Feature Lifecycles as They Spread, Migrate, Remain, and Die in App Stores

Federica Sarro, Afnan A. Al-Subaihin, Mark Harman, Yue Jia, William Martin, and Yuanyuan Zhang
CREST, Department of Computer Science, University College London, London, UK

***Abstract*—We introduce a theoretical characterisation of feature lifecycles in app stores, to help app developers to identify trends and to find undiscovered requirements. To illustrate and motivate app feature lifecycle analysis, we use our theory to empirically analyse the migratory and non-migratory behaviours of 4,053 non-free features from two App Stores (Samsung and BlackBerry). The results reveal that, in both stores, intransitive features (those that neither migrate nor die out) exhibit significantly different behaviours with regard to important properties, such as their price. Further correlation analysis also highlights differences between trends relating price, rating, and popularity. Our results indicate that feature lifecycle analysis can yield insights that may also help developers to understand feature behaviours and attribute relationships.**

I. Introduction

Requirements elicitation is crucial to the development of any software. Mobile apps are not an exception; however in this environment there may be no formal requirements elicitation process, yet application distribution platforms (or app stores) offer new avenues to gather software requirements [1][2][3][4][5][6]. The App Store marketplace can be thought of as a highly user-participatory cyclic development model that partly involves "requirements for the masses; requirements from the masses". In this adaptive development space, users express their needs and desires by voting apps up and down and contributing product reviews. Users also tacitly express support for a feature by downloading apps that offer it. Developers may observe this behaviour and respond accordingly by adding popular features, where appropriate, to their own products. In this way, the developers triage the perceived desires of their users and make strategic decisions as to which features to adopt [7].

Capturing user reactions helps developers to select and prioritise feature inclusions in the next releases [4]. Performing a market-wide analysis to identify trends across the entire app marketplace can also allow developers to find undiscovered requirements [8].

Therefore, in this paper we study the lifecycle and migration of app features across product categories to gain insights into this non-traditional world of requirements elicitation. We introduce a simple set-theoretic formal characterisation of the migratory behaviours of features through app stores (some spread, some remain, some relocate, and some die out) and use it to empirically investigate how different features behave in the existing app stores. We are interested in the relationship between the migratory behaviour of these features, their value to developers (price), and the customers' reactions to them (rating and popularity). For example, developers may ask *Which migratory behaviours carry monetary value?*, and *Which migratory behaviours involve more popular features?*.

To this end we analysed features claimed for apps by their developers, which we extracted using natural language processing from the app descriptions available in the existing app stores [9]. Previous studies have shown that it is possible to extract features from product descriptions available on-line [10][11][12] and that the use of text mining is growing as a new form of requirements elicitation [2][13].

To carry out the empirical study, we mined a total of 4,053 non-free features from the Samsung Apps and BlackBerry World app stores, and automatically classified them according to our migratory behaviour theory. We then measured the price, rating, and popularity (rank of downloads) of these features in terms of their averages (both mean and median) calculated over all apps that share the features [9][14], to investigate the differences and relationships between these attributes for the different migratory behaviours.

The paper has two primary contributions:

1. Theory: We introduce and formalise concepts of feature lifecycle, migration, exodus, extinction, and intransitivity using a set theoretic formalism that casts all features into a subsumption hierarchy of migratory behaviours and the relationships between them.

2. Empirical Results and Analysis: To illustrate the value of our approach, we use it to analyse the BlackBerry World and Samsung Apps stores. Our findings reveal that it is possible to classify the mined features into the different migratory categories defined by our theory, and that the features in such categories exhibit different characteristics with respect to the important attributes of price, rating, and popularity. These differences manifest themselves as statistically significant differences in the mean (and median) values between categories. We also found differences in the relationships between the three attributes, as expressed in terms of correlation analysis (both linear and rank based).

In the remainder of the paper, Section II provides background information on our approach to mining app store repositories. Section III introduces our Set-Theoretic Theory of Feature Lifecycles and Migration. Section IV explains our empirical study design, while Section V presents the results of the BlackBerry World and the Samsung Apps study. Section VI discusses threats to validity. Section VII considers related and future work, and Section VIII concludes.

978-1-4673-6905-3/15 © 2015 IEEE

RE 2015, Ottawa, ON, Canada
Research Paper

II. Background

Our approach is based on the extraction of feature claims from software product descriptions [9][10][11][12][14], which we briefly review here (details can be found elsewhere [9][14]). Our feature extraction approach consists of four phases. The first phase extracts raw data from the app store (in this case BlackBerry World and Samsung Apps, though our approach can be applied to other app stores with suitable changes to the extraction front end). The second phase parses the raw data extracted in the first phase to retrieve all the available attributes of each app relating to price, rating, and textual descriptions of the app itself. The third phase analyses app descriptions to identify the features claimed for apps by their developers. The fourth phase computes the price, rating, and popularity for the identified features.

Phase 1 uses a customised web crawler to collect raw data from the app store, from which we parse the HTML to extract the descriptions and other data (rating, price, and popularity measured in terms of the rank of downloads) in Phase 2. Phase 3 uses natural language processing to extract, from each description, the features claimed for the app by its developers. Such feature claims can be written in many ways by developers. We developed a four-step NLP algorithm to extract feature information and implemented it using the Python Natural Language Toolkit (NLTK) [15]. The first step extracts raw feature patterns, thereby identifying the 'coarse features' of apps. We locate raw feature patterns by searching for an HTML list in the description of apps. If the sentence prior to an HTML list contains at least one keyword from the set of words "include, new, latest, key, free, improved, download, option, feature", the HTML list is saved as the raw feature pattern for this app. Non-English and numerical characters are removed along with unimportant English language STOPWORDS such as {'the', 'and', 'to'}. The words that remain are transformed into 'lemma form' using the WordNetLemmatizer function from NLTK, thereby homogenising singular/plural, gerund endings, and other non-germane grammatical details. From this lemmatised, stop-word-reduced token stream, the algorithm extracts a set of 'featurelets'; a set of commonly occurring co-located words, identified using NLTK's N-gramCollocationFinder package. We use a greedy hierarchical clustering algorithm to aggregate all similar featurelets together. The algorithm initially treats each featurelet as a single cluster, and then repeatedly combines clusters that are more than 50% similar. The result is a set of feature descriptions consisting of either 2 or 3 keywords (which we call 'bitri-grams') that describe the claimed feature.

Because of the importance of the feature mining process to any kind of analysis, Finkelstein et al. [14] performed a sanity check of the feature extraction algorithm to verify whether the features extracted were meaningful to humans. As sanity check, software developers working at UCL were asked to say whether they believed that a given claimed feature represented a feature or not. To this end a questionnaire containing claimed (i.e., bitri-grams extracted by the mining technique used herein) and random features (i.e., bitri-grams created by randomly selecting words from app descriptions) was used. The results showed that developers often classify the claimed features as a feature and the random features as a non-feature (i.e., Precision=0.71, Recall=0.77).

In Phase 4 we use a set of metrics that compute the rating, price, and popularity of a feature in terms of the median value of the corresponding ratings, prices, and popularities of all apps that possess the feature [9][14]. We used the median, because app popularity is measured as an ordinal rank (called 'rank of downloads' by several app stores) and the rating is a star rating (recorded for each app as a value from 0 to 5 stars in half star increments). These two measurements are clearly ordinal scale measurements and so the median is the most suitable average computation [16]. For price, the use of median (instead of mean) for value aggregation is more questionable. We did observe ordinal pricing behaviour. For example, the app store requires developers to charge in whole dollar increments. Furthermore, prices chosen by developers tend to cluster around ten, twenty, and thirty dollar 'price points', suggesting some kind of implicit 'ordinal scale' properties. However, the scale could equally well be argued to be a ratio scale. In order to check that our choice of median aggregation did not affect the results we report here, we re-computed all results using mean to aggregate over app prices, ratings, and popularities. The findings remained as reported here, suggesting that the choice of aggregation technique is relatively unimportant for the features studied. For completeness, we provide all of our data on the accompanying website[1].

III. A Theory of App Store Feature Lifecycles and Migratory Behaviours

We are interested in features that migrate, because movement of features between categories suggests that these features have some form of transferable value beyond the category of apps in which they initially emerge in the app store ecosystem. In order to define migration, we need to describe, first, the categories in which a feature resides at a given time in a given app store database. We define this formally as follows:

Definition 1 (Category Membership)**.** If a feature f in an app store database D is a member of category C at time t then we shall write $f \in C_{D\{t\}}$. We define the set of categories, $\mathcal{C}^f_{D\{t\}}$, of which a feature f is a member at time t in D, by extension, as $\{C \mid f \in C_{D\{t\}}\}$.

There are various behaviours that could be termed 'migratory'. We start with the weakest possible notion of migration, according to which a feature migrates if it resides in at least one new category at the end of the time period considered. More formally, we define the weak migration predicate on features as follows:

[1] http://www0.cs.ucl.ac.uk/staff/F.Sarro/projects/UCLappA/home.html

Definition 2 (Weak Migration)**.** A feature f in an app store database D (weakly) migrates between time t_0 and t_1, written $\mathcal{WM}^f_{D\{t_0,t_1\}}$ if and only if $\mathcal{C}^f_{D\{t_1\}} - \mathcal{C}^f_{D\{t_0\}} \neq \emptyset$.

We use set comprehension notation, $\{t_0, t_1\}$, for the time period from t_0 to t_1 to allow our theory to be more conveniently extended to multiple time periods, though we restrict ourselves to a single period in the analysis in this paper. Our definition of migration is termed '*weak* migration' because *any* newly entered category counts as a migration, even if the feature disappears from (some or all of) the categories from which it is migrating. If a feature does not weakly migrate, written $\mathcal{NM}^f_{D\{t_0,t_1\}}$, then it does not enter any new categories over the time period considered.

We also define *strong* migration, where a feature strictly spreads from at least one category to at least one new category (and remains in all categories in which it originated). More formally, we define strong migration as follows:

Definition 3 (Strong Migration)**.** A feature f in an app store database D strongly migrates between time t_0 and t_1, written $\mathcal{SM}^f_{D\{t_0,t_1\}}$ if and only if (iff)

$$\begin{array}{l}(\mathcal{C}^f_{D\{t_0\}} - \mathcal{C}^f_{D\{t_1\}} = \emptyset) \ \wedge \\ (\mathcal{C}^f_{D\{t_0\}} \cap \mathcal{C}^f_{D\{t_1\}} \neq \emptyset) \ \wedge \\ (\mathcal{C}^f_{D\{t_1\}} - \mathcal{C}^f_{D\{t_0\}} \neq \emptyset)\end{array}$$

That is, a strongly migratory feature has no categories that it abandons ($\mathcal{C}^f_{D\{t_0\}} - \mathcal{C}^f_{D\{t_1\}} = \emptyset$), at least one category in which it remains ($\mathcal{C}^f_{D\{t_0\}} \cap \mathcal{C}^f_{D\{t_1\}} \neq \emptyset$), and at least one new category that it spreads into ($\mathcal{C}^f_{D\{t_1\}} - \mathcal{C}^f_{D\{t_0\}} \neq \emptyset$).

A feature that strongly migrates also weakly migrates, but not necessarily *vice versa*, hence the choice of terminology (strong and weak).

A specific category of weak migration, which we term 'exodus', is also worthy of definition. There are also weak and strong forms of exodus. In a weak exodus, a feature disappears from *at least one* of the categories in which it previously resided, while appearing (for the first time) in at least one new category. In a strong exodus, a feature disappears from *all* categories in which it previously resided to take up residence in at least one new category. More formally:

Definition 4 (Weak Exodus)**.** A feature f in an app store database D experiences weak exodus between time t_0 and t_1, written $\mathcal{WE}^f_{D\{t_0,t_1\}}$, iff $\mathcal{WM}^f_{D\{t_0,t_1\}} \wedge \neg(\mathcal{SM}^f_{D\{t_0,t_1\}})$.

Definition 5 (Strong Exodus)**.** A feature f in an app store database D experiences strong exodus between time t_0 and t_1, written $\mathcal{SE}^f_{D\{t_0,t_1\}}$, iff $\mathcal{WE}^f_{D\{t_0,t_1\}} \wedge (\mathcal{C}^f_{D\{t_0\}} \cap \mathcal{C}^f_{D\{t_1\}} = \emptyset)$.

Our definitions are so-construed that weak migration captures all possible migratory behaviours. It is the union of those features that strongly migrate and those that weakly exodus (which, in turn, includes those that strongly exodus).

There is a special case of strong exodus, permitted by our definitions, in which a feature appears for the first time at the end of the time period considered. That is, such a feature resides in *no* categories at the start of the time period (so $\mathcal{C}^f_{D\{t_0\}} = \emptyset$) and is in at least one new category at the end of the time period (so $\mathcal{C}^f_{D\{t_1\}} \neq \emptyset$). This situation is a special case of strong exodus, a feature's 'birth', in which it undergoes an 'exodus into the app store from nowhere'.

In our empirical analysis that follows, we do not include the 'Birth' of features, since we wish to focus on migration of existing features through the app store. However, for completeness, we define the Birth category, formally, as follows:

Definition 6 (Birth)**.** The Birth of feature f in an app store database D between time t_0 and t_1, written $\mathcal{B}^f_{D\{t_0,t_1\}}$, occurs iff $\mathcal{SE}^f_{D\{t_0,t_1\}} \wedge \mathcal{C}^f_{D\{t_0\}} = \emptyset$.

All of the migratory behaviours we describe and formalise involve some form of change in the categories in which the feature resides, except one, which we term the 'intransitive' features. An intransitive feature neither appears in any new categories nor does it disappear from any between the start and the end of the time period considered. More formally, we define intransitivity as follows:

Definition 7 (Intransitive)**.** A feature f in an app store database D is intransitive between time t_0 and t_1, written $\mathcal{I}^f_{D\{t_0,t_1\}}$ iff

$$\begin{array}{l}(\mathcal{C}^f_{D\{t_0\}} - \mathcal{C}^f_{D\{t_1\}} = \emptyset) \ \wedge \\ (\mathcal{C}^f_{D\{t_0\}} \cap \mathcal{C}^f_{D\{t_1\}} \neq \emptyset) \ \wedge \\ (\mathcal{C}^f_{D\{t_1\}} - \mathcal{C}^f_{D\{t_0\}} = \emptyset)\end{array}$$

That is, an intransitive feature has no categories that it abandons ($\mathcal{C}^f_{D\{t_0\}} - \mathcal{C}^f_{D\{t_1\}} = \emptyset$) and at least one category in which it remains ($\mathcal{C}^f_{D\{t_0\}} \cap \mathcal{C}^f_{D\{t_1\}} \neq \emptyset$) and it has no categories to which it spreads ($\mathcal{C}^f_{D\{t_1\}} - \mathcal{C}^f_{D\{t_0\}} = \emptyset$).

If a feature neither migrates, nor remains intransitive then it must be dying out (either from some or all categories) which we term 'extinction' in this paper. Once again, there is a strong and a weak form of extinction. In a weak extinction, the feature disappears from *at least one* category in which it resided and does not migrate to any new ones. In a strong extinction, a feature completely disappears; it disappears from *all* categories in which it resided and does not migrate to any new ones. More formally, we define weak and strong extinction as follows:

Definition 8 (Weak Extinction)**.** A feature f in an app store database D experiences weak extinction between time t_0 and t_1, written $\mathcal{WX}^f_{D\{t_0,t_1\}}$, iff $\mathcal{NM}^f_{D\{t_0,t_1\}} \wedge \neg(\mathcal{I}^f_{D\{t_0,t_1\}})$.

Definition 9 (Strong Extinction)**.** A feature f in an app store database D experiences strong extinction between time t_0 and t_1, written $\mathcal{SX}^f_{D\{t_0,t_1\}}$, iff $\mathcal{WX}^f_{D\{t_0,t_1\}} \wedge \mathcal{C}^f_{D\{t_1\}} = \emptyset$.

There is special case of strong extinction, in which no category contains the feature of interest, so $\mathcal{C}^f_{D\{t_0\}} = \mathcal{C}^f_{D\{t_1\}} = \emptyset$.

In this situation the feature is not in the app store at the start, nor at the end, of the time period considered: 'it is *unborn*', or equivalently we might say that 'it is *undead*'. That is, though the feature may exist *outside* the app store

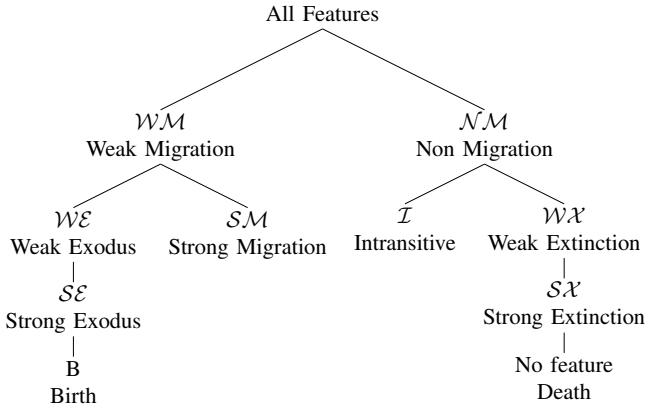

Fig. 1. **The Feature Migration Subsumption Hierarchy.**

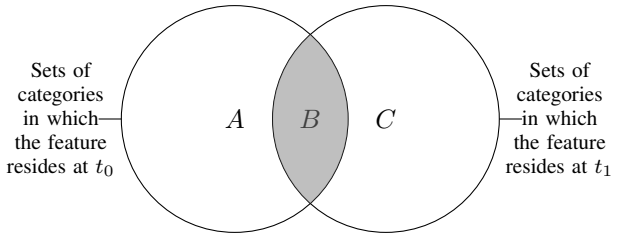

Fig. 2. **Venn diagram showing the sets of categories a feature resides in at both snapshots.** A = Categories that have the feature at t_0 but not t_1. C = Categories that have the feature at t_1 but not t_0. B = Categories that have the feature at both t_0 and t_1. Each of A, B, and C could be empty or not, so we have 8 possibilities (shown in Table I, with their corresponding definitions).

time period considered, it does not exist (is 'dead') in the app store within the period considered. Without meaning to become unreasonably supernatural, we might say that a dead feature that does, however, exist in a previous time period is 'undead', while one that does not is 'unborn'. We make this distinction in the interests of theoretical completeness; it has no further bearing on the study on which we report.

As can be seen, our definitions are loosely analogous to animal migration terminology, where features are analogous to animals and categories to geographic regions. These definitions of the different kinds of migratory behaviour form the set-theoretic subsumption relationship depicted in Figure 1. The theory is also complete; it captures all possible features in a single subsumption hierarchy of behaviours with respect to the birth, migration, and extinction of features.

To see that this theory captures all possible features and to help visualise each, consider the Venn diagram in Figure 2 and the associated mapping of all possible set configurations and their corresponding migratory definitions in Table I. This subsumption relationship allows us to speak formally and precisely about feature movement through the app store in terms of their birth, migration and death. It also precisely captures the relationships between the different kinds of feature movement that we observe in practice. We call this feature movement 'migratory behaviour'.

TABLE I
COMPLETENESS OF MIGRATORY DEFINITIONS.
Set names (A, B, and C) refer to the sets in the Venn diagram (Figure 2). Empty sets are denoted by 0, non-empty sets by 1.

Set			**Meaning**
Migratory behaviours ($\mathcal{WM}$)			
A	**B**	**C**	**Behaviour**
0	1	1	$\mathcal{SM}$
1	1	1	$\mathcal{WE}$
1	0	1	$\mathcal{SE}$, $\mathcal{WE}$
0	0	1	$\mathcal{B}$, $\mathcal{SE}$, $\mathcal{WE}$
Non-Migratory behaviours ($\mathcal{NM}$)			
A	**B**	**C**	**Behaviour**
0	1	0	$\mathcal{I}$
1	1	0	$\mathcal{WX}$
1	0	0	$\mathcal{SX}$,$\mathcal{WX}$
0	0	0	No Feature, $\mathcal{SX}$, $\mathcal{WX}$

IV. EMPIRICAL STUDY DESIGN

This section explains our empirical study design and motivates our research questions and the statistical tests we use.

A. Dataset

We extracted data about non-free apps at two time points from the BlackBerry (weeks 3 and 36 in 2011) and Samsung stores (weeks 5 and 36 in 2011). Table II presents summary data for these two 'snapshots' in each store. We were able to mine all the data available in both stores at that time, thus this study does not suffer from the App Sampling Problem [17].

The choice of time points for this first investigation of feature lifecycles is partly arbitrary, since any two time points could be used to illustrate it. However, we wanted to select two time points that were sufficiently separated that we might reasonably expect some changes, yet not so far apart that any migratory behaviour observed could not reasonably be acted upon by developers. Thus, we selected two points within the same year, but separated by at least 30 weeks. Future work will explore other time granularities to identify the smallest and largest time periods over which migration can be meaningfully observed.

In 2011, BlackBerry had 19 categories containing 18,276 and 42,625 apps in the first and second snapshots, respectively, while Samsung was a smaller store with 14 categories, and 5,206 and 12,358 apps in the first and second snapshots, respectively. We excluded from the Samsung Apps study the Brand category since it contained only 8 free apps and the Handmark category that contained different kind of apps (e.g., games, advertisement) developed by the same software company (i.e., Handmark), so it does not represent a category of apps offering similar functionalities. No categories were excluded from the BlackBerry study. Using the algorithm presented in Section II we extracted 623 and 689 unique features from the first and second Samsung snapshots, respectively, while we have mined 1,324 and 1,417 unique features from the first and second BlackBerry snapshots, respectively.

TABLE II
SUMMARY DATA FOR THE BLACKBERRY AND SAMSUNG APPS STUDIED BETWEEN TWO TIME INTERVALS.

(a) BlackBerry

Category	2011-week03 Apps	Features	Mean Price	Median Price	Mean Rating	Median Rating	Mean Rank of Downloads	Median Rank of Downloads	Min Rank of Downloads	Max Rank of Downloads	2011-week36 Apps	Features	Mean Price	Median Price	Mean Rating	Median Rating	Mean Rank of Downloads	Median Rank of Downloads	Min Rank of Downloads	Max Rank of Downloads
Business	205	82	14.81	4.99	1.86	2.00	8337	8108	803	18201	348	77	12.61	4.99	1.79	0.00	19250	18183	807	42585
Education	163	47	10.88	4.99	1.29	0.00	9526	9674	715	18320	592	80	5.66	2.99	1.38	0.00	22535	22682	1608	42673
Entertainment	456	106	4.99	2.99	1.82	2.00	7363	6622	97	18253	920	85	6.28	2.99	1.87	1.00	18593	16697	135	42628
Finance	107	77	5.42	3.99	2.10	2.00	7552	6196	168	17963	194	73	4.50	2.49	1.93	1.25	19842	16863	257	41787
Games	1633	49	3.54	2.99	1.91	2.00	7343	6432	165	18312	2618	35	2.64	1.99	2.14	2.50	16087	13730	156	42635
Health & Wellness	379	100	17.26	3.99	1.32	0.00	9045	9106	220	18077	632	87	15.76	3.99	1.57	0.00	20058	18562	258	42260
IM & Social Networking	78	67	5.13	2.99	1.78	2.00	6670	5046	19	18217	152	69	4.12	1.99	2.43	3.00	14992	11843	23	41924
Maps & Navigation	140	67	11.20	3.99	1.89	2.00	7167	5812	639	18126	284	69	12.40	9.99	1.98	2.00	18382	16066	664	42653
Music & Audio	94	69	4.51	2.99	1.75	1.50	8132	6653	142	18236	512	81	2.02	0.99	1.01	0.00	24882	27532	208	42620
News	43	38	3.36	2.99	1.33	0.50	9271	9018	1165	16151	75	42	2.31	0.99	1.75	1.00	17864	15640	1402	41957
Photo & Video	80	101	4.34	2.99	2.36	2.50	5180	3324	8	18273	423	91	2.48	1.99	1.34	0.00	22118	24710	16	42644
Productivity	334	87	8.49	4.99	2.37	3.00	6688	5692	124	18315	506	82	6.21	2.99	2.59	3.00	15023	11824	259	42643
Reference & eBooks	4356	89	5.73	2.99	0.14	0.00	13869	14491	1151	18319	11597	77	4.26	0.99	0.12	0.00	30759	31570	1181	42663
Shopping	22	61	4.31	2.99	2.20	2.50	6064	4066	676	15878	45	53	2.68	1.99	2.11	2.50	15896	11866	2585	37814
Sports & Recreation	172	35	6.31	2.99	1.73	1.00	8991	8614	216	18272	254	37	4.90	2.99	1.93	1.00	19577	16791	954	42651
Themes	4481	34	3.63	2.99	1.87	0.00	9326	9601	88	18314	11131	28	3.11	2.99	1.69	0.00	21347	21543	19	42674
Travel	450	70	6.95	5.99	0.55	0.00	11827	11986	1124	18309	769	79	4.77	2.99	0.66	0.00	25798	26477	558	42671
Utilities	715	69	5.04	2.99	2.15	2.50	6938	6021	32	18239	1377	66	4.64	2.99	2.31	2.50	16549	14267	63	42642
Weather	41	76	8.43	9.99	2.28	2.50	5364	5074	198	13629	60	66	7.39	5.99	2.40	2.50	13051	10786	311	42045
Mean	734	70	7.07	3.99	1.72	1.47	8140	7449	408	17758	1710	67	5.72	3.12	1.74	1.17	19611	18296	603	42219
Median	172	69	5.42	2.99	1.86	2.00	7552	6622	198	18239	506	73	4.64	2.99	1.87	1.00	19250	16791	259	42635

(b) Samsung

Category	2011-week05 Apps	Features	Mean Price	Median Price	Mean Rating	Median Rating	Mean Rank of Downloads	Median Rank of Downloads	Min Rank of Downloads	Max Rank of Downloads	2011-week36 Apps	Features	Mean Price	Median Price	Mean Rating	Median Rating	Mean Rank of Downloads	Median Rank of Downloads	Min Rank of Downloads	Max Rank of Downloads
E-Book/Education	34	3	12.51	3.00	1.24	0.00	3924	4497	320	5204	72	67	6.49	1.00	1.03	0.00	8502	9116	543	12353
Entertainment	186	54	2.23	1.25	2.71	3.25	3200	3435	0	5185	407	95	1.83	1.00	1.48	0.00	7677	7951	89	12313
Games	715	98	2.08	1.50	2.78	3.50	2719	2677	5	5197	1082	75	1.70	1.25	2.01	2.00	8090	9696	3	12355
Handmark	25	0	5.62	3.00	2.12	0.00	4083	4717	1659	5162	26	11	5.63	4.00	0.38	0.00	11970	12000	9238	12319
Health/Life	189	58	1.32	1.00	3.02	4.00	3210	3093	68	5198	254	52	1.28	1.00	1.88	1.00	9529	11139	23	12368
Music/Video	35	19	1.48	1.25	1.64	1.00	2165	1988	528	4300	74	35	1.39	1.25	1.92	2.00	7121	8109	138	11785
Navigation	57	59	5.01	1.25	2.88	3.50	2614	2339	45	5012	130	80	10.69	3.00	1.68	1.00	8121	9045	139	12044
News/Magazine	9	13	1.67	1.00	2.50	3.00	3149	3314	1633	4250	12	7	1.50	1.00	2.25	3.00	9121	9564	1000	12093
Productivity	76	114	4.50	1.50	2.83	3.50	3046	2924	544	5189	147	87	2.91	1.25	2.16	2.50	7891	8847	48	12357
Reference	259	59	12.65	12.00	1.45	0.00	4313	4686	141	5207	352	53	10.72	6.00	0.73	0.00	9812	10679	558	12371
Social	15	14	4.03	1.25	2.83	3.00	2302	2322	481	4784	34	19	2.46	1.25	1.90	2.00	6965	9476	97	12255
Theme	533	0	1.16	1.25	1.85	0.00	3256	3469	47	5110	4041	15	1.07	1.00	1.25	0.00	6995	7031	0	11326
Utilities	215	96	2.67	1.00	2.82	3.50	2924	3058	33	5187	468	93	1.88	1.00	1.76	0.00	8138	8519	4	12328
Mean	168	42	4.07	2.16	2.36	2.17	3147	3271	423	4999	507	53	3.81	1.85	1.57	1.04	8456	9321	914	12174
Median	67	37	2.45	1.25	2.71	3.00	3149	3093	141	5185	139	53	1.88	1.25	1.76	1.00	8121	9116	97	12319

B. Research Questions

As a starting point we ask whether each of the migratory behaviours we defined theoretically, also exists in practice. If it does, what is the distribution of features over the subsumption hierarchy of migratory behaviours. This motivates RQ1:

RQ1. Feature Migration: How do the features distribute over the different migratory behaviours in the subsumption hierarchy? If we find that our theoretical migratory behaviours exist in practice, then automatic migratory behaviour classification could be used to support developers with strategic requirements elicitation decisions. Our classification would then assume a greater practical significance if we observe important differences in the price, rating or popularity of different kinds of migratory behaviour. This motivates RQ2:

RQ2. Are there any significant differences in the price, rating, popularity of features that exhibit different migratory behaviours? We use a 2-tailed, unpaired Wilcoxon test [18] to compare the median values of the price, rating, and popularity of each of the migratory behaviours. We use the Wilcoxon test because we are investigating ordinal data and therefore need a non-parametric statistical test, with fewer assumptions about the underlying data distribution. The test is 2-tailed because there is no assumption about which median will be higher, and it is unpaired, because there are different numbers of features exhibiting each behaviour. The Null Hypothesis is that there is no difference in price (respectively rating or popularity) between categories. In common with most scientific inferential statistical testing, we set the significance level 95%, so that we have only a 0.05 probability of committing a Type 1 error (incorrectly rejecting the Null Hypothesis). This choice is justified by the fact that rejection of the Null Hypothesis would be a finding that would lead to actionable conclusions. That is, developers should start to measure and take note of migratory behaviours in app stores. Therefore, we require relatively strong evidence to support such findings. We also use the Benjamini-Hochberg correction [19] to ensure that we retain only a 0.05 probability of Type 1 error when we perform multiple statistical tests. If there is a significant difference between the price, rating or popularity of features that exhibit different migratory behaviours then we also investigate the statistical effect size of the difference using the Vargha-Delaney $\hat{A}_{12}$ metric for effect size [20]. Like the Wilcoxon test, the Vargha-

Delaney $\hat{A}_{12}$ metric makes few assumptions and is suited to ordinal data such as ours. It is also highly intuitive: for a given feature attribute (price, rating or popularity), $\hat{A}_{12}(A, B)$ is simply an estimate of the probability that the attribute value of a randomly chosen feature from migratory behaviour group A will be higher than that of migratory behaviour group B. Whereas RQ2 is concerned with differences between distributions of price, rating, and popularity, RQ3 asks about difference in the correlation between these attributes:

RQ3. Are there differences in the correlations between price, rating, and popularity within each form of migratory behaviour? To answer this question we use both Pearson [21] and Spearman statistical correlation tests [22]. Karl Pearson and Charles Spearman were, respectively 18th and 19th century professors (at University College London) who were interested in measuring correlation. While Pearson first introduced the measurement of linear correlation [21], Spearman subsequently extended Pearson's work to include rank-based correlation [22]. Each statistical measurement reports a ρ value. A ρ value of 1 indicates perfect correlation, while -1 indicates perfect inverse correlation, and 0 indicates no correlation. Values between 0 and 1 (-1) indicate the degree of correlation (inverse correlation, respectively) present. Different interpretations can be placed on the ρ values reported for linear and rank correlation. However, we may conservatively state that there is some evidence of a correlation when the absolute ρ value is greater than 0.5 and strong evidence when ρ is greater than 0.7. We also report the p value which denotes the probability that a ρ value is different to zero (no correlation). Strictly speaking, since our data is measured on an ordinal scale, findings reported using the Pearson correlation should be treated with a degree of caution. However, as observed in Section II, there are grounds for considering price to be a ratio scale measurement, so Pearson correlations may be more intuitively applied in this case (as well as Spearman rank correlations).

V. Results

RQ1. Feature Migration: According to the definitions given in Section III, we augment the Subsumption Hierarchy with the number of features found in each category (see Figure 3). As the figure shows, we found that in the BlackBerry store 1,292 features do not migrate and 32 features do and of the 623 Samsung features only 3 migrate. This is an encouraging finding for app store developers: it means that they are sufficiently few in number that they could be tracked and considered in some detail. We carefully checked whether these migrations happened because of app re-categorisation (i.e., apps migrating from one category to a different one over time). We found that some apps (i.e., 9 for Blackberry, 11 for Samsung) re-categorised over the two snapshots we considered, however none of the feature migrations observed in the present study is due to app migrations. In general, we found that features migrate to a category that has similar characteristics (e.g., the feature [find, location] migrated from Maps & Navigation to the apparently related Travel category). However, there are

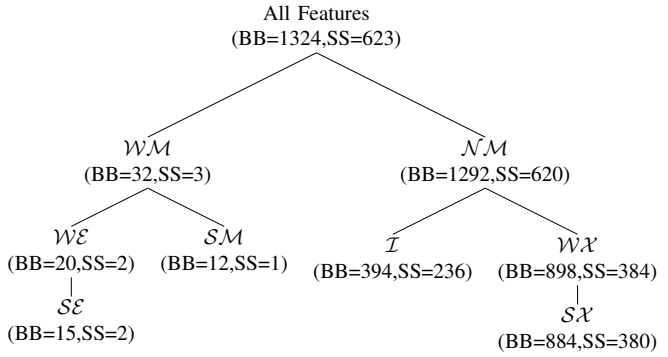

Fig. 3. **RQ1.** Observed Number of Features for each Migratory Behaviour for BlackBerry (BB) and Samsung (SS) App Stores.

also features that have clearly 'transferable value' allowing them to migrate across category boundaries (e.g., the feature [latest, news] moved from News to Sport & Recreation). In both app stores there are many features that die out (68% in BlackBerry and 62% in Samsung), while about one third (30% in BlackBerry and 38% in Samsung) are intransitive. We conjecture that intransitive features are features that are crucial to a given category. A manual analysis tends to support this view. For example, the feature [view, gps, status] is an intransitive feature of the Navigation category, while the feature [sort, track] is intransitive in the Music/Video category.

The automatic classification of app features to each of the considered migratory behaviours can support developers to make strategic decision during requirements elicitation. As an example, the early identification of the migratory features may allow them to find undiscovered requirements, while being aware of the intransitive features in a given category may support developers in identifying crucial ('must-have') requirements for their apps.

RQ2. Differences in Migratory Behaviours: Figure 4 shows the boxplots of the price, rating, and rank of downloads values of the features that have the same migratory behaviours. These results reveal some interesting differences, particularly with regard to the price of intransitive features relative to that of others. In the BlackBerry store these features appear to carry higher monetary value (a finding of great potential interest to app developers), whereas, in the Samsung store, intransitive features appear to have a lower monetary value than those which die out. In order to investigate these observations from the box plots more rigorously, we turn to an inferential statistical analysis of the differences in these sets of features, and the size of any effects observed. For the BlackBerry store, the Wilcoxon test revealed a significant difference between the price of $\mathcal{I}$ and its counterpart in the non-migratory category $\mathcal{WX}$ ($p = 0.001, \hat{A}_{12} = 0.56$ and $p = 0.007, \hat{A}_{12} = 0.55$, for mean-based and median-based feature price computation, respectively). There is also a significant difference between the price of $\mathcal{I}$ and $\mathcal{SX}$ ($p < 0.001, \hat{A}_{12} = 0.56$ and $p = 0.007, \hat{A}_{12} = 0.55$, for mean-based and median-based feature price computation, respectively). For the Samsung store, we found a statistically significant difference in the

median price values between $\mathcal{I}$ and $\mathcal{WX}$: the features that die out are higher priced than features that remain intransitive ($p = 0.048, \hat{A}_{12} = 0.55$). We make all data and analysis available on the paper's companion website[1] to support future statistical analysis and investigations by others. This data can also be used to support replication and subsequent comparisons from studies of other app stores.

In conclusion, in answer to RQ2, we find that the intransitive features behave statistically significantly differently (in both app stores) to the other features. Combined with our more qualitative finding that these features appear to be germane to the categories in which they reside, this quantitative statistical analysis suggests that intransitive features of app stores are an interesting class of feature in their own right.

RQ3. Correlations among Price, Popularity, and Rating: Table III presents the Pearson and Spearman correlations for the raw data for the BlackBerry and Samsung stores[1]. In both cases, the correlations analysis is based on scatter plots of each pair of {Price, Popularity, Rating} values for each feature.

The results reveal a strong inverse correlation between rating and rank of downloads of the migratory and non-migratory features in the BlackBerry store (see Table III(a)). This correlation has also been observed for features as a whole in previous work [9]. Moreover, we find a strong correlation between price and each of rating and popularity (reverse rank of downloads) for the strongly migratory ($\mathcal{SM}$) BlackBerry features (Pearson $\rho = -0.74$ and $\rho = 0.76$, respectively). It indicates that the more expensive a strongly migratory feature, the lower its rating and the higher its popularity. Other correlation coefficients are significant (so there is evidence that they have at least a 0.95 probability of being non-zero), but are not nearly as strong.

We also compute the median rating (respectively, rank of downloads) for all features that share a given price point. When we do this over all features, we observe a correlation between the price point and both the median rating and the median rank of downloads [14]. We also investigate whether this correlation is observed for each of the migratory behaviours. Table IV(a) reports the results. The significant correlation observations provide further evidence that there is price sensitivity for BlackBerry's migratory features (the observation that higher prices correlate to lower popularity is even stronger for them). It also provides further evidence for the potential attractiveness to developers of the intransitive features: there appears to be notably less price sensitivity to these features. That is, the inverse correlation between price and both rating and popularity is notably weaker for the intransitive features compared to all features and to the other features, which either tend to die out or migrate.

[1] We only report the correlation coefficient (ρ value) where the p value indicates that the correlation coefficient is reliable (i.e., we have evidence that it is significantly different to zero). Where the $p > 0.05$ we leave the entry blank, since there are insufficiently data points to allow us to draw reliable conclusions about correlations. A missing row indicates that there are not enough data points to conduct the test.

TABLE IV
RQ3. PRICE POINT CORRELATIONS.
Pearson and Spearman correlation values for median (R)ating and Rank of (D)ownloads for each price point. Only significant correlation values are reported[1].

(a) BlackBerry

Migratory Behaviour	Pearson		Spearman	
	PR	PD	PR	PD
$\mathcal{NM}$	-0.57	-0.66	-0.67	-0.62
$\mathcal{WX}$	-0.51	-0.62	-0.60	-0.64
$\mathcal{SX}$	-0.51	-0.62	-0.60	-0.64
$\mathcal{I}$	-0.49	-0.52	-0.51	-0.40
$\mathcal{WM}$	-0.75	0.68	-0.73	
$\mathcal{SM}$		-0.88		
$\mathcal{WE}$				
$\mathcal{SE}$		-0.75		

(b) Samsung

Migratory Behaviour	Pearson		Spearman	
	PR	PD	PR	PD
$\mathcal{NM}$	-0.53	0.70		0.78
$\mathcal{WX}$	-0.59	0.75	-0.40	0.77
$\mathcal{SX}$	-0.59	0.75	-0.40	0.77
$\mathcal{I}$	-0.44	0.64		0.74

Turning to the correlation results for Samsung (Table III(b)), we observe a strong positive correlation (Pearson $\rho = 0.70$ and $\rho = 0.71$) between price and rank of download (mean and median values) in features that face weak ($\mathcal{WX}$) and strong ($\mathcal{SX}$) extinction. This reveals that the higher price the higher rank of download (i.e., the less popular) for features that go extinct. We also observe a mild negative correlation between median price and rating for $\mathcal{WX}$ (Pearson $\rho = -0.60$) and $\mathcal{SX}$ (Pearson $\rho = -0.61$) features, respectively, i.e., the higher the prices the lower the rating (and vice versa). The correlation tests are also performed at price points for each migratory class (see Table IV(b)). The results confirm the positive correlations between price and rank of download for $\mathcal{WX}$ and $\mathcal{SX}$ features (Pearson $\rho = 0.75$, Spearman $\rho = 0.77$).

Overall, in both app stores, we therefore found interesting differences between the behaviours of features that follow different lifecycles. This provides further empirical evidence to suggest that the formal definition of these behaviours and their empirical study may yield insights. Since the correlations we studied involve relationships between price, customer rating, and popularity, it is also reasonable to assume that any such insights may prove useful to developers. Indeed, app developers are fortunate because, unlike the developers of more traditional applications, they have this information (and the analysis opportunities it offers) available for both their own apps and those of their competitors.

VI. THREATS TO VALIDITY

Threats to External Validity: Though our feature migration theory is general, our empirical results are specific to the stores considered. More work would be needed to investigate whether the findings generalise to other time periods and app stores.

Internal Validity Threat Risk Reduction: The inferential statistical values and correlations, and all the derived metrics reported in this paper were independently computed by two different authors, and cross-checked.

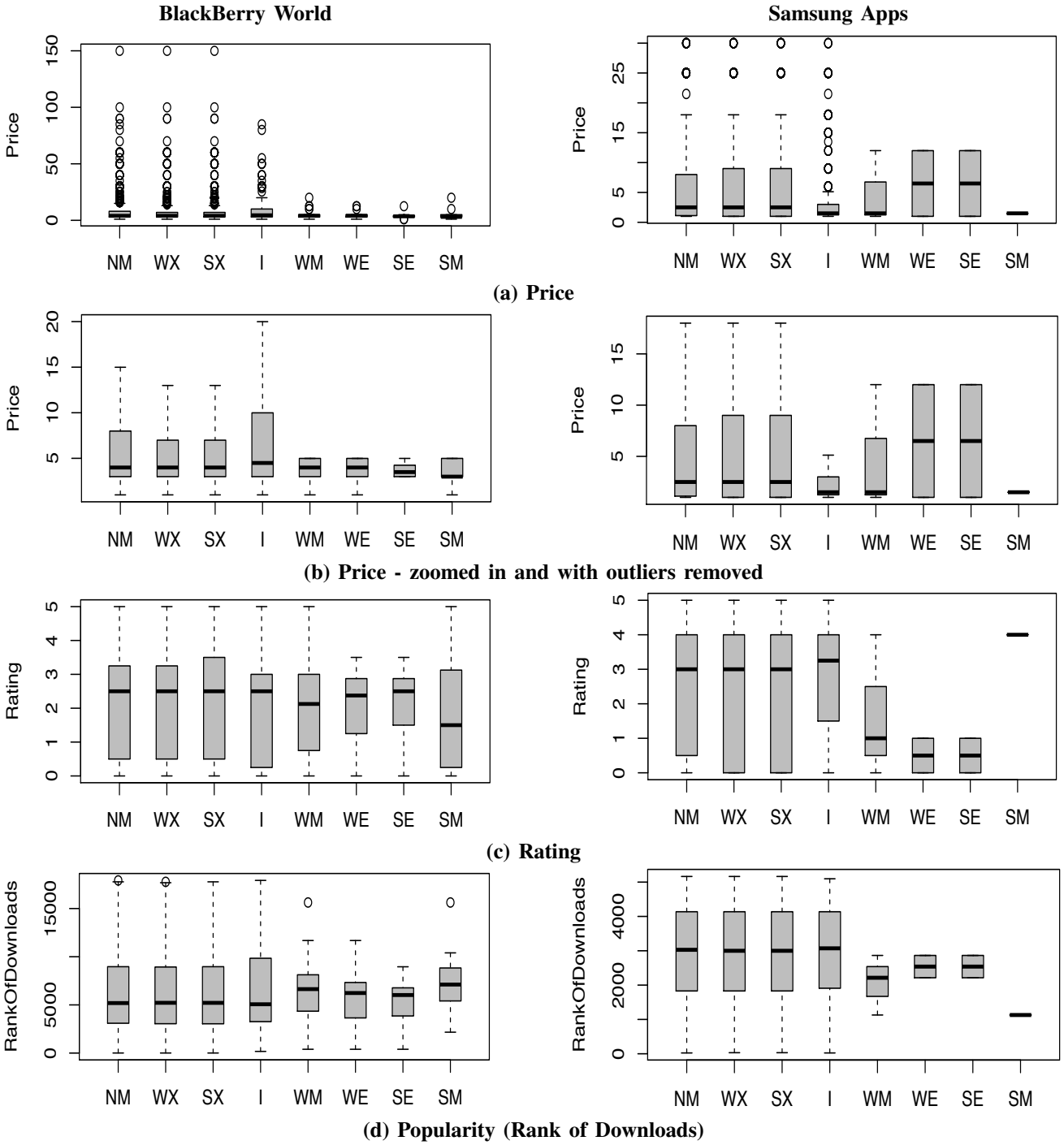

Fig. 4. **RQ2**: Boxplots of Price, Rating, and Popularity (Rank of Downloads) for all behaviours in the BlackBerry World (left side) and Samsung Apps (right side) stores. The first four boxplots of each rectangle of eight are non-migratory, while the second four are migratory. A higher Rank of Downloads indicates lower popularity. It is interesting to note that in the BlackBerry store the migratory features are lower rated, cheaper, and less popular, yet they colonise new categories. Most striking of all, the strongly migratory features which carry most transferable value, spreading through the app store, are also the cheapest, least popular, and lowest ranked features. Also, importantly for app developers, the intransitive features carry the highest monetary value; notably higher than either those features that migrate or those that die out. We observed also in the Samsung store that the migratory features are cheaper and lower rated than the non-migratory ones, however, differently from BlackBerry, they are more popular. It is also interesting to note that among the non-migratory behaviours, the intransitive features are cheaper and higher rated than those that die out, despite exhibiting the same popularity.

TABLE III
RQ3. RAW VALUE CORRELATIONS.
Pearson and Spearman correlation values for (P)rice, (R)ating, and Rank of (D)ownloads. Only significant correlation values are reported[1].

(a) BlackBerry

	Pearson						Spearman					
Migratory Behaviour	Mean PR	Median PR	Mean PD	Median PD	Mean RD	Median RD	Mean PR	Median PR	Mean PD	Median PD	Mean RD	Median RD
$\mathcal{NM}$	-0.30	0.30	0.34	0.34	-0.80	-0.81	-0.19	-0.20	0.21	0.20	-0.79	-0.77
$\mathcal{WX}$	-0.31	-0.31	0.36	0.35	-0.78	-0.79	-0.19	-0.20	0.22	0.20	-0.77	-0.75
$\mathcal{SX}$	-0.31	-0.31	0.35	0.35	-0.78	0.79	-0.18	-0.18	0.21	0.21	-0.77	-0.77
$\mathcal{I}$	-0.26	-0.27	0.30	0.32	-0.84	-0.85	-0.18	-0.17	0.19	0.20	-0.83	-0.80
$\mathcal{WM}$					-0.80	-0.74					-0.83	-0.79
$\mathcal{SM}$	-0.74		0.76	0.77	-0.82	-0.65	-0.79	-0.61	0.66	0.51	-0.85	-0.80
$\mathcal{WE}$					-0.84	-0.86					-0.84	-0.84
$\mathcal{SE}$					-0.64	-0.69					-0.76	-0.72

(b) Samsung

	Pearson						Spearman					
Migratory Behaviour	Mean PR	Median PR	Mean PD	Median PD	Mean RD	Median RD	Mean PR	Median PR	Mean PD	Median PD	Mean RD	Median RD
$\mathcal{NM}$	-0.47	-0.57	0.65	0.65	-0.44	-0.45	-0.50	-0.39	0.66	0.61	-0.46	-0.37
$\mathcal{WX}$	-0.51	-0.60	0.70	0.71	-0.44	-0.46	-0.51	-0.46	0.66	0.66	-0.46	-0.37
$\mathcal{SX}$	-0.51	-0.61	0.70	0.71	-0.44	-0.46	-0.50	-0.46	0.66	0.66	-0.46	-0.37
$\mathcal{I}$	-0.37	-0.48	0.56	0.56	-0.45	-0.44	-0.34	-0.27	0.58	0.50	-0.48	-0.37

Threats to Construct Validity: We extract feature claims reported by app store developers and cannot be sure that these necessarily correspond to *requirements* (nor even features actually implemented in the code itself, since developers do not always deliver on their claims [23]). We mitigate this threat by extracting the features from a large and varied collection of app descriptions, and clarifying that it is clearly a constraint of our method (and of most NLP-based approaches [10]). Nevertheless, we believe that developers' technical claims about their apps are inherently interesting to requirement engineers. That is, our results show that, *however* we view them, the developer claims we extract have interesting properties in real world app stores.

VII. RELATED AND FUTURE WORK

The goal of App Store Analysis is to combine technical data with non-technical data such as user and business data to understand their inter-relationships [9]. This section briefly summarises this area, its relationship to our findings, and the possible avenues for future work it opens up.

Recent studies have highlighted how the use of new platforms, such as app stores, mobile phones, and social network increases developers' opportunities to connect with users and listen to their needs [24][25][26][27]. User feedback post-release is a rich source of information for engineers involved in requirements elicitation [3][28][29][30] and many authors have focused their analysis on this aspect of app stores (e.g., [4][5][6][8][31][32][33][34][35]). Iacob and Harrison [31] report that 23.3% of the reviews they studied were found to be feature requests, further underscoring the importance of features in app store ecosystems. One natural extension of our work would be to investigate the interplay between feature migration and user requests. Pagano and Maalej [6] also found that review feedback was correlated with higher ratings and that most reviews appear very soon after a new version of an app is released. This offers the hope that developers could react to feature requests, perhaps particularly targeting likely migratory features in a timely fashion.

We extract features from the descriptions of apps uploaded to the app store by developers. Therefore, when we speak of a 'feature', we are speaking about a claimed feature; a feature that the developers claim to offer in their app description. Other authors have studied other features, in various forms, which exist in the code itself and also the relationship between feature claims in descriptions and features found in apps [23][36]. For example, Gorla et al. [36] used API calls to detect aberrant or otherwise suspicious behaviour. Pandita et al. [23] compared the permissions requested by the app and the app description, thereby identifying suspicious apps. Future work could examine the way these kinds of features migrate through app stores, and whether there is a relationship between migrations of claims and migrations of code.

In order for us to capture feature movement (which we call migration), we need to consider the status of an app store at different snapshots, taken at different times during its evolution. To the best of our knowledge no previous work has considered any form of analysis over more than one 'snapshot' of the app store state. However, we believe that future work may find many other possible applications and implications for such 'longitudinal' studies of app stores over periods of time. Future work might also consider extending our feature migration theory to other kinds of software system, such as software product lines, for which the relationship between products and features is known to be important [37].

VIII. CONCLUSION

We have introduced a theory of feature life cycles and empirically investigated the migratory behaviour of 4,053 non-free features mined from two App Stores (Samsung and BlackBerry). The results showed that the classification of app features according to our migratory behaviour theory can support developers to track trends and to identify user-relevant requirements that may otherwise be missed. We found that features generally migrate to a category that has similar characteristics, however there are also a few features that migrate to apparently non-related categories. The early identification

of these features may allow developers to find undiscovered requirements. We found also evidence that, in both app stores, approximately one third of all features are intransitive; they neither migrate nor do they die out over the period studied. Being aware of which are the intransitive features in a given category may support developers in identifying crucial ('must-have') requirements for their apps. Our statistical analysis revealed also that these intransitive features have significantly different behaviours, suggesting they may denote an interesting class of features in their own right. These differences have an intrinsic interest for researchers, since they may help to better understand the lifecycles of app store features. Since the differences we observe in practice relate to commercially sensitive attributes such as price, rating, and popularity, we also believe they may be valuable to app developers.

ACKNOWLEDGEMENT

Thanks to Jane Cleland-Huang for her insightful comments on an earlier draft of this paper. The research is funded by the EPRSC CREST Platform Grant (EP/G060525) and DAASE programme grant (EP/J017515).

REFERENCES

[1] C. Alves, G. Ramalho, and A. Damasceno, "Challenges in requirements engineering for mobile games development: The meantime case study," in *IEEE International Requirements Engineering Conference*, 2007, pp. 275–280.

[2] A. Sutcliffe and P. Sawyer, "Requirements elicitation: Towards the unknown unknowns," in *IEEE International Requirements Engineering Conference*, 2013, pp. 92–104.

[3] D. Pagano and B. Brügge, "User involvement in software evolution practice: A case study," in *International Conference on Software Engineering*, 2013, pp. 953–962.

[4] L. Galvis Carreno and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," in *International Conference on Software Engineering*, 2013, pp. 582–591.

[5] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *International Conference on Requirements Engineering*, 2014, pp. 153–162.

[6] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *IEEE International Requirements Engineering Conference*, 2013, pp. 125–134.

[7] A. J. Ko, M. J. Lee, V. Ferrari, S. Ip, and C. Tran, "A case study of post-deployment user feedback triage," in *Int. Workshop on Cooperative and Human Aspects of Software Engineering*, 2011, pp. 1–8.

[8] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 1276–1284.

[9] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: MSR for App Stores," in *IEEE Working Conference on Mining Software Repositories*, 2012, pp. 108–111.

[10] 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 *International Conference on Software Engineering*, 2011, pp. 181–190.

[11] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans, "Feature model extraction from large collections of informal product descriptions," in *Joint Meeting on Foundations of Software Engineering*, 2013, pp. 290–300.

[12] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher, "Supporting domain analysis through mining and recommending features from online product listings," *IEEE Transactions on Software Engineering*, vol. 39, no. 12, pp. 1736–1752, 2013.

[13] A. Massey, J. Eisenstein, A. Anton, and P. Swire, "Automated text mining for requirements analysis of policy documents," in *IEEE International Requirements Engineering Conference*, 2013, pp. 4–13.

[14] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "App store analysis: Mining app stores for relationships between customer, business and technical characteristics," Tech. Rep. RN/14/10.

[15] E. Loper and S. Bird, "NLTK: The Natural Language Toolkit," in *Proc. of TeachNLP'02*, 2002, pp. 69–72.

[16] M. J. Shepperd, *Foundations of software measurement*. Prentice Hall, 1995.

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

[18] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[19] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: A practical and powerful approach to multiple testing," *Journal of the Royal statistical Society (Series B)*, vol. 57, no. 1, pp. 289–300, 1995.

[20] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *International Conference on Software Engineering*, 2011, pp. 1–10.

[21] K. Pearson, "Notes on regression and inheritance in the case of two parents," *Proc. of the Royal Society of London*, vol. 58, pp. 240–242, June 1895.

[22] C. E. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904.

[23] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "WHYPER: Towards automating risk assessment of mobile applications," in *USENIX Security Symposium*, 2013.

[24] N. Seyff, F. Graf, and N. Maiden, "Using mobile re tools to give end-users their own voice," in *International Requirements Engineering Conference*, 2010, pp. 37–46.

[25] N. Seyff, G. Ollmann, and M. Bortenschlager, "Appecho: a user-driven, in situ feedback approach for mobile platforms and applications," in *International Conference on Mobile Software Engineering and Systems*, 2014, pp. 99–108.

[26] S. L. Lim and A. Finkelstein, "Stakerare: Using social networks and collaborative filtering for large-scale requirements elicitation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 3, pp. 707–735, 2012.

[27] N. Seyff, I. Todoran, K. Caluser, L. Singer, and M. Glinz, "Using popular social network sites to support requirements elicitation, prioritization and negotiation," *J. Internet Services and Applications*, vol. 6, no. 1, p. 7, 2015.

[28] I. Morales-Ramirez, A. Perini, and R. S. S. Guizzardi, "Providing foundation for user feedback concepts by extending a communication ontology," in *International Conference on Conceptual Modeling*, 2014, pp. 305–312.

[29] M. Bano and D. Zowghi, "A systematic review on the relationship between user involvement and system success," *Information and Software Technology*, vol. 58, no. 0, pp. 148 – 169, 2015.

[30] M. Bano, "Aligning services and requirements with user feedback," in *IEEE International Requirements Engineering Conference*, 2014, pp. 473–478.

[31] C. Iacob and R. Harrison, "Retrieving and analyzing mobile app feature requests from online reviews," in *Working Conference on Mining Software Repositories*, 2013.

[32] H. Khalid, "On identifying user complaints of iOS apps," in *International Conference on Software Engineering*, 2013, pp. 1474–1476.

[33] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan, "What do mobile app users complain about?" *Software, IEEE*, vol. 32, no. 3, pp. 70–77, May 2015.

[34] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 1276–1284.

[35] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "AR-miner: Mining informative reviews for developers from mobile app marketplace," in *International Conference on Software Engineering*, 2014, pp. 767–778.

[36] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," in *International Conference on Software Engineering*, 2014, pp. 1025–1035.

[37] J. Rubin and M. Chechik, "A framework for managing cloned product variants," in *International Conference on Software Engineering*, 2013, pp. 1233–1236.