the Computer Science subject area, and specific venues (e.g., IEEE TSE, Springer EMSE). The full string is available in our supplementary material [40]. Scopus' coverage is considered optimal when compared to other databases (e.g., IEEE Xplore, ACM Digital library) [42]. The highly selective search string identified 82 publications. We set as main inclusion criterion "*The paper mentions one or more benefits or challenges of app store analysis*". After screening the papers based on this criterion, we finally selected 35 items. These were analysed by the researchers with open coding, to extract further benefits and challenges that could apply to ASE. We remark that the goal of the SLR was to *complement* the benefits and challenges identified. In the SLR, we did not identify any comprehensive analysis of benefits/challenges, and many items were not found in the collected papers. The reason is that other papers tend to focus solely on one single aspect (e.g., review classification), and none of them qualitatively studies app store analysis as a whole task. Furthermore, most papers focus on *solutions* to single issues, rather than analyzing how ASE occurs in practice.

12. RQ3: IBE Benefits/Challenges Extraction The input data are the reflections from step 8, answers to q_{IBE}. An analysis analogous to the one described in step 10 was carried out on the answers to q_{IBE}, to produce two codebooks with benefits and challenges of IBE. These are used in step 14. We do not perform an SLR in this case, as ASE is our main focus.

13. RQ3: Differences Extraction I Thematic analysis was performed on the reflections from step 8, answers to q_{DIF}, to produce a first comparative table, contrasting the characteristics of ASE and IBE. For example, the sentence "*interviews are designed to elicit what the stakeholder wants, and product-inspired elicitation is designed to elicit what the stakeholder potentially didn't know he wanted or needed*" led to two contrasting items: (ASE) "*identifies what the stakeholder could need*" vs (IBE) "*identifies what the stakeholder wants*". This led to a first version of the comparative table.

14. RQ3: Differences Extraction II To complete the table, the researchers jointly analysed the codebooks for ASE and IBE (four codebooks). For each item in each codebook of an elicitation style, they brainstormed on possible differences with the other style, considering the other themes in the other codebooks. For example, in relation to the challenge

for IBE previously coded as *conflicting requirements*, the researchers identified the contrasting theme *conflicting reviews* (cf. Table I). The contrast was reformulated as (IBE) "*Need to deal with stakeholders' inconsistencies*" vs (ASE) "*Need to deal with user review inconsistencies*". The activity led to the final version of the comparative table (Table II).

IV. Results

A. RQ1: Strategies

In the following, we report the set of different strategies identified, together with representative quotes from our data (in *italic*). The strategy for the selection of an app is not unique, in the sense that multiple strategies could be combined to drive the selection of a certain app. To better understand the quotes, it is useful to report the features of the original app idea. This is an app for the management of summer camps, including the following features: (1) managing the information, registration, and activities of the participants, (2) giving the participants' guardians the opportunity to register and follow their children, (3) managing the employees' performance and schedule, (4) communicating with parents and employees, and (5) social multi-media features. The following strategies are identified.

- **similarity by functionality enhancement**: the app implements *only one* functionality that is considered to be missing, or not sufficiently developed, in the tool. To select these apps, one can assume that analysts strategically considered the elicited requirements, and searched for products implementing specific functionalities, e.g., *"Sling is an app to manage employee scheduling. [...] This app is related to my product through its messaging and schedule building features."*
- **lexical similarity**: the app is considered because it has been likely returned by the available search engines with straightforward app-related keywords ("summer camp" or "camping", in our case) but does not have much to do with the original app. This rationale is not explicitly stated by the analysts, but it is visible in the following statement, in which the identified similarity is rather minimal: "*The Dyrt is a camping app that lets you get access to camping information in the US. [...] This app is related to the developed product because there is a feature for you to create a profile and also leave reviews about the campground*". The Dyrt is one of the first apps retrieved when querying Google Play with "camping".
- **similarity by functionality subset**: only one specific subset of functionalities of the app is considered as a possible inspiration. This case frequently occurred when the analysts retrieved an app based on lexical similarity and then found that certain functionalities could be considered. For example, considering the app *Recreation.gov* (again a camping app) an analyst said: *"Regarding the system being created, allowing the users to select what activities they want would be beneficial. They can see photos of the activities and what they will be doing [...]. The registration will be secure when the campers make their account, and they will be able to see real-time feed"*.
- **domain similarity**: the app is considered because it belongs to a similar application domain, e.g., hiking, orienteering, outdoor games. The domain similarity of a certain app can enable the identification of interesting functionalities not initially planned. For example, *Cairn*, a hiking app to keep users safe and connected, allows users to "*set the paths that they are taking and share them with friends and family. If they are not back from the hike by a certain time that the user has chosen, the app will send a notification to their friends*".
- **common use** the app is well-known and commonly used by a large user base (e.g. Facebook, Instagram), and it has been likely chosen without searching the app store. For example, *Garmin* is considered because "*[the stakeholder] wants a means to track guests and staff while on the campgrounds. The Garmin wearable allows for the device to transmit GPS coordinates which can be used for tracking*".
- **user profile similarity**: as exemplified in Section III-D (step 9), the app is selected because the profile or interest of potential users is considered similar.
- **similarity by software scope**: the app is domain-agnostic or belongs to a completely different domain, but it has a similar general scope (e.g., management). Considering these apps can help to build a generic product, but also to enhance existing features: "*This app allows users to create custom business apps for yourself and your team. The app comes with templates for inventories, invoicing/accounting, [...] it has so many of the features necessary to manage a business*".
- **generic product**: the app belongs to the same or similar application domain, and it is a context-independent product of the specific software, specifically designed to reach a broader audience. These apps can be particularly useful to identify generic requirements, which cannot be gathered through interviews in single specific contexts: "*The application is customized for each camp site and requires specific log on information. This is an additional requirement for this application to be marketable to many other businesses, and certainly would not have been covered as part of the interviews as the interviewee has no interest in a product for anyone else*".
- **similarity by business mission**: the app is similar because the overall goal of the *business* is similar, e.g., education. Here, the similarity is not specifically driven by the software features, but by the mission of the actual business supported by the software. For example, in the app of the daycare *Kriyo School*, "*if you are an administrator of a daycare or preschool you can create an account to manage any aspect of the daycare. This could be the broader audience for the product.*"
- **full match**: the app belongs to the exact same application domain, and it is implementing the same functionalities. Looking at these apps enables analysts to mimic certain features. *CampMinder*, an application specific for summer camp management, is a full match, as "*it*

allows for users to access the records associated with the children in their camp and it specifically integrates online registration and forms".

- **popularity** the app is selected because it appears to be widely used or highly rated/awarded. For example, as "*there is no better tool for managing customers than a good CRM [Customer Resource Manager]*", some analysts select CRMs. Among them, *HubSpot* has been selected as it "*is a fairly famous CRM product that I think would have good features to get inspired by*".

It should be noted that, while in some cases the analysts strictly relied on the output of the app search engines on the basis of simple app-related queries (e.g., *functionality subset, lexical, domain, generic product*), in other cases they appear to have made an effort to think and search more strategically based on a rationale for extension of their product (e.g., *functionality enhancement, common use*).

B. RQ2: Benefits and Challenges

Table I reports the summary of the results for RQ2. The superscript * indicates items that were identified through the SLR. In the following, we consider the main categories, and present representative themes, with associated example quotes.

1) Benefits: these are divided into *concept inspiration*, *requirements inspiration*, *implementation inspiration*, *user satisfaction*, *market satisfaction* and *process support*.

a) Concept Inspiration: looking at a variety of different products can generate novel ideas for adaptations of the app and broaden the scope and *vision* of the initial idea [43], leading to a transformation of the original product to satisfy a possibly different market. In this regard, one of the analysts said "*[ASE] gave me a different outlook on how to gather new ideas*" (**inspiration for ideas**).

b) Requirements Inspiration: looking into other products helps to identify new possible features, extend the existing requirements, or better define lower-level ones, thanks to the possibility of analysing product implementations: "*best features that are successful in other products can be brought in, new ideas can be implemented by looking at other products*" (**identify novel features**).

Looking at the market can also help analysts to overcome their lack of domain knowledge, which is a central issue in requirements elicitation activities [44], [45], and can possibly lead to the identification of tacit/unknown requirements [46], [47]. For example, one analyst said that ASE "*can help to shape and provide more specific requirements. It is focused on the application functionality and could also help when there is not a lot of domain knowledge about the product*" (**overcoming the lack of domain knowledge**).

c) Implementation Inspiration: ASE can provide ideas to guide the development phase. Problems with certain implementations can be easily identified, by looking at other products and their users' feedback, which "*can show you issues users had with these products, so you can attempt to avoid those issues when designing your product*" (**issue identification**). Identification of common issues is particularly useful, as many apps tend to use similar libraries [48]. Irrelevant features and enhancements of existing ones can be discovered, as well as possible implementation solutions. To this end, *hands-on* evaluation of existing products is a most valuable support. Indeed, *"You can even try out the functionality to see how things look and feel."* (**hands-on evaluation**). ASE can also help to implement a product that is more adaptable to different needs, as products in the app store are generally oriented towards a broad audience:"*[ASE] gives us a bigger picture of how the features could be implemented and what other requirements could be developed later so the system would be ready for such features later."* (**create adaptable product**). Overall, analysts and developers can build upon

Benefits	Challenges
CONCEPT	
Concept Inspiration	***Product Concept***
inspiration for ideas	limitation of creativity
broaden vision	loss of original purpose
	no support for initial idea
	risk to copy products
REQUIREMENTS	
Requirements Inspiration	***Requirements and Features Definition***
identify novel features	difficult to understand goals
expanding requirements	irrelevant requirements
narrowing requirements	relevant/key feature selection
identify unknown requirements	difficult adaptation of features
overcoming the lack of domain knowledge	risk to miss relevant requirements
	identification of demographic preferences*
PRODUCT	
Implementation Inspiration	***Product Search and Evaluation***
issue identification	difficult to identify relevant products
leverage developers knowledge	too many similar products
hands-on evaluation	insufficiency of the app store
inspiration for implementation	static categorisation*
enhancing product	need to log-in to evaluate products
create adaptable product	need to purchase app functions
identify irrelevant elements	problems with ads*
	repackaged apps*
USER	
Market Satisfaction	***Reviews (Quality)***
identify market needs	uninformative reviews
identify market trends	vague/poor reviews
create competitive product	short reviews
identify successful solutions	unstructured reviews*
User Satisfaction	fake reviews*
improve usability	***Reviews (Content)***
improve accessibility*	polarised reviews
user feedback	limited constructive criticism
assess usability	conflicting reviews
consider large audience	partial information*
satisfy variety of users	reviews do not comment features
identify user values	***Reviews (Quantity)***
	uncategorised reviews*
	too many reviews
	insufficient number of reviews
	filtering reviews
PROCESS	
Process Support	***Process Weaknesses***
	time consuming activity
limited stress	unfocused activity
save development time	difficult to trace stakeholders
idea validation	requirements validation
assess feasibility	intellectual property issues
support decision process	no follow-up

TABLE I
BENEFITS AND CHALLENGES OF ASE

the knowledge of other developers through their implemented solutions, thereby overcoming their possible limitations in terms of competences.

d) User Satisfaction: trying out other products and checking their reviews can help to assess whether certain solutions are usable or not, possibly preventing usability issues: *"as I previewed some of the software currently available, I can see how the interfaces may be easier to use"* (**assess usability**). Usability comments are frequently associated with lower ratings with respect to other types of reviews [49]. Also, looking at user feedback can also identify accessibility issues [50]. In addition, user feedback informs the analyst on what is required and what is not needed by users of similar apps, and even infer what are user *values*, which are related to the core benefits that one expects from existing apps: "*the end-goal should not be about how well they created something; it should be about how much value this brings to the customer*" (**identify user values**). As suggested by Sutcliffe *et al.* [51], user values can motivate the download and use of certain apps.

Finally, looking at similar products also helps to understand what are the different types of potential user profiles, and how to satisfy them. With ASE, "*it becomes possible to create a system that is more likely to meet the needs of a variety of users*" (**satisfy variety of users**). User profiling is recognised as particularly important for personalised applications, especially when they include some form of content recommender system [52], e.g., music or video apps.

e) Market Satisfaction: by looking into apps and their reviews, analysts can better identify what is required by the market: *"[ASE] gives an opportunity to understand what current systems are fulfilling the user needs and also the unfulfilled users requirements."* (**identify market needs**). By monitoring the market trends, one can understand how the market evolves, thus anticipating future needs. Looking at other products satisfying similar requirements can be a reality test to check whether the product idea is sufficiently in line with the market expectations, and in which way it needs to be improved to achieve a competitive advantage: *"I feel that if I had the opportunity to implement these features into the product that it would have a fair chance competing in the app marketplace*" (**create competitive product**). In this regard, the analysis of successful solutions, based on the number of downloads and positive reviews—typically considered as indicators of popularity [53]—, can be particularly helpful to understand what is the current benchmark.

f) Process Support: different phases and aspects of the development process can be supported by ASE. Specifically, it can help during the initial validation of previous ideas, and also to assess their feasibility. If similar products have implemented certain solutions, these should not pose insurmountable technical barriers: "*The benefit of ASE is that you get your requirement off a product that already exists so you know that whatever requirement gotten from this process is feasible for the intended product*" (**idea validation**; **assess feasibility**). Since there is no interaction with stakeholders, ASE is not considered stressful, and can also save development time by helping to estimate it. Indeed, "*building something from scratch is hard and time inefficient, so by taking a look at what is out there one can gauge factors such as development time"* (**save development time**). From the development standpoint, understanding what is relevant in competing products can facilitate the prioritisation of features and better direct the decision process towards implementation.

2) Challenges: these are divided into *product concept*, *requirements and features definition*, *product search and evaluation*, *reviews*, and *process weaknesses*.

a) Product concept: while ASE can be useful to extend a product idea, it does not provide sufficient support when one needs to define a novel concept from scratch. As noted by one of the analysts, "*if we are building an application from scratch, I think it is not that useful as we are still working on building the basic functionality of the product.* " (**no support for initial idea**). Furthermore, looking at other products can lead to a limitation of creativity, and also to the risk of copying other products: "*The danger of ASE is to avoid simply copying another product that exists. You want to design a product that is unique to your customer*" (**risk to copy products**). Finally, by integrating more and more features adapted from other products, one could lose the original purpose of the app under development.

b) Requirements and features definition: while ASE can drive implementation, it provides little help in understanding possible high-level goals of the stakeholders, as what is available is only the software and its reviews. Given the absence of well-defined goals, it is difficult to understand what are the relevant, key features to select, and, without the stakeholders at hand, there is a risk of missing relevant requirements. In addition, after selecting features, it is also hard to adapt and integrate them into an existing product feature set, as "*it's necessary to try the products and understand how the products work. It's the understanding that helps SE determine how features can be modified and adapted to work for their clients most effectively*" (**difficult adaptation of features**). Demographic preferences are also hard to identify, especially when one aims to develop an app for different countries [54].

c) Product search and evaluation: the app store is not designed to specifically support ASE, and it does not offer a structured way for comparison, as it happens, e.g., on the Amazon marketplace. A product search typically returns a set of too many similar products, with several features to compare. Furthermore, their categorisation is static, and not in line with the evolution of the market [55]. Additional confusion to the search results is introduced by repackaged apps [56]. The information overload makes it difficult to identify which products can be relevant for inspiration. On this topic, one of the analysts said "*The challenge I faced [during ASE] is that there are not many applications available on the App Store in the camping genre. So, I had to do some intense research, surf many websites, and find links to applications from the websites, which took a lot of time*" (**insufficiency of the app store**). Evaluating products often requires creating an account to log-in, which complicates the evaluation. Furthermore, the

test of many relevant features often requires a subscription to a premium account, thus making the comparison of several products a costly activity from the economic standpoint, or an incomplete one in case of limited resources. When one uses free app versions, fair evaluation is often complicated by an excessive number of ads [57].

d) Reviews: reviews pose challenges in terms of quality, content, and quantity. Specifically, they are often vague and superficial, poorly written, short, and thus not informative. Their format is unstructured, which makes them hard to analyse [58]. One of the analysts, discouraged by the quality of the reviews stated: "*one must sift through the countless valueless comments to find the gold nuggets that are missing, nonfunctional, or unnecessary features. People are naturally lazy, and their reviews and comments normally reflect that*" (**uninformative reviews**). In terms of content, reviews are often highly polarised with enthusiastic comments or extremely negative ones. The former usually praise the product and do not contain any useful suggestions and the latter often do not include constructive criticism that can be exploited by analysts or developers. The polarisation also leads to conflicting reviews, and it is hard to understand which opinion is more trustworthy, also due to the problem of fake reviews [59]. On review inconsistency, one of the analysts said "*ASE has a lot to do with what you personally see and feel plus the advice and comments from the masses. This can be hard to filter through because people may not always know what they want or there are so many conflicting reviews*" (**conflicting reviews**). Also, reviews tend to comment on the product as a whole, and it is hard to find reviews that comment on features, thereby making tools for associating user sentiment to features in-app reviews particularly useful [60]. The quantity of reviews is also an issue, especially combined with their quality. Filtering tools that can select only relevant reviews based on a certain query (e.g., Appbot) can provide valuable support, as they help to identify review categories on-demand, and address the problem of uncategorised reviews [61]. In some cases, the number of reviews can also be insufficient to get an idea of the product. This happens especially if one wants to develop a system for a rather limited market—such as in our experimental simulation—where similar apps may not be sufficiently popular to have a substantial volume of reviews. Both extremes in the number of reviews can be a problem: "*Depending on the product's popularity, reviews can be insufficient or abundant. When there are a lot of reviews you need to find the right ones that are helpful and are not biased and blatant*" (**too many reviews**; **insufficient number of reviews**). In addition, even when high-quality reviews are available, these always include partial information to make sense of the raised issues, and need to be complemented with external sources such as app crash reports, tweets, community blogs and code repositories [2], [6], [19].

e) Process weaknesses: searching, selecting, and analysing products are quick activities compared to developing prototypes, but they unavoidably require time. Since there are no structured guidelines, they also tend to be unfocused, without a linear or directed process. For example, an analyst pointed out that ASE "*is a time-consuming process that requires hours of research to find the best products to compare and then even longer to comb through the publicly available reviews for the given product.* " (**time-consuming activity**; **unfocused activity**). It is also difficult to understand who are the stakeholders who can use a certain product. As stated by one of the analysts "*the requirement you get [from ASE] cannot be traced to the accurate stakeholder so that you can have a better understanding of why the requirement is needed*" (**difficult to trace stakeholders**). In addition, one cannot have follow-up questions with the users who left an unclear review, or with those that appear to have something more to say about, e.g., a certain bug and its reproduction. Since the stakeholders cannot be contacted for in-depth interactions, requirements validation is also not possible. This has been noted by some analysts as one of the biggest challenges with ASE: "*my biggest challenge in applying this type of elicitation is validating the requirements. I had to make decisions based on what I have received from the user. Therefore, any inclusion of ambiguous requirements will lead to modeling and production of a bad product*" (**requirements validation**). Borrowing features from other products can also potentially lead to intellectual property issues, which product developers could raise after the app is released to the public.

C. RQ3: Differences

Table II reports the identified differences between IBE and ASE, divided into six categories. The table also reports whether a certain characteristic is positive (+), negative (-) or neutral ($\sim$)—this evaluation is arguably made by the authors.

a) Main focus: the main focus of IBE is eliciting goals, thus answering *why* questions, while ASE aims to understand what could be the possible solutions to address certain goals and answer *how* questions. To this end, IBE is based on asking explicit inquiries that can inform future development (*forward thinking*), while ABE is based on observing implementations, and thus relies on previously developed apps (*backward thinking*). As pointed out by one of the analysts, "*Interviews help us during the initial phase of software development [...] where product-inspired elicitation is useful when we are working on improving the system based on customer feedback, review and bug reporting*" (**appropriate for initial development stage** vs **appropriate for later development stage**). So, interviews are useful to create personalised products and can help the analysts in the initial phases of the development, when they need to perform problem decomposition. Instead, ABE helps to develop novel ideas to create a general product oriented to satisfy market needs.

b) Elicitation of needs: concerning this aspect, greater advantages are observed for IBE, compared to ASE. In IBE, needs are directly elicited from stakeholders, and the analyst can ask detailed questions, which need to be well formulated to acquire relevant, and mainly qualitative, information. ASE relies on the analysis of user feedback to understand the customers' needs, and collection of details is incidental and

	Interview-based Elicitation	App Store-inspired Elicitation	
	Main focus		
+	Focus on goals (WHY)	Focus on solutions (HOW)	+
+	Focus on asking	Focus on observing	+
+	Focus on future development (forward thinking)	Focus on past development (backward thinking)	+
+	Personalised product oriented so satisfy a customer	General product oriented to satisfy market needs	+
+	Support problem decomposition	Support idea generation	+
+	Appropriate for initial development stage	Appropriate for later development stages	+
	Elicitation of needs		
+	Relies on direct stakeholder interaction	Relies on analysis of user feedback	+
+	Explicit questions about details	Collection of details is incidental	-
∼	Relies on proper questions	Relies on proper search queries	∼
+	Mainly qualitative information	Qualitative and quantitative information from reviews	+
+	Can help to collect stakeholders' goals	Stakeholders' goals are hard to identify	-
+	Explicit questions to stakeholders	Questions to stakeholders not possible	-
	Interpretation of needs		
+	Identify what the stakeholder wants	Identify what the stakeholders could need	+
+	Requirements within the project scope	Can lead to out of scope requirements	-
+	Probing/follow-up questions can reduce misinterpretations	Reviews can be misinterpreted	-
+	Explicit answers can remove wrong assumptions	Wrong assumptions can be made about user needs	-
+	Understand product goals and vision	Create product goals and vision	+
∼	Need to deal with stakeholders' inconsistencies	Need to deal with user review inconsistencies	∼
+	Success depends on agreed criteria	Success also depends on luck factors	-
	Relationship with stakeholders		
+	Fosters stakeholders' relationship	Limited relationship with stakeholders	-
+	Can change stakeholders' viewpoint	Does not affect stakeholders' viewpoint	-
+	Can exploit non-verbal cues	No face-to-face interaction	-
+	Allow access to multiple stakeholders	Access only to users	-
	Process		
+	Structured process	Unstructured process	-
∼	Requires soft skills	Requires technical skills	∼
-	Stakeholder-directed process	Creative process	+
-	Stressful activity	Not stressful activity	+
-	Requires preparation	No preparation needed	+
-	Requires time management	No time constraints	+
-	More time consuming	Less time consuming	+
-	Requires active control of the conversation	Inherent control of the analysis process	+
	Requirements assessment		
+	Requirements can be validated by the customer	Difficult to validate requirements	-
-	Evaluation possible only with prototypes/mockups	Hands-on evaluation of requirements implementation	+
-	Usability requirements not testable	Can enable the test of usability requirements	+
-	Feasibility cannot be assessed	Can enable assessment of feasibility	+

TABLE II
DIFFERENCES BETWEEN INTERVIEW-BASED AND APP STORE-INSPIRED ELICITATION.

not driven by the analyst's investigation. The technique relies on proper search queries, and can be useful to collect both qualitative and quantitative information (number of downloads, rating). While interviews can help to collect goals, these are harder to identify with ASE, because the analyst cannot pose explicit questions to the stakeholders: *"it [ASE] can inspire new ideas [...], but you will not be able to elicit new goals from product-inspired elicitation"* (**can help to collect stakeholders' goals** vs **stakeholders' goals hard to identify**).

c) Interpretation of needs: this aspect is facilitated by IBE, while some challenges exist for ASE. IBE helps to identify what the stakeholders need, while ASE helps to interpret what they *could* need. In this sense, the former helps to *understand*, while the latter aims to *create* product goals and vision. With IBE, elicited requirements remain within the project scope thanks to the continuous exchange of information with the stakeholders, where probing/follow-up questions can reduce misinterpretations, and remove wrong assumptions. ASE can lead to requirements that are out of the project scope, and misinterpretations of the users' voice expressed through reviews is likely, possibly leading to wrong assumptions: "*The other main difference is that it is much easy to gain clarity from the client in an interview [...] In product-inspired elicitation you are making decisions or assumptions based on your interpretation of the product so you may have one understanding of the project and its needs which may not necessarily line up with the client*" (**explicit answers can remove wrong assumptions** vs **wrong assumptions can be made about user needs**). While with IBE application success depends on agreed criteria, with ASE it depends also on luck factors, as the market can be moody, and search engines cannot be fully controlled. Inconsistency is a common pain point between IBE and ASE, with conflicting stakeholder requirements and conflicting reviews.

d) Relationship with stakeholders: IBE revolves around the creation of rapport and fostering good relationships with the stakeholders, while with ASE the stakeholders are not reachable and no actual relationship can be established: *"The main difference between interviews and product-inspired elicitation, is the fact that in the interview, I was able to get a better connection to the human needing something"* (**foster stakeholders' relationship** vs **limited relationship with stakeholders**). With IBE one can engage in dialogue and possibly change the stakeholders' viewpoint in case of misunderstanding, which is not possible with ASE. During dialogues, non-verbal cues, such as facial expressions and body language in general, can be exploited to better reveal the stakeholders' inner feelings and thoughts, while no face-

to-face, synchronous interaction is possible with ASE. Finally, reviews used in ASE are typically written by users, while interviews can reach a larger set of stakeholder types, e.g., domain experts, sponsors.

e) Process: while IBE can follow a structured, mainly sequential, process based on interview scripts prepared beforehand, with ASE the process is unstructured and iterative, as no guideline exists to perform it. The skills required for the two approaches are different, as IBE requires soft skills to understand and create rapport with stakeholders, while with ASE technical skills are central to understand what certain solutions entail from the development standpoint, and how they can be incorporated into an existing app: "*The main difference is that one is a soft skill, and the other is technical. When interviewing, you have to think on your feet, adjust conversation based on stakeholder needs, adjust, build rapport, and make the stakeholder feel comfortable enough to share real goals with you. Product-inspired elicitation is a technical skill*" (**requires soft skills** vs **requires technical skills**). Concerning other process-related aspects, ASE has several advantages over IBE. Specifically, the IBE process is mainly directed by the stakeholders, and in particular, the customer and the sponsor, while ASE is a creative process. IBE can be also stressful, as interaction with other, unknown people, can create some tension. Interviews require preparation, while with ASE one is free to improvise, and no specific planning is necessarily needed. Another crucial difference concerns *time*. The involvement of stakeholders in IBE requires to schedule appointments in advance, and manage time well during the interview. Instead, with ASE the analyst has full control of time and of the analysis process and does not need to depend on others' schedules or actively control the conversation to properly manage time and information acquisition.

f) Requirements assessment: this aspect is generally easier with ASE, except for requirements validation. In IBE requirements can be explicitly validated by the customer, while validation of ASE requirements implicitly comes after the product has been released on the market. On the other hand, ASE offers the possibility of hands-on evaluation of requirements implementations, while in IBE one can evaluate requirements only with prototypes/mockups. Furthermore, tests of usability requirements, and assessment of implementation feasibility, can be hard with IBE.

V. Discussion

a) Implication for researchers: Our results are based on a manual version of ASE (i.e., not supported by tools), and confirm the need for many of the technical solutions that have been developed by researchers. These include app review classifiers [4], [16], fake review detectors [59], review rationale identifiers [62], but also the more recent works on app comparison and feature recommendation [12]–[14], which would be the core of ASE. Together with other works [6], [11], this shows that there is a *practical need* for tool support in app store analysis, thus addressing the demand for more evidence in this regard observed by Dabrowski *et al.* [10]. While research tools address only one problem at a time, *integrated* platforms that collect multiple capabilities, considering our list of observed challenges, are required to fully exploit the potential of ASE. Furthermore, our analysts observed that this practice can be unfocused, as they were not able to devise an intuitive and structured process to perform it. We are not aware of guidelines for ASE, and researchers are called to define them. In this regard, the identified *strategies* can be taken as a starting point to provide suggestions on how to perform ASE in a fruitful manner, depending on the development stage of the product, e.g., using *user profile similarity* during market positioning or substantial app renewal, or *functionality enhancement*, when performing minor releases. The guidelines should be integrated into agile processes, as these are the most common in app development [29], [63]. Still on strategies, experimental evaluations can be carried out to assess which ones are more effective given a certain development goal, e.g., extending a product, or generalising it for a wider market. Implementing recommender systems in which users can tune the search strategy based on their needs is another research direction. Finally, our work calls researchers to better study traditional elicitation techniques, such as interviews, in combination with ASE. While these practices have been largely studied independently, our results show that they can provide *complementary* contributions to the development.

b) Implication for practitioners: The main message concerns the list of *benefits* of practicing ASE, which should encourage companies to invest more on it as a part of their development process, and not as a mere complementary activity. Among benefits, developers should primarily consider the assessment of feasibility—to be performed when planning for a certain feature—and usability—to be performed during GUI design, a core aspect of app development [29]. This suggests that different *stages* of development may profit from ASE. Also, different *roles* may exploit it, as, e.g., requirements analysts (concept inspiration), advanced developers (implementation inspiration), and novices. Indeed, one of the benefits of ASE is also the possibility of overcoming the lack of knowledge, by leveraging the competence of other developers. The activity can thus be particularly useful for young developers and can be potentially exploited as an onboarding exercise for companies, given the need for appropriate strategies in this regard, as observed by Britto *et al.* [64]. Training should push for the adoption of planned strategies for app search and selection, rather than clerical usage of search engines, as some of our analysts chose to do. App developers need also to be aware of ASE limitations, which can be addressed through IBE. For example, the possibility of explicitly eliciting goals and removing wrong assumptions. IBE can be applied both at the initial development stage, and later, to confirm the unclear feedback coming from reviews. ASE can then be fully exploited to generalise the app idea to more users. In case both strategies are used, practitioners should consider that different analyst profiles can be more appropriate, having soft-skills for interviewers, and technical skills for ASE analysts.

VI. Threats to Validity

a) Construct Validity: we used thematic analysis to analyse the data according to Braun and Clarke [38], and adopted the guidelines of Saldaña for coding [39]. We consider these well-established references suitable for our case, where a classification of themes emerging from participant responses is required. We did not adopt the more powerful Grounded Theory (GT) framework [65], given that our problem is focused on already pre-defined, intuitive, macro-categories (*strategies*, *benefits*, *challenges*, *differences*), and data collection is not intertwined with data analysis as in GT.

b) Internal Validity: the participants were involved in a course, and the overall assignment was part of their evaluation. This could have biased their activity, as the participants could feel compelled to produce more, and possibly unreliable, information. Furthermore, reflections were written *after* performing the different tasks, thus leading to possible recall bias. These aspects could not be entirely mitigated. To prevent participant bias, the customer was not a lecturer, but an external subject, well-trained in playing the customer. A total of 29h30min of interviews (15min x 2 x 59) in one month were required, which mitigates fatigue. Fatigue effects may still emerge, but here we aim to study qualitative aspects and not to make precise measurements, which further reduces the influence of fatigue effects on the results.

The qualitative analysis relies on data interpretation, which entails subjectivity. To mitigate this, we triangulated the produced codebooks between researchers, with open-coding activities followed by closed-coding ones on a different portion of the datasets, and with meetings to consolidate and homogenise the identified themes. We also share the raw and tagged data, the codebooks, including examples of participants' quotes for each theme identified, and we report the data analysis procedure with details [40], as recommended for qualitative studies [65].

c) External Validity: this is an experimental simulation, and thus its results may not apply to real-world cases. However, this type of research strategy is rather close to a study in a natural setting [15], thus reaching a reasonable compromise between realism and generalisability. The results apply to cases that are similar to ours, in which IBE is followed by ASE. As mentioned, this is an intentional choice, since our goal is to consider a realistic case in which an app is developed traditionally for a specific customer, and is then generalized for a wider public. An inversion of the tasks (first a market-oriented app, then a customized one) is also a reasonable elicitation scenario, though less common. Other benefits, challenges, and differences could be identified for that case, and would not be strictly comparable to ours.

Though we used students as participants, an accepted practice in software engineering [66], most of them are *professionals*, and thus our results combine the viewpoint of both novice and advanced analysts. A residual threat is the usage of a single case to elicit our data, similarly to other studies in requirements engineering [32], [33]. Following case-based generalisation [67], we deem the case as representative of social apps with different user profiles (e.g., managers, staff), multimedia-sharing features, and a specific application domain. This may have restricted the number of similar applications found by the participants during the experimental simulation. Different themes may be identified with other types of apps, e.g., more domain-generic ones. About the completeness of the findings, we performed a SLR to complement the codebooks. This applies to RQ2 and as a by-product to RQ3, as data from RQ2 were used as input.

d) Literature Review: we followed the widely adopted guidelines from Kitchenham [41], and adopted the assumptions of Martínez-Fernández *et al.* [42] for the choice of Scopus as a single search engine. The search string is narrow, and we may have unavoidably missed relevant publications. We argue that this risk is acceptable, given that the SLR is a secondary source of information. Finally, while our research focuses on ASE, the SLR searched for studies about app store analysis. Though this is a broader activity, (1) we are not aware of studies specifically focused on benefits/challenges of ASE, and (2) in our data extraction we considered solely those benefits/challenges that are applicable to ASE.

VII. Conclusion

ASE is the practice of using app stores as a source of inspiration by selecting apps considered relevant for the product under development, understanding how they work, and browsing their reviews. While ASE is typically used in industry during the development process, the literature offers very little insight into this practice. In this work, we contribute to filling this gap by identifying the most commonly used strategies to select apps and the benefit and challenges that analysts associate with ASE. We conducted an experimental simulation with 58 analysts and performed a systematic analysis of the collected data. The results of this work contribute to the *theory* of requirements engineering, providing the initial foundations of a well-known but yet not formalized activity, and outlining research and practice directions. In future works, we plan to: (1) survey practitioners to quantify how relevant are the identified benefits and challenges from their viewpoint; (2) evaluate the effect of specific inspiration strategies on the final requirements of a product.

Data Availability All the data associated with this study are publicly available [40].

Acknowledgment

This study was carried out within the MOST – Sustainable Mobility National Research Center and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 – D.D. 1033 17/06/2022, CN00000023). This work was also partially supported by National MIUR-PRIN 2020TL3X8X project T-LADIES (Typeful Language Adaptation for Dynamic, Interacting and Evolving Systems), by the and by the National Science Foundation under grant CCF-1718377.

REFERENCES

[1] O. Dieste and N. Juristo, "Systematic review and aggregation of empirical studies on elicitation techniques," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 283–304, 2010.

[2] N. Genc-Nayebi and A. Abran, "A systematic literature review: Opinion mining studies from mobile app store user reviews," *Journal of Systems and Software*, vol. 125, pp. 207–219, 2017.

[3] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *IEEE Transactions on Software Engineering*, vol. 43, no. 9, pp. 817–847, 2016.

[4] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Engineering*, vol. 21, no. 3, pp. 311–331, 2016.

[5] N. Jha and A. Mahmoud, "Mining non-functional requirements from app store reviews," *Empirical Software Engineering*, vol. 24, no. 6, pp. 3659–3695, 2019.

[6] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, "Recommending and localizing change requests for mobile apps based on user reviews," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 106–117.

[7] E. Noei, F. Zhang, S. Wang, and Y. Zou, "Towards prioritizing user-related issue reports of mobile applications," *Empirical Software Engineering*, vol. 24, no. 4, pp. 1964–1996, 2019.

[8] M. Haering, C. Stanik, and W. Maalej, "Automatically matching bug reports with related app reviews," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 970–981.

[9] E. Oehri and E. Guzman, "Same but different: Finding similar user feedback across multiple platforms and languages," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 2020, pp. 44–54.

[10] J. Dabrowski, E. Letier, A. Perini, and A. Susi, "Analysing app reviews for software engineering: A systematic literature review," *Empirical Software Engineering*, vol. 27, no. 2, pp. 1–63, 2022.

[11] A. A. Al-Subaihin, F. Sarro, S. Black, L. Capra, and M. Harman, "App store effects on software engineering practices," *IEEE Transactions on Software Engineering*, vol. 47, no. 2, pp. 300–319, 2021.

[12] H. Jiang, J. Zhang, X. Li, Z. Ren, D. Lo, X. Wu, and Z. Luo, "Recommending new features from mobile app descriptions," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 4, pp. 1–29, 2019.

[13] H. Liu, X. Yin, S. Song, S. Gao, and M. Zhang, "Mining detailed information from the description for app functions comparison," *IET Software*, vol. 16, no. 1, pp. 94–110, 2022.

[14] F. Dalpiaz and M. Parente, "Re-swot: From user feedback to requirements via competitor analysis," in *International working conference on requirements engineering: foundation for software quality*. Springer, 2019, pp. 55–70.

[15] K. Stol and B. Fitzgerald, "The ABC of software engineering research," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 3, pp. 11:1–11:51, 2018. [Online]. Available: https://doi.org/10.1145/3241743

[16] N. Jha and A. Mahmoud, "Using frame semantics for classifying and summarizing application store reviews," *Empirical Software Engineering*, vol. 23, no. 6, pp. 3734–3767, 2018.

[17] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 14–24.

[18] S. Scalabrino, G. Bavota, B. Russo, M. Di Penta, and R. Oliveto, "Listening to the crowd for the release planning of mobile apps," *IEEE Transactions on Software Engineering*, vol. 45, no. 1, pp. 68–86, 2017.

[19] M. Nayebi, H. Cho, and G. Ruhe, "App store mining is not enough for app improvement," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2764–2794, 2018.

[20] T. Johann, C. Stanik, W. Maalej *et al.*, "Safe: A simple approach for feature extraction from app descriptions and app reviews," in *2017 IEEE 25th international requirements engineering conference (RE)*. IEEE, 2017, pp. 21–30.

[21] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: Msr for app stores," in *2012 9th IEEE working conference on mining software repositories (MSR)*. IEEE, 2012, pp. 108–111.

[22] F. Sarro, M. Harman, Y. Jia, and Y. Zhang, "Customer rating reactions can be predicted purely using app features," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*. IEEE, 2018, pp. 76–87.

[23] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "Investigating the relationship between price, rating, and popularity in the blackberry world app store," *Information and Software Technology*, vol. 87, pp. 119–139, 2017.

[24] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli, "On-demand feature recommendations derived from mining public product descriptions," in *Proceedings of the 33rd international conference on software engineering*, 2011, pp. 181–190.

[25] N. H. Bakar, Z. M. Kasirun, and N. Salleh, "Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review," *Journal of Systems and Software*, vol. 106, pp. 132–149, 2015.

[26] A. Ferrari, G. O. Spagnolo, and F. Dell'Orletta, "Mining commonalities and variabilities from natural language documents," in *Proceedings of the 17th International Software Product Line Conference*, 2013, pp. 116–120.

[27] A. Holzer and J. Ondrus, "Mobile application market: A developer's perspective," *Telematics and informatics*, vol. 28, no. 1, pp. 22–31, 2011.

[28] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real challenges in mobile app development," in *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2013, pp. 15–24.

[29] R. Francese, C. Gravino, M. Risi, G. Scanniello, and G. Tortora, "Mobile app development and management: results from a qualitative investigation," in *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE, 2017, pp. 133–143.

[30] M. Nayebi, B. Adams, and G. Ruhe, "Release practices for mobile apps–what do users and developers think?" in *2016 ieee 23rd international conference on software analysis, evolution, and reengineering (saner)*, vol. 1. IEEE, 2016, pp. 552–562.

[31] A. Ferrari, P. Spoletini, and S. Debnath, "How do requirements evolve during elicitation? an empirical study combining interviews and app store analysis," *Requirements Engineering*, pp. 1–31, 2022.

[32] S. Debnath, P. Spoletini, and A. Ferrari, "From ideas to expressed needs: An empirical study on the evolution of requirements during elicitation," in *29th IEEE International Requirements Engineering Conference, RE 2021, Notre Dame, IN, USA, September 20-24, 2021*, 2021, pp. 233–244.

[33] M. G. Pitts and G. J. Browne, "Improving requirements elicitation: an empirical investigation of procedural prompts," *Information systems journal*, vol. 17, no. 1, pp. 89–110, 2007.

[34] M. Bano, D. Zowghi, A. Ferrari, P. Spoletini, and B. Donati, "Teaching requirements elicitation interviews: An empirical study of learning from mistakes," *Requirements Engineering*, vol. 24, no. 3, pp. 259–289, 2019.

[35] A. Ferrari, P. Spoletini, M. Bano, and D. Zowghi, "SaPeer and ReverseSaPeer: Teaching requirements elicitation interviews with role-playing and role reversal," *Requirements Engineering*, vol. 25, no. 4, pp. 417–438, 2020.

[36] F. Dalpiaz. Requirements data sets (user stories). Mendeley Data, V1. [Online]. Available: http://dx.doi.org/10.17632/7zbk8zsd8y.1

[37] M. Pitts and G. Browne, "Stopping behavior of systems analysts during information requirements elicitation," *Journal of Management Information Systems*, vol. 21, pp. 203–226, 06 2004.

[38] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qual. Res. Psychol.*, vol. 3, no. 2, pp. 77–101, 2006.

[39] J. Saldaña, *The Coding Manual for Qualitative Researchers*. SAGE, 2021.

[40] A. Ferrari and P. Spoletini, "Strategies, Benefits and Challenges App Store- inspired Requirements Elicitation - Supplementary Material," Sep. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7578078

[41] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[42] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. M. Vollmer, and S. Wagner, "Software engineering for ai-based systems: a survey," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 2, pp. 1–59, 2022.

[43] O. Karras, K. Schneider, and S. A. Fricker, "Representing software project vision by means of video: a quality model for vision videos," *journal of Systems and Software*, vol. 162, p. 110479, 2020.

[44] I. Hadar, P. Soffer, and K. Kenzi, "The role of domain knowledge in requirements elicitation via interviews: An exploratory study," *Requirements Engineering*, vol. 19, no. 2, p. 143–159, 2014.

[45] A. Niknafs and D. M. Berry, "The impact of domain knowledge on the effectiveness of requirements engineering activities," *Empirical Software Engineering*, vol. 22, no. 1, pp. 80–133, 2017. [Online]. Available: https://doi.org/10.1007/s10664-015-9416-2

[46] V. Gervasi, R. Gacitua, M. Rouncefield, P. Sawyer, L. Kof, L. Ma, P. Piwek, A. De Roeck, A. Willis, H. Yang *et al.*, "Unpacking tacit knowledge for requirements engineering," in *Managing Requirements Knowledge*. Berlin Heidelberg: Springer, 2013, pp. 23–47.

[47] A. Ferrari, P. Spoletini, and S. Gnesi, "Ambiguity and tacit knowledge in requirements elicitation interviews," *Requirements Engineering*, vol. 21, no. 3, pp. 333–355, 2016.

[48] I. J. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger, and A. E. Hassan, "A large-scale empirical study on software reuse in mobile apps," *IEEE software*, vol. 31, no. 2, pp. 78–86, 2013.

[49] Q. Chen, C. Chen, S. Hassan, Z. Xing, X. Xia, and A. E. Hassan, "How should i improve the ui of my app? a study of user reviews of popular apps in the google play," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 3, pp. 1–38, 2021.

[50] J. E. Reyes Arias, K. Kurtzhall, D. Pham, M. W. Mkaouer, and Y. N. Elglaly, "Accessibility feedback in mobile application reviews: A dataset of reviews and accessibility guidelines," in *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 2022, pp. 1–7.

[51] A. Sutcliffe, P. Sawyer, W. Liu, and N. Bencomo, "Investigating the potential impact of values on requirements and software engineering," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 2021, pp. 39–47.

[52] M. Zanker, L. Rook, and D. Jannach, "Measuring the impact of online personalisation: Past, present and future," *International Journal of Human-Computer Studies*, vol. 131, pp. 160–168, 2019.

[53] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang, "The app sampling problem for app store mining," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 123–133.

[54] S. L. Lim, P. J. Bentley, N. Kanakam, F. Ishikawa, and S. Honiden, "Investigating country differences in mobile app user behavior and challenges for software engineering," *IEEE Transactions on Software Engineering*, vol. 41, no. 1, pp. 40–64, 2014.

[55] F. Ebrahimi, M. Tushev, and A. Mahmoud, "Classifying mobile applications using word embeddings," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 2, pp. 1–30, 2021.

[56] L. Li, T. F. Bissyandé, and J. Klein, "Rebooting research on detecting repackaged android apps: Literature review and benchmark," *IEEE Transactions on Software Engineering*, vol. 47, no. 4, pp. 676–693, 2019.

[57] C. Gao, J. Zeng, D. Lo, X. Xia, I. King, and M. R. Lyu, "Understanding in-app advertising issues based on large scale app review analysis," *Information and Software Technology*, vol. 142, p. 106741, 2022.

[58] M. Tavakoli, L. Zhao, A. Heydari, and G. Nenadić, "Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools," *Expert Systems with Applications*, vol. 113, pp. 186–199, 2018.

[59] D. Martens and W. Maalej, "Towards understanding and detecting fake reviews in app stores," *Empirical Software Engineering*, vol. 24, no. 6, pp. 3316–3355, 2019.

[60] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in *2014 IEEE 22nd international requirements engineering conference (RE)*. Ieee, 2014, pp. 153–162.

[61] S. McIlroy, N. Ali, H. Khalid, and A. E Hassan, "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1067–1106, 2016.

[62] Z. Kurtanović and W. Maalej, "On user rationale in software engineering," *Requirements Engineering*, vol. 23, no. 3, pp. 357–379, 2018.

[63] R. Jabangwe, H. Edison, and A. N. Duc, "Software engineering process models for mobile app development: A systematic literature review," *Journal of Systems and Software*, vol. 145, pp. 98–111, 2018.

[64] R. Britto, D. Smite, L.-O. Damm, and J. Börstler, "Evaluating and strategizing the onboarding of software developers in large-scale globally distributed projects," *Journal of Systems and Software*, vol. 169, p. 110699, 2020.

[65] R. Hoda, "Socio-technical grounded theory for software engineering," *IEEE Transactions on Software Engineering*, 2021.

[66] D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka, and M. Oivo, "Empirical software engineering experts on the use of students and professionals in experiments," *Empirical Software Engineering*, vol. 23, no. 1, pp. 452–489, 2018.

[67] R. Wieringa and M. Daneva, "Six strategies for generalizing software engineering theories," *Science of computer programming*, vol. 101, pp. 136–152, 2015.