Empirical Software Engineering

Application Portability Measurement: A Systematic Literature Review

--Manuscript Draft--

Manuscript Number:	EMSE-D-16-00022
Full Title:	Application Portability Measurement: A Systematic Literature Review
Article Type:	Research Papers
Keywords:	Application portability; measurement; quality · systematic literature review

Empirical Software Engineering manuscript No.
(will be inserted by the editor)

Application Portability Measurement: A Systematic Literature Review

Hamza Ghandorh · Ali Bou Nassif · Roy Eagleson

Received: date / Accepted: date

Abstract Software developers agree that application portability is a desirable attribute for their software projects; application portability provides long-term cost-effectiveness and is essential to reach diverse and bigger markets. In smartphone environments, an understanding of how to measure application portability in terms of labor, hours, and line of code is an important for software designers. These factors help to maintain portable products and ensure their survival in the fragmented smartphone environment and rapidly changing user needs. Application portability allows software stackholders to make decisions on whether or not to migrate current software products to targeted environments. However, an application portability measurement confronts several obstacles, such as a lack of standards and experience. In addition, a systematic study that examines the application portability measurement for smartphone/non-smartphone environments does not exist. This paper presents the results of a systematic literature review conducted to collect evidence on measuring application portability for smartphone/non smartphone environments. The evidence was gathered from selected studies and based on a set of meaningful and focused questions. Although 19,657 studies were initially retrieved, only 53 studies of these were selected for data extraction performed against the research questions. Our results suggested that each development team would find their own way to measure the portability degree of their applications. We concluded that the application portability measurement, especially for smartphone environments, has the potential for a worthwhile investigation and further research.

Keywords Application portability · measurement · smartphone · quality · systematic literature review.

1 Introduction

One of the desirable attributes to have in software[1] is portability. Over their lifetime, all types of software may need to migrate to a variety of environments due to the growing diversity of computing environments [37]. Software developers agree that application portability is a desirable attribute for their software projects due to long-term cost-effectiveness and the need to reach bigger and more diverse markets [37][21]. Concerns in application portability include maintaining functionality, while at the same time saving time and money, and leveraging an existing

H. Ghandorh
Department of Electrical and Computer Engineering
Western University
London, Ontario, Canada
E-mail: hghandor@uwo.ca

A. BouNassif
Department of Electrical and Computer Engineering
University of Sharjah
Sharjah, UAE
E-mail: anassif@sharjah.ac.ae

R. Eagleson
Department of Electrical and Computer Engineering
Western University
London, Ontario, Canada
E-mail: eagleson@uwo.ca

[1] Both terms -‘software’- and -‘application’- will be used interchangeably throughout this paper.

effort in the deployment of software design in new ways [37]. Application portability is important for enduring software products for users and developers. Users could utilize their portable applications not only in different environments but also in totally new system versions. Developers will not waste their resources (efforts and time) on redevelopment of their products in order to target new or different environments with their portable applications [37].

With viral usage of smartphone applications and highly fragmented markets that have an array of different models, the development of dedicated applications for each platform is not a trivial task for software engineers when considering labor and maintenance costs [37]. In addition, current programming methodologies are proven to be effective for desktop software products, and these methodologies are applied to smartphone software products. However, lower performance and fewer requirements appear on these software products, which make them undesirable [51]. This phenomenon is amplified with respect to two perspectives. First, there exists a lack of systematic and comprehensive research of application portability in most environments. Second, industry suffers from many implementation issues and difficult administrative duties [37][71]. As a consequence, maintaining portable smartphone applications when mobile devices differentiate in screen size, input/output facilities, and their graphical user interface (GUI), which usually needs to be significantly adjusted, is not a trivial matter [41]. In smartphone environments, an understanding of how to measure application portability in terms of labor, hours, and line of code is not only an important factor for software designers to cope with within the fragmented smartphone markets but also empowers development teams to improve their products with portable capabilities. However, infrastructure design fragmentation, OS fragmentation, and lack of standards and experience are a few issues that thwart an accurate application portability measurement. Moreover, an evidence-based study that reflects the reality of application portability measurement for smartphone environments could not be found in the literature.

The purpose of this study is to review the current status of measuring application portability for smartphone/non-smartphone environments using available resources to conduct a *systematic literature review* (SLR) within the 1990-2013 time period. Our goal was to summarize all relevant existing information about application portability in a thorough and unbiased manner, where this work could be a prelude to further research. It was apparent beforehand that measuring application portability for smartphone environments is a new topic and sometimes finding many direct resources was not possible. Therefore, gathering evidence in non-smartphone application portability would give us an inspiring and better experience in the field.

The remainder of this paper is organized as follows: Section 2 involves related aspects of application portability and presents some previous work of software quality attributes in smartphone environments. Section 3 describes our systematic literature review and the procedure we used. Section 4 explains the validity threats to our systematic literature review. Section 5 and Section 6 state our results and discuss its implications, respectively. Section 7 presents our suggestions for our future work.

2 Background and Related Work

Many resources have been published about application portability basics. A few examples include [5],[37], and [9]. Several empirical studies have been conducted to study application portability, especially case studies, such as [32],[33], [28], and [57]. Several terminologies have been used to address application portability. For example, [4] saw application portability as a special form of software modifiability, and [35] saw application portability as a form of software reusability.

According to [18], application portability is a "degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.". [37] describes portability as follows: "an application is portable across a class of environments to the degree that the effort required to transport and adapt it to a new environment in the class is less than the effort of redevelopment". Each application or software unit[2] has a quantifiable degree to exhibit portability to a particular environment[3] based on associated cost [32].

The primary goal of portability is to facilitate the activity of *porting* of an application from the environment in which it currently operates to a new or target environment prior to allowing reuse of the complete existing application in the new environment [37][32]. Porting is mapping $(P, E(\{l_I, ..., l_i\}, o, m)) \rightarrow \left(P', E'(\left\{l'_I, ..., l'_i\right\}, o', m'\right))$, where $\left(\{l_I, ..., l_i\} \neq \left\{l'_I, ..., l'_i\right\}\right) v\,(o \neq o')\, v\,(m \neq m')$. P is the program to be ported, P' is the resulting program after

[2] Software unit entails traditional applications to large software systems.

[3] Environment is a complete collection of external elements through which a software unit interacts with other software, such as operating systems, hardware, etc.

the port, E is the original environment, E' is the target environment, l_I is the original language processor, l'_I is the targeted language processor, o is the original operating system, o' is the targeted operating system, m is the original machine, and m' is the targeted machine [23].

Porting is performed on three levels: 1) source portability, 2) binary portability, and 3) intermediate-level portability. Source portability porting is when software in its human-readable form is adapted in its source-level then recompiled for new target environments. Binary portability porting is when that software is adapted in its source-level then recompiled for its or similar execution environments with little adaptation. Intermediate-level porting involves porting software from its source to a binary environment. There are two main phases for application porting: adaptation and transportation. Adaptation is a process of modifications needed to be made on a software unit including automated code retranslation, while transportation is concerned with the physical movement of a software unit to a new environment [37]. Mooney [36] suggested that development teams always have two choices to port an existing software: software porting or software redevelopment. The former is concerned with keeping architecture with changing code to accommodate targeted environments while the latter is concerned with the possibility of building a new program from scratch in terms of reimplementing and redesigning the architecture.

Application portability is mostly acquired by ad-hoc techniques when trying to port existing products [21]. In the particular case of a software unit, application portability is achieved based on several factors including software structure, and the properties of a targeted environment in terms of hardware, programming language, and operating systems [32]. When thinking of developing a portable application, software designers need to pay attention to four dimensions: 1) code portability, 2) data portability, 3) end-user portability, and 4) developer/documentation portability [14]. Software designers need to think whether or not their code will run exactly the same in all environments, whether related data files will be transferable and run as is in all environments, whether the GUI of their program will look and feel the same in all environments, and whether or not users will be able to understand the program with their set of skills [14]. Silakov and Khoroshilov [58] reviewed current approaches to achieve application portability between different software-hardware environments. They suggested that to accomplish application portability, it is important to focus on several methods such as reusing binary files or source code and using interpretable code, API emulators virtualization, or web technologies.

[18] categorizes the portability quality attribute to three sub-characteristics as follows: adaptability, installability, and replaceability. Adaptability is the degree to which a software unit can be adapted effectively and efficiently to a target environment. Installability is the degree to which a software unit can be installed/uninstalled successfully in a target environment. Replaceability is the degree to which a software unit can be replaced with another software unit for the same purpose in the same environment.

Application portability has not only been seen as a technical issue but also as an economic issue [52]. From a technical perspective, infrastructure designs fragmentation, OS fragmentation, and lack of standards and experience are a few issues. Also, application portability is usually confused with other software qualities such as software reusability [34], where application portability is concerned with the complete reuse of existing software artifacts and application reusability is concerned with reusing some components of software artifacts to be deployed in a targeted application [35]. Further, the industry has few software developers who are equipped to write platform-independent code for portable software products[14]. Moreover, application portability may produce a small reduction in performance, storage efficiency, or extra delay of software product delivery, which is not always tolerable [37]. Sometimes, the lack of source files or documentation can stop software from being ported [37]. From a non-technical perspective, dominant software companies, e.g. Microsoft, have embraced a client-oriented paradigm when it cannot support a heterogeneous paradigm that supports a wide range of platforms with different technologies. In seeking standardization, a single paradigm is more convenient for software companies, yet the current reality forces the existence of multiple paradigms. Consequently, such situations have both users and application developers handling a new set of challenges [67]. Furthermore, application portability is not usually considered by software project managers due to the lack of suitable tools and a lack of standardization in terms of porting techniques [34].

Software metrics define the basis for quantitative measurements of key properties of software and its processes, such as development and maintenance. Two kinds of software metrics exist: process metrics, and product metrics [34]. Process metrics characterize the costs of software processes where they identify resources used - such as money, equipment, and person-days - or production measures - such as lines of debugged code. Product metrics characterize software quality where they identify size, performance, complexity, or occasionally other attributes such as maintainability [34]. Several steps are required to maintain a software measurement method including identifying measurement aims, identifying entities to be measured and characteristics of other related entities designing a measurable construct or model, and maintaining numerical assignment rules [1]. To any environment, a variety of software metrics were used which differ from one environment to another. In the context of application portability, software metrics help to characterize the costs and benefits of involving application portability, whether in the porting process of an existing software unit or in a software design process [34]. For example, measurement of application portability in web service processes requires calculating runtimes of each web service process and its configuring each web service process. The degree of application portability is identified by the number of runtimes by which each web service is supported in a targeted environment [25]. According to [5], the main goal of application portability measurement was to identify potential performance bottlenecks. Two measuring methods were used to measure application portability: extreme case testing, and synthetic benchmark. Extreme case testing is concerned with decomposing a high-level software language code to smaller programs prior to being invoked by a test program. If the smaller programs worked in the low-level software language code, it is possible that they will work in a targeted environment. Synthetic benchmark is concerned with using a benchmark or program by which its execution time would indicate the overall speed of a system. These programs will be constructed to report the available statistics of system use.

Several researchers aimed to examine software quality in smartphone environments. Mojica et al. [31] reviewed the reusability attribute in smartphone environments by empirically investigating 200,000 free Android applications. The reviewers studied inheritance and code reuse in terms of a few code classes or entire codes prior to building new applications. They found that on average 61% of the classes were reused by smartphone application developers through inheritance, libraries, or frameworks. They stated that the practice of software reuse in smartphone application is high compared to open source projects.

Nayebi et al. [40] reviewed the usability attribute in smartphone environments by studying eleven applications available at the Apple App Store. They proposed an expert-based usability evaluation framework of iOS applications with a well-defined set of criteria, where they considered the limited resources of smartphone and small screen size as the main obstacles. They suggested that to achieve success in mobile application markets, usability evaluation methods need to be tailored to each of the various mobile operating systems where they could deal with the particular characteristics of each system.

Meskini et al. [30] considered the reliability attribute in smartphone environments by evaluating the efficacy of several desktop reliability models (NHPP, Musa-Basic, Musa-Okumoto, Software Reliability Growth Models (SRGMs)) that were applied to several crash files in iOS and Windows smartphone devices. They considered the high usage of smartphones, their extraordinary functionalities, and customer trust as the main players that application developers have to consider along with the reliability attribute in smartphone environments.

Franke et al. [10] proposed a mobile software quality model concerning software quality attributes in a smartphone environment. They conducted a case study on proprietary Android software to investigate the evaluation of the software quality attribute in smartphone environments. They targeted software quality attributes that contributed directly to the end-user experience, such as adaptability, usability, efficiency, and data persistence attributes. They concluded that not only do current software quality models not satisfy mobile-specific characteristics, such as interactivity between users and smartphone devices, but they were also not designed for the mobile realm of smartphone applications.

Syer et al. [64] conducted a case study on five Android mobile applications to investigate whether there was a relationship between defect proneness of the source code files and platform dependence. They found that when mobile applications were highly dependent on certain APIs, these APIs lock these applications to a specific platform, reduce their quality, and prevent them from coping with the rapid evolution of mobile environments. They concluded that the more software is dependent on a platform, the higher the software defect appears in a source code file, especially if the file is defect prone.

Our work will review the current status of measuring application portability for smartphone/non-smartphone applications through a mapping study using available resources from 1990 until 2013. We will summarize all relevant existing information about application portability.

3 Systematic Literature Review

An SLR is a method intended to gather, evaluate, and interpret all relevant research available regarding a specific topic area or phenomena of interest, and each study included study is referred to as a *primary study*. SLR can be used to summarize existing evidence about a specific topic area, to determine any possible research gaps, to maintain a research base for further research activities, and to facilitate building new hypotheses. It serves as a thorough and unbiased tool to sum up all possible existing information about a certain topic area. It takes two shapes: a systematic mapping study and a tertiary review. The former takes every little evidence or very broad information about a certain topic area while the latter summarizes existing systematic mapping studies to answer wider research areas [24].

In order to conduct our SLR, we used the CRD-based systematic literature review procedure [3] suggested by [24]. An SLR is usually made up of several steps as follows:

Step 1: Identification of the need for the review
Step 2: Identification of research questions
Step 3: Search for studies
Step 4: Study of selection criteria
Step 5: Study of quality assessment
Step 6: Data extraction strategy
Step 7: Data synthesis strategy
Step 8: Reporting the review

Kitchenham and Charters [24] suggested that some SLR steps are more relevant for the software engineering area, and the scope of the study would determine the SLR procedure used. Therefore, they suggested that some steps could be overlooked, if applicable. This section presents the details of the steps taken during this SLR.

3.1 The Need for a Systematic Literature Review

Several studies were devoted to a long investigation of application portability. However, when new situations arose with negative consequences, it became clear that application portability had not been given sufficient attention [58]. Smartphone applications have became a recent trend in the field of software engineering where many researchers have been investigating smartphone applications from several perspectives, such as cross platform development. Nevertheless, smartphone applications lack academic efforts from a software engineering perspective [64]. In addition, application portability in smartphones is one area that suffers a lack of empirical-based efforts due to immature development in smartphones and fragments of smartphone OS, GUIs, and API dependency. Our goal with this SLR is to identify any possible gaps in current literature regarding application portability in smartphone environments and to build a solid background in order to appropriately position new research activities. Our main motivation for this work is to take a snapshot of application portability in smartphone environments from a software engineering perspective. Such a snapshot would define a potential basis for any grounded research for smartphone applications that has practical relevance.

3.2 Research Questions

Two research questions were used in order to identity the existing basis for this SLR. The research questions and their motivation are presented in Table 1:

Table 1: Research questions and its motivations

Research Question	Motivation
RQ1: What key research topics are being addressed in the area of application portability?	To identify most of the topics that captured researchers' attention in application portability
RQ2: What are the current methods to measure application portability?	To identify existing measurement models and metrics to evaluate application portability

3.3 Search for Studies

The search terms used for our SLR, developed based on Kitchenham and Charters [24] search approach, follow:

- We identified keywords as noted in relevant papers.
- We identified alternative spellings and synonyms for the search terms.
- We used boolean 'AND' and 'OR' operators to construct each search string based on our search terms.

The resulting search strings are given below:

- *(software portability OR application portability) AND (portability of software OR portability of application) AND (smartphone OR smart-phone OR smart phone OR mobile)*
- *(application portability OR adaptability OR installability OR modifiability OR reusability) AND (quality attr* OR factor* OR aspect* OR character* OR trait* OR feature*) AND (software OR program OR application software OR computer system) AND (measure* OR evaluat*) AND (smartphone OR smart-phone OR smart phone OR mobile)*

3.4 Study Selection Procedure

3.4.1 Primary search process

The primary search process was performed through manual searching in online databases for relevant journal papers and grey literature regarding application portability from 1990 until 2013. The nominated online databases or e-libraries are shown in Table 2. Table 2 includes common e-libraries for the software engineering field. ACM, IEEExplore, Science Direct, Springer, and Wiley carry many journals with Impact Factor (IF)[4], and other relevant journals and conferences were found through Google Scholar tools. It is worth noting that the order of e-libraries in the table does not represent priority or a ranking system.

Table 2: Sources to be searched

No	Source
1	ACM
2	Citeseerx
3	IEEExplore
4	Google Scholar
5	ProQuest
6	Science Direct
7	Springer
8	Wiley

3.4.2 Secondary search process

The secondary search process was performed through three sub-processes:

- Primary study references: we reviewed reference lists of each selected primary study to find any relevant materials.
- Google “Related articles” feature: we clicked on the “Related articles” feature in Google Scholar for each selected primary study.
- Google “Cited by” feature: we clicked on the “Cited by” feature in Google Scholar for each selected primary study.

3.5 Study Selection Criteria

3.5.1 Inclusion criteria

We included any primary study that fulfilled the following requirements:

- Journals with IF, related conferences, and book chapters
- Well-defined empirical studies (i.e., surveys and case studies)
- Well-defined technical reports
- Overlapping area journals and conferences

[4] Impact Factor is a metric, proposed by Thomson Reuters, that is used to indicate the average number of citations to recent articles published in a certain journal.

3.5.2 Exclusion criteria

We excluded any primary study based on the following requirements:

- Papers not subject to peer-review, unpublished work, or white papers.
- Informal empirical studies.
- Papers not concerned with application portability at the software level (i.e. programming languages, operating systems, and data and user portability were excluded).
- Papers focused on software quality models and marginalized application portability.

3.6 Data Extraction

All relevant papers were gathered and filtered according to the first author. The author used the extraction form shown in Table 3:

Table 3: Data collection form

Data item	Value	Note
Study Identifier	RQ1, RQ2 (S#)	
Paper type	Journal / Conference / Thesis / Book / Report	
Name of e-library	e.g. IEEE	
Year of publication	1990 - 2013	
Name of journal		Not applied to non journals
Which RQ was answered	RQ1 or RQ2	
Software quality attribute		
Main theme		
Secondary theme		
Environment	Smartphone / non-smartphone	
Outcomes of paper		
Motivation of paper		
Method of paper	Techniques / approaches / technologies	
Validation of paper	Analysis models/ implementation	

Another iteration was performed to revise data extraction from each primary and secondary study to check for inconsistency or typing errors.

4 Validity of the Systematic Review

Our SLR was conducted to investigate measurement of application portability by gathering all available evidence. The main threats to the validity of our review that we have identified are publication selection bias and lack of sufficient information resources.

A possible threat to our SLR is publication selection bias. We tried to reach all possible and relevant information resources, and we can see that only a few studies have been published directly in this research area. Many publications overlap other attributes with application portability, such as software reusability. Our exclusion criteria filtered the overlapping articles. We reiterated our search strings within a wider time-frame period; however, our results were basically the same as the current results.

Lack of sufficient information resources is a possible threat for our SLR. One approach to avoid the lack of sufficient resources is to adjust the search time-frame period beyond our current period. As a result, we found very outdated or non-published materials; hence, we preferred to keep our current search time-frame period.

5 Methodology

At the beginning, initial results returned 19,657 studies from all sources. Because the initial results covered many fields, such as medicine, communication engineering, etc., we focused on software engineering related papers. Then we compiled the results using our inclusion and exclusion criteria, which resulted in 213 papers. Furthermore, we noticed that many papers were indexed in more than one e-library, such as Google Scholar and IEEExplore, which led to many overlapping results. For example, many of the 77 papers reviewed that were indexed in IEEExplore could be found in Google Scholar. Therefore, we had to eliminate duplicate articles - leading to a total of 53 papers. The results are tabulated in Table 4. From Table 4, the ACM e-library returned 221 results. After we excluded non-software engineering fields and compiled the results with our inclusion and exclusion criteria, ACM returned 12 papers, but a few of its papers had been indexed in other e-libraries, such as Google Scholar. We removed duplicated papers from Google Scholar, which eliminated 7 items (one journal and six conferences; these did not include any dissertation, technical report, or book chapter).

Table 4: Sources to be searched with relevant papers

E-library	No. of Retrieved Results	Filtered Results	Reviewed Papers					
			Journal	Conference	Dissertation	Technical Report	Book Chapter	Total
ACM	221	12	1	6	0	0	0	7
Citeseerx	488	8	0	0	1	1	0	2
IEEExplore	241	77	4	15	0	0	0	19
Google Scholar	17800	47	3	3	4	1	1	12
ProQuest	268	18	0	0	2	0	0	2
Science Direct	209	19	2	1	0	0	0	3
Springer	285	29	3	0	0	0	4	7
Wiley	145	3	1	0	0	0	0	1
Total	19657	213	14	25	7	2	5	53

Many of the papers reviewed indicated 12 journals with IF values. The 12 journals and their descriptions in terms of title, where to access them, which research question was answered, and the values of its current IF are shown in Table 5. For example, the *IEEE Computer Journal*, with its IF of 1.438, was mentioned in three papers, where they answered our RQ1. Also, several journals with IF were not indexed in our nominated e-libraries, such as the *International Journal of Computer Science*, yet Google Scholar assisted us to have them included in our journal list. It is worth noting that the *Interdisciplinary Journal of Information, Knowledge and Management* had a different metric to measure the average number of citations for their journal. Moreover, the *International Journal of Computer Science and Network Security* did not post their current IF. Based on our extracted data, we answered the research questions as follows.

5.1 RQ1: What key research topics are being addressed in the area of application portability?

Our search results indicated three main topics that were discussed in application portability literature and shown in Fig 1. Our results indicated that 83% of all papers reviewed focused on development approaches and tools to achieve a high degree of application portability, and 15% of the papers talked about the relationship of application portability with other software quality attributes. Only 2% of the papers mentioned application portability testing as a recent topic for this area. In addition, when we looked into the papers that were concerned with development approaches and tools of application portability, we found that five approaches were used: 1) the redevelopment of native applications in different environments, 2) the use of cross-platform applications or cross-platform programming languages, 3) the standardization of programming languages, libraries, GUI, or operating systems, 4) the use of intermediate-level tools, and 5) the use of abstract layers or virtual machines. These results are presented in Figure 2. From our results, using abstract layers was practiced by 29% of the papers reviewed using cross-platform applications was practiced by 26% of the papers reviewed, and the redevelopment of native applications was used by 24% of the papers reviewed. The standardization of programming languages and intermediate tools represented 13% and 8% of the papers reviewed, respectively. Moreover, when we looked further into the papers that were

Table 5: Descriptions of the reviewed journals

No	Journal Name	E-Library	Reviewed Papers	Answered Research Question	Impact Factor Value
1	Software: Practice & Experience	Wiley	[2]	RQ1	1.148
2	Communications of the ACM	ACM	[11]	RQ1	2.51
3	Computer	IEEExplore	[32], [42], [55]	RQ1	1.438
4	IEEE Transaction of Software Engineering	IEEExplore	[56]	RQ1	2.292
5	Programming & Computer Software	Springer	[58]	RQ1	0.233
6	Journal of Systems & Software	Science Direct	[16]	RQ2	1.245
7	Software Quality Journal	Springer	[54]	RQ2	0.880
8	Interdisciplinary Journal of Information, Knowledge & Management	Google Scholar	[13]	RQ2	6.96
9	Mobile Networks & Application	Springer	[48]	RQ1	1.496
10	International Journal of Computer Science	Google Scholar	[17]	RQ1	0.242
11	Future Generation Computer Systems	Science Direct	[45]	RQ1	2.639

concerned with the relationship of application portability with other software quality attributes, we found six significant elements related to application portability as follows : 1) functionality, reliability, usability, efficiency, and maintainability (FRUEM), 2) performance, 3) reusability, 4) usability, 5) efficiency, and 6) maintainability, and others. FRUEM and reusability attributes represented 22.7% of the papers reviewed and performance attribute represented 13.6% of the papers. The usability, efficiency, and maintainability attributes were similarly discussed with the portability attribute, and they represented 9.1% of the papers reviewed. Other quality attributes were mentioned with application portability such as security, interoperability, and other attributes that appeared in 4.5% of the papers reviewed.

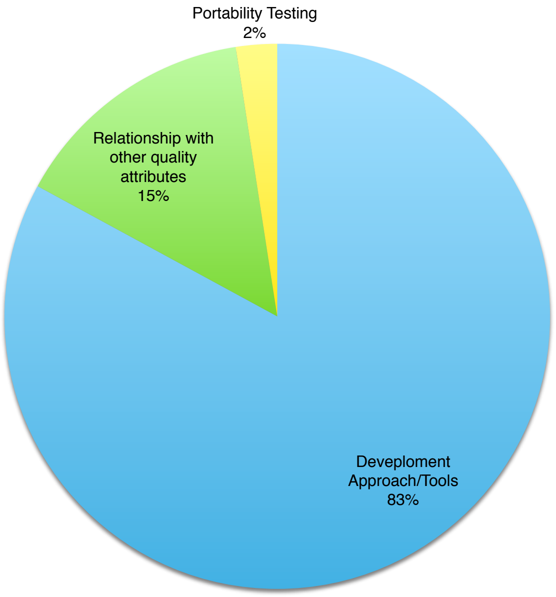


Fig. 1: Key research area in application portability

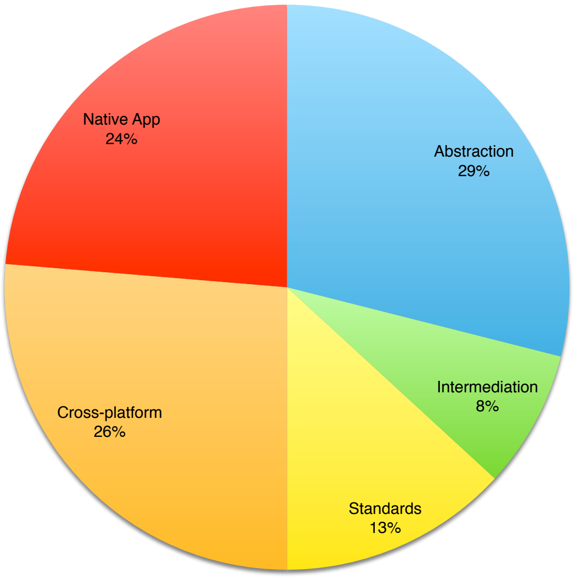


Fig. 2: Frequency of development approaches

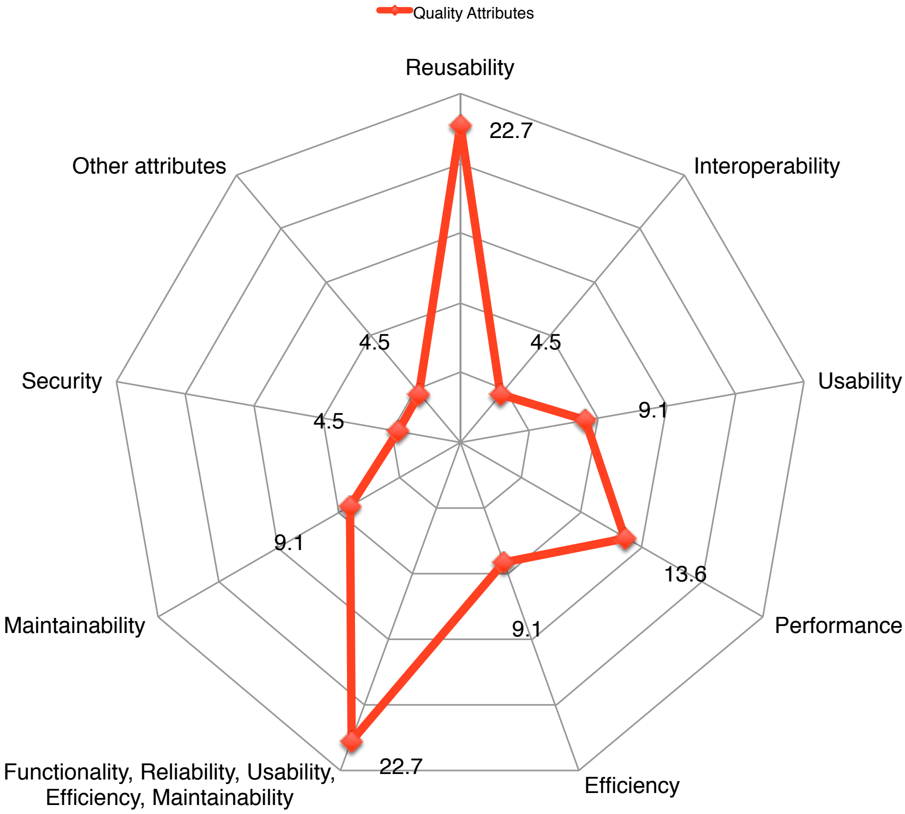


Fig. 3: Software quality attributes discussed with application portability

5.2 RQ2: What are the current methods to measure application portability?

Our results were summarized by tabulating them in Table 7. They have been organized by author name, software quality attribute, and measurement approaches. Our results indicate three perspectives to measuring application portability. First, some studies (S1, S2, S4, S10, S11) have shown cost estimation models to evaluate application portability for several environments. Second, some studies (S6, S7) considered architecture evaluation models (QADA, SAAM) to evaluate application portability. Last, other studies (S3, S5, S9) saw application portability as a factor to evaluate application reusability. Also, the S8 study paid more attention to smartphone usability evaluation even though they stated metrics used to measure application portability.

6 Discussion

The results from RQ1 showed that the research in application portability is scattered but it currently is more concerned with on development approaches/tools to maintain a high degree of application portability than other aspects. We could interpret these results as it has not been possible to have a fit-all method to achieve full application portability within different environments with the available technology and procedures. Development approaches/tools received more attention due to industry-based research interests and the lack of standards to implement complete portable applications. RQ2 results showed that there is no consensus regarding how to measure application portability. We could interpret these results as showing that various designs of hardware and software systems maintain a huge obstacle to standardizing the means to application portability measurement. As a matter of fact, understanding application portability in a variety of environments received more priority than other aspects.

The frequency of non-smartphone environments discovered through our search results is presented in Fig 4. The papers reviewed were concerned about dealing with non-specific environments (40%), desktop/server systems (26%), embedded systems (robots, control terminals, cellphone) (20%), Web services (6%), Cloud (6%), and learning objects (3%). In addition, the papers reviewed were found in eight electronic libraries, as shown in Fig 5. IEEExplore represents 35.8% as the richest source for the papers reviewed while Google Scholar ranked second representing 22.6%. ACM and Springer ranked third as a source for the papers reviewed representing 13.2%. Other electronic libraries, such as Science Direct, ProQuest, Citeseerx, and Wiley, exposed us to a variety of papers representing between 5.7%-1.9% of the papers reviewed. We gave more priority to journals with IF, and we looked at other conferences and overlapping journals to build a solid view of application portability.

It is important to notice that university dissertations and case studies were valuable sources for our SLR. Moreover, Fig 6 indicates the frequency of paper publishing on application portability as a topic in our search time-frame period. We can see that there was a relatively low and stable rate of publications from 1990 to 2002 with a small spike in 2004. A stable publications rate existed from 2006 to 2010. After 2010, a bigger spike appeared with the development of many new technologies, such as cross-platform development platforms and Cloud. We could interpret the low publications rate from 1990 until 2002 in that researchers were focused on porting applications on certain programming languages, such as FORTRAN. An increase in publication rates between 2002 and 2012 could exist due to three major events in smart-phone environments that led to new options. First, Blackberry introduced email service through its headsets in 2002. Second, Apple introduced its massive App store with smartphone applications for every purpose. Last, the Android operating system was introduced and supported by many development companies, such as Google, HTC, Intel, etc.

We noticed several interesting points from our results and we could not find reasonable justifications for them. First, we noticed that some papers considered application portability with FRUEM quality attributes to be more important than dealing with application portability individually with other software attributes such as usability or maintainability. Second, about a quarter of the papers reviewed were used to redevelop existing applications in order to be ported to other environments with the foreseen knowledge of imminent loss of time and development efforts.

Several issues worth noting arose during our SLR process. When we searched for relevant papers, sometimes no results were shown when using full search strings. We afterwards reduced our search strings to "*(software portability OR application portability) AND (smartphone OR smart-phone OR smart phone OR mobile) AND (measure* OR evaluat*)*". Such phenomena affirmed our speculation that measuring application portability for smartphones is a research gap worth investigating. Further, application portability was usually discussed or confused with other software quality attributes and this showed that we needed to tighten our exclusion criteria. We found that many articles discussed software quality attributions within quality models. We had to exclude these

Table 6: Summary of measurement approach for application portability

Study ID	Quality attribute	Measurement method
S1 [66]	Portability	Portability $= 100 * \left[1 - \frac{(\text{number of modified LOC })*\alpha}{(\text{number of total LOC})}\right]$ where LOC is lines of code and $\alpha = \frac{(\text{workload for modifying 1 LOC})}{(\text{workload for developing 1 LOC})}$
S2 [34]	Portability	$\text{DP}(su) = 1 - \left(\frac{C_{port}(su,e_2)}{C_{rdev}(req,e_2)}\right)$ where DP is a degree of portability, C_{port} is porting cost, C_{rdev} is redevelopment cost, req is requirement specification, su is software unit, and e_2 is target environment. Porting costing C_{port} is $C_{port}(su, e_2) = C_{mod}(su, e_2) + C_{ptd}(req, e_2) + C_{pdoc}(req, e_2)$, where $C_{mod}(su, e_2)$ is modeling cost, $C_{ptd}(req, e_2)$ is cost of added expenses of testing and debugging, and $C_{pdoc}(req, e_2)$ is cost of added expenses of documentation. Redevelopment costing C_{rdev} is $C_{rdev}(req, e_2) = C_{rdes}(req) + C_{rcod}(req, e_2) + C_{rtd}(req, e_2) + C_{rdoc}(req, e_2)$, where $C_{rdes}(req)$ is redesigning cost, $C_{rcod}(req, e_2)$ is recoding cost, $C_{rtd}(req, e_2)$ is retesting and redebugging cost, and $C_{rdoc}(req, e_2)$ is redocumentation cost.
S3 [47]	Reusability	Portability was used to measure reusability. There are two criteria used to judge portability: 1) Modularity, and 2) Environment independence. To measure portability, modularity uses code or number of methods metrics, and environment independence uses executable code and machines/software-depended code metrics.
S4 [16]	Portability	Four factors are considered to measure portability: 1) size of program to be ported, 2) contents of program to be ported, 3) porting impediment factors, and 4) porting cost factors. Porting impediment factors ($s_1 \sim s_{11}$) include 11 elements and they are summed as portability impediment index α_p. Portability impediment index $\alpha_p = \eta * \sum_{i=1}^{11} w_i * s_i$ where η is the portability design index that returns 0, 1, 2, or 3, and w_i is the weight assigned to each porting impediment factor s_i. Porting cost factors include two elements: 1) Human factors H_i, 2) Environmental factors E_i. Human factors ($H_1 \sim H_5$) include 5 factors and it is summed as $\alpha_H = \sum_{i=1}^{5} H_i$ Environmental factors ($E_1 \sim E_3$) include 3 elements and it is summed as $\alpha_E = \sum_{i=1}^{3} E_i$
S5 [70]	Reusability	Portability was used to measure reusability. Portability is judged by external dependency criteria, which uses two metrics: 1) Self-completeness of component's return value $SCC_r(c)$, and 2) Self-completeness of component's parameter $SCC_p(c)$. Self-completeness of component's return value $SCC_r(c)$ is $SCC_r(c) = \begin{cases} \frac{B_v(c)}{B(c)} & (B(c) > 0) \\ 1 & (otherwise) \end{cases}$, where $B_v(c)$ is number of business methods without return value in c, and $B(c)$ is number of business methods in c, c is a software component, and business methods are normal Java method that can be invoked from the outside of a software component. Self-completeness of component's parameter $SCC_p(c)$ is $SCC_p(c) = \begin{cases} \frac{B_p(c)}{B(c)} & (B(c) > 0) \\ 1 & (otherwise) \end{cases}$, where $B_p(c)$ is number of business methods without parameters in c.
S6 [29]	Portability	An evaluation method called Quality-driven Architecture Design and Quality Analysis (QADA) were used to measure portability and maintainability. QADA is a method for designing and evaluating software architecture of service-oriented systems through six steps.
S7 [22]	Portability	An evaluation method called Software Architecture Analysis Method (SAAM) was considered as a possible method to evaluate portability. SAAM concerns with describing and analyzing software architectures. SAAM is a scenario-based method that uses five steps to measure software quality attributes.
S8 [13]	Portability	Portability was judged on heterogeneity of devices criteria where it uses two metrics: 1) use of user profile, and 2) use of middleware.

Table 7: Cont, summary of measurement approach for application portability

Study ID	Quality attribute	Measurement method
S9 [54]	Reusability	Educational and technological portability metrics were used to measure reusability. An estimation model was used to measure reusability through four techniques : 1) average absolute error, 2) average relative error, 3) correlation between the real value and the estimated value, and 4) quality of the prediction.
S10 [25]	Portability	Four metrics are proposed to measure portability: 1) basic portability metric $M_{port}(p)$, 2) weighted elements portability metric $M_e(p)$, 3) activity portability metric $M_a(p)$, and 4) service communication portability metric $M_s(p)$. Basic portability metric $M_{port}(p)$ is $M_{port}(p) = 1 - \frac{C_{port}(p)}{C_{new}(p)}$, where $C_{port}(p)$ is a cost of porting a web process, $C_{new}(p)$ is a cost of redeveloping web process, and p is a web service process. Weighted elements portability metric $M_e(p)$ is $M_e(p) = 1 - \frac{C_{port}(p)}{C_{new}(p)} = 1 - \frac{\sum_{i=1}^{N_{el}} C_{el}(el_i)}{N_{el} * N_{engines}}$, where N_{el} is a total number of web process elements, $N_{engines}$ is a number of web service engines, and C_{el} is the cost for each element el_i. The cost of for each element C_{el} is $C_{el}(el_i) = \max_{j=1 \cdots N_{ta}} \left(V\left(ta_j, el_i\right) * D_{ta}\left(ta_j\right)\right)$, where $\max_{j=1 \cdots N_{ta}}$ is a max function, $V\left(ta_j, el_i\right)$ is a testing function that returns 1 or 0, and $D_{ta}\left(ta_j\right)$ is a degree of a test assertion ta_j for each web service engine. Activity portability metric $M_a(p)$ is $M_a(p) = 1 - \frac{C_{port}(p)}{C_{new}(p)} = 1 - \frac{\sum_{i=1}^{N_a} C_{el}(a_i)}{N_a * N_{engines}}$, where N_a is a total number of web service activities, and C_{el} is cost of the activities in process p. Service communication portability metric $M_s(p)$ is $M_s(p) = 1 - \frac{C_{port}(p)}{C_{new}(p)} = 1 - \frac{\sum_{i=1}^{N_s} C_{el}(s_i)}{N_s * N_{engines}}$, where s is a service interaction, and N_s is a total number of web service activities for service interaction.
S11 [1]	Portability	An estimation model was proposed to measure portability as follows: $\text{Efforts}_{new} = \text{portability\_coefficient}_{New/Ref} * \text{Effort}_{Ref} + E$, where Efforts_{new} is a total effort to redevelop software functionalities on new environment, $\text{portability\_coefficient}_{New/Ref}$ is a ratio between functional size of the portability requirement FSP of new and reference environments, Effort_{Ref} is total effort needed to develop software functionalities in reference environment, E is an error/extra value in working time. $\text{Portability\_coefficient}_{New/Ref} = \frac{(\text{FSP}_{New} - \text{FSP}_{Shared})}{\text{FSP}_{Ref}}$, where FSP_{Shared} is functional size of portability requirement of any shared software components. Functional size of portability requirement is FSP is FSP = total number of entry system interactions + total number of exit system interactions = $\sum$ System Interactions Of One functional Process

articles that only defined application portability without relevant elaboration. Finally, when we searched for resources by alternative search strings, such as "cellphone OR cell-phone OR cell phone," we did not get many results related to application portability where these strings focused on other research areas such as signal communication engineering.

7 Conclusion and Future Work

This paper presents the results of a systematic literature review on measuring application portability for smartphones/non-smartphone environments. Initially, a total of 19,657 papers were discovered, of which 213 were selected based on predefined inclusion/exclusion criteria. After eliminating duplicated papers from the e-libraries nominated, 53 papers were chosen to be reviewed. A complete description of our SLR process was presented with our results and our discussions about these results. Our results suggested that development approaches/tools for application porta-

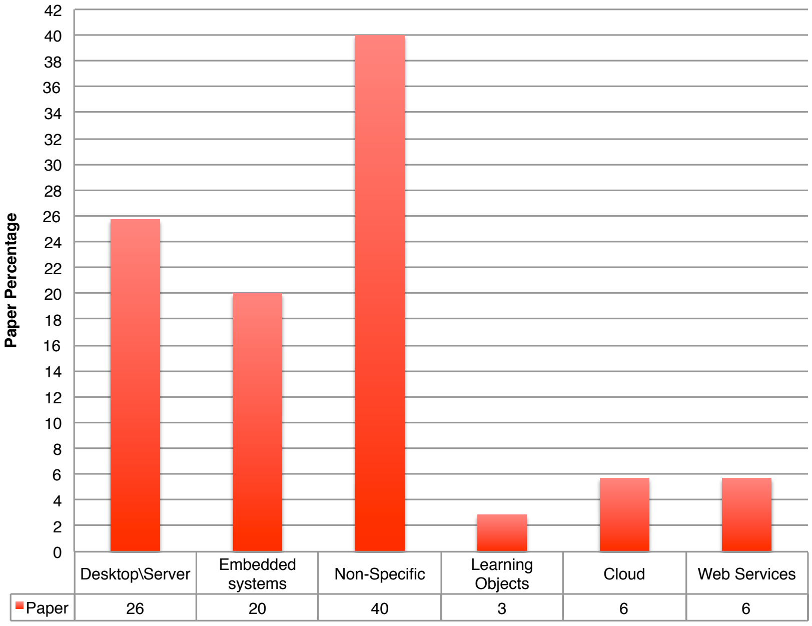


Fig. 4: Frequency of application portability discussed in non-smartphone environments

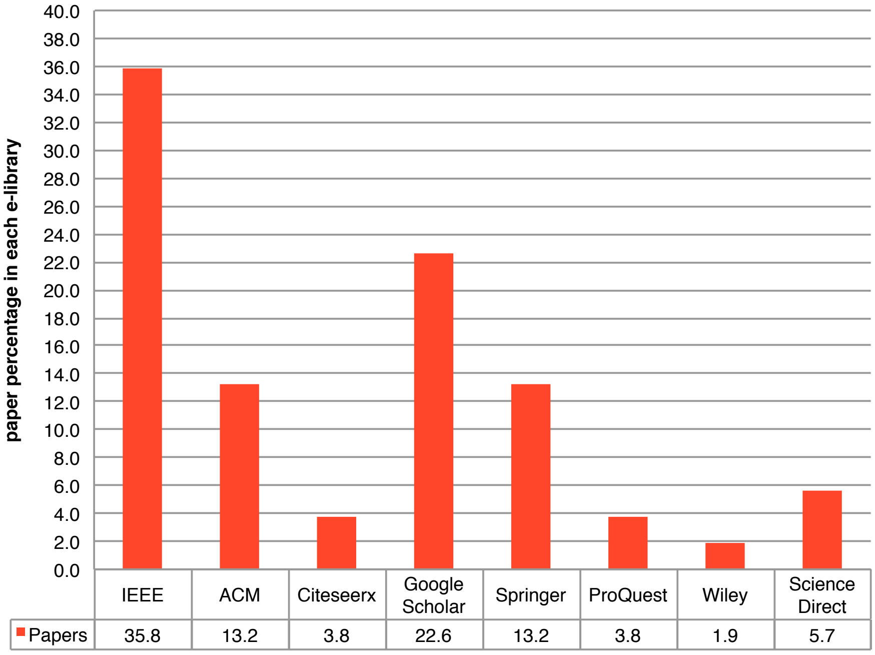


Fig. 5: Frequency of papers published in e-libraries

bility is the topic the receiving the most discussion throughout our search period. When software teams considered measuring application portability in their software projects, a customized cost estimation model was proposed with different factors and metrics. In the context of smartphone environments, application portability has no clear measuring method; while current measurement methods were designed for non-smartphone environments. This work indicates several research gaps that require further research and investigation:

- Which development approaches of application portability (redevelopment of native application, cross-platform application, web-based application, or abstract-layers application) would maintain a high degree of application portability with less impact on other software quality attributes in smartphone environments.

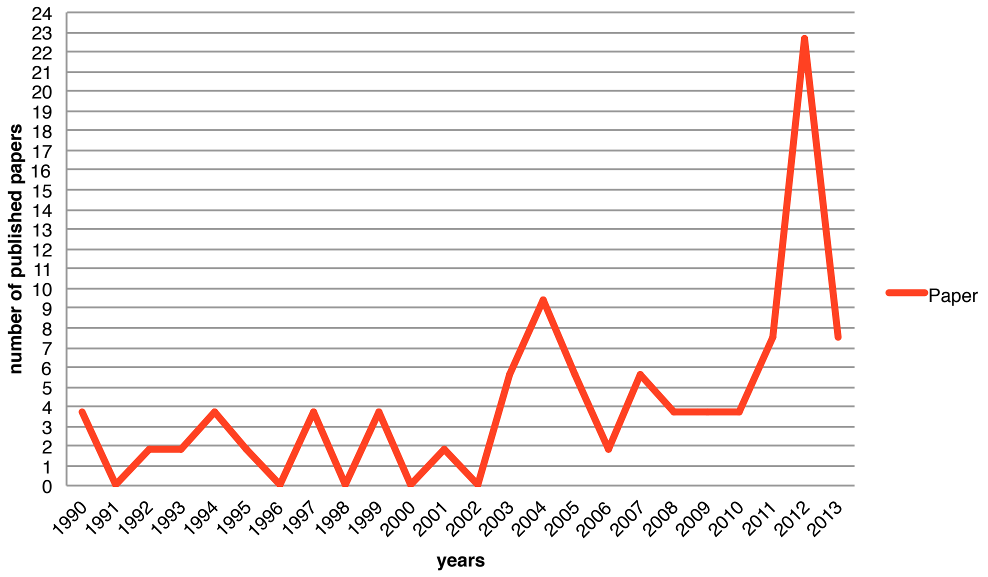


Fig. 6: Frequency of paper publishing through 1990 until 2013

- How would predication models such as Artificial Neural Network would help in cost estimates of application portability in smartphone environments ?
- How can scenario-based architecture evaluation methods help in to cost estimates of application portability in smartphone environments ?

We began this study with the intuition that we would not find any surprising or controversial results about application portability measurement for smartphone/non-smartphone environments from our search due to the novelty of the topic. However, we pursued our SLR for two reasons: there is no empirical-based evidence that concerned with this topic that reflects the reality, and it is hoped that this effort would pave a new way toward future work in smartphone research. As future work, we will investigate to what degree a scenario-based architecture evaluation method can help to cost estimate application portability in smartphone environments. We argue that software designers and researchers need to know where the research of application portability in smartphone environments is going. This is essential to be able to build effective tools for achieving a high degree of application portability by which accurate measurement is visible in the foreseen future.

Acknowledgements The research for this paper was financially supported by the Ministry of Education of Saudi Arabia, and College of Computer Science and Engineering at Taibah University, Madinah, Saudi Arabia[5].

References

1. AbuTalib, F., Giannacopoulos, D., Abran, A.: Designing a measurement method for the portability non-functional requirement. In: 23rd International Workshop on Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement, pp. 38–43 (2013)
2. Aguiar, A., Hessel, F.: Current techniques and future trends in embedded system's virtualization. Software: Practice and Experience **42**(7), 917–944 (2012). URL http://dx.doi.org/10.1002/spe.1156
3. Akers, J., Aguiar-Ibez, R., Sari, A.B.A., Beynon, S., Booth, A., Others: Systematic reviews: CRD's guidance for undertaking systematic reviews in health care. Tech. rep., Centre for Reviews and Dissemination Center, University of York (2009). http://www.york.ac.uk/inst/crd/pdf/Systematic_Reviews.pdf
4. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. SEI Series in Software Engineering. Pearson Education (2012). URL http://books.google.ca/books?id=-II73rBDXCYC
5. Brown, P.J.: Software Portability: An Advanced Course. Cambridge Computer Science Texts. Cambridge University Press (1979). URL http://books.google.ca/books?id=5Ic6AAAAIAAJ
6. Burkhart, H., Gutzwiller, S.: Software reuse and portability of parallel programs. In: Proceedings of the 29th Hawaii International Conference on System Sciences, vol. 2, pp. 289–298 (1995)

[5] Taibah University site https://www.taibahu.edu.sa/Pages/AR/Home.aspx

7. Chen, F., Zhou, H., Li, J., Liu, R., Yang, H., Li, H., Guo, H., Wang, Y.: An ontology-based approach to portable embedded system development. In: Proceedings of the 21st International Conference on Software Engineering & Knowledge Engineering, pp. 1–3. Boston, Massachusetts, USA (2009). URL https://www.dora.dmu.ac.uk/handle/2086/9322
8. Corral, L., Janes, A., Remencius, T.: Potential advantages and disadvantages of multiplatform development frameworksa vision on mobile environments. In: Procedia Computer Science, vol. 10, pp. 1202 – 1207 (2012). URL http://www.sciencedirect.com/science/article/pii/S1877050912005303
9. Dahlstrand, I.: Software portability and standards. Computers and their applications. Ellis Horwood (1984). URL http://books.google.ca/books?id=O8smAAAAMAAJ
10. Franke, D., Kowalewski, S., Weise, C.: A mobile software quality model. In: 12th International Conference on Quality Software, pp. 154–157 (2012)
11. Franz, M., Kistler, T.: Slim binaries. Commun. ACM **40**(12), 87–94 (1997). URL http://doi.acm.org/10.1145/265563.265576
12. Franz, M.S.: Code-generation on-the-fly: A key to portable software. Ph.D. thesis, Swiss Federal Institute of Technology (1994). URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.1424
13. Gafni, R.: Framework for quality metrics in mobile-wireless information systems. Interdisciplinary Journal of Information, Knowledge & Management **3**, 23 – 38 (2008). URL http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=31626699&site=ehost-live
14. Garen, K.: Software portability weighing options, making choices. The CPA Journal **Nov** (2007). URL http://www.nysscpa.org/cpajournal/2007/1107/perspectives/p10.htm
15. Gonidis, F., Paraskakis, I., Simons, A.J.H., Kourtesis, D.: Cloud application portability: An initial view. In: Proceedings of the 6th Balkan Conference in Informatics, pp. 275–282. ACM, New York, NY, USA (2013). URL http://doi.acm.org/10.1145/2490257.2490290
16. Hakuta, M., Ohminami, M.: A study of software portability evaluation. Journal of Systems and Software **38**(2), 145 – 154 (1997). URL http://www.sciencedirect.com/science/article/pii/S0164121296001185
17. Hasan, Y., Zaidi, M., Haider, N., W.U.Hasan, I.Amin: Smart phones application development using html5 and related technologies: A tradeoff between cost and quality. International Journal of Computer Science Issues **9**(3), 455 (2012). URL http://goo.gl/W9KZxD
18. ISO/IEC: ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. Tech. rep. (2010)
19. Janka, R.: A new development framework based on efficient middleware for real-time embedded heterogeneous multicomputers. In: Proceedings of IEEE Conference on Engineering of Computer-based Systems, pp. 261–268. IEEE Computer Society, Washington, DC, USA (1999). URL http://dl.acm.org/citation.cfm?id=1867775.1867810
20. Jansson, E., Jonasson, N.: Speedflirt for android, a study of porting a mobile application and its effects on user experience. Master's thesis, Ume University, Department of Applied Physics and Electronics (2011)
21. Johansson, A., Svensson, J.: Techniques for software portability in mobile development. Master's thesis (2009). URL http://goo.gl/VFCDuR
22. Jönsson, P.: Developer-oriented quality attributes and evaluation methods. In: Software quality attributes and trade-offs. Blekinge Engineering Software Qualities Research Center (2005). URL http://goo.gl/wpbVsR
23. Kaindl, H.: Portability of software. ACM Special Interest Group on Programming Languages **23**(6), 59–68 (1988). URL http://doi.acm.org/10.1145/44546.44551
24. Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Tech. Rep. Evidence-Based Software Engineering 2007-001, Keele University and Durham University Joint Report (2007). http://community.dur.ac.uk/ebse/biblio.php?id=51
25. Lenhard, J., Wirtz, G.: Measuring the portability of executable service-oriented processes. In: 17th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pp. 117–126 (2013)
26. Long, D.T., Page, D.C.: Supporting portable applications with jain slee. In: Australasian Telecommunication Networks and Applications Conference, pp. 454–459 (2007)
27. Marinho, E.H., Resende, R.F.: Quality factors in development best practices for mobile applications. In: B. Murgante, O. Gervasi, S. Misra, N. Nedjah, A. Rocha, D. Taniar, B. Apduhan (eds.) Computational Science and Its Applications 2012, *Lecture Notes in Computer Science*, vol. 7336, pp. 632–645. Springer Berlin Heidelberg (2012). URL http://dx.doi.org/10.1007/978-3-642-31128-4_47
28. Mathew, J.: Cross-platform application development on symbian. Master's thesis (2010). URL http://theseus17-kk.lib.helsinki.fi/handle/10024/14566
29. Matinlassi, M.: Evaluating the portability and maintainability of software product family architecture: terminal software case study. In: Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture 2004, pp. 295–298 (2004)
30. Meskini, S., Nassif, A.B., Capretz, L.F.: Reliability models applied to mobile applications. In: Proceedings of the 7th International Conference on Software Security and Reliability-Companion, pp. 155–162 (2013)
31. Mojica, I.J., Adams, B., Nagappan, M., Dienst, S., Berger, T., Hassan, A.E.: A large-scale empirical study on software reuse in mobile apps. IEEE Software **31**(2), 78–86 (2014)
32. Mooney, J.D.: Strategies for supporting application portability. Computer **23**(11), 59–70 (1990)
33. Mooney, J.D.: A course in software portability. In: Proceedings of the 23rd Technical Symposium on Computer Science Education, pp. 53–56. ACM, New York, NY, USA (1992). URL http://doi.acm.org/10.1145/134510.134522
34. Mooney, J.D.: Issues in the specification and measurement of software portability. http://www.csee.wvu.edu/~jdm/research/portability/reports/TR_93-6.html (1993)
35. Mooney, J.D.: Portability and reusability: Common issues and differences. In: Proceedings of the 23rd Annual Conference on Computer Science, pp. 150–156. ACM, New York, NY, USA (1995). URL http://doi.acm.org/10.1145/259526.259550
36. Mooney, J.D.: Bringing portability to the software process. Dept. of Statistics and Comp. Sci., West Virginia Univ., Morgantown WV (1997)
37. Mooney, J.D.: Developing portable software. In: R. Reis (ed.) Information Technology, *IFIP International Federation for Information Processing*, vol. 157, pp. 55–84. Springer US (2004). URL http://dx.doi.org/10.1007/1-4020-8159-6_3
38. Moore, N., Conti, A., Leeser, M., Cordes, B., King, L.S.: An extensible framework for application portability between reconfigurable supercomputing architectures (2007). URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.92.7392
39. Mosbeck, R.A., Reeve, L.C., Thedens, J.R.: Software portability in open architectures. In: 20th Conference Digital Avionics Systems, vol. 2 (2001)

40. Nayebi, F., Desharnais, J.M., Abran, A.: An expert-based framework for evaluating ios application usability. In: Joint Conference of the 23rd International Workshop on Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement, pp. 147–155 (2013)
41. Nystrom, L.: Computer Scientist Seeks to Improve Portability of Mobile Device Applications. http://goo.gl/4jAdMG (2012). Online; accessed 02-July-2014
42. Ohrt, J., Turau, V.: Cross-platform development tools for smartphone applications. Computer **45**(9), 72–79 (2012)
43. Paal, S., Kammller, R., Freisleben, B.: A cross-platform application environment for nomadic desktop computing. In: M. Weske, P. Liggesmeyer (eds.) Object-Oriented and Internet-Based Technologies, *Lecture Notes in Computer Science*, vol. 3263, pp. 185–200. Springer Berlin Heidelberg (2004). URL http://dx.doi.org/10.1007/978-3-540-30196-7_14
44. Papanikolaou, K., Mavromoustakos, S.: Critical success factors for the development of mobile learning applications. In: Proceedings of the 24th International Conference on Internet and Multimedia Systems and Applications, pp. 19–24. ACTA Press, Anaheim, CA, USA (2006). URL http://dl.acm.org/citation.cfm?id=1169167.1169171
45. Petcu, D., Macariu, G., Panica, S., Crciun, C.: Portable cloud applicationsfrom theory to practice. Future Generation Computer Systems **29**(6), 1417 – 1430 (2013). URL http://www.sciencedirect.com/science/article/pii/S0167739X12000210
46. Post, E.: Designing and implementing computer simulation models for portability and reuse. In: Centre for Advanced Computational Solutions (2003). URL http://hdl.handle.net/10182/5406
47. Poulin, J.S.: Measuring software reusability. In: Proceedings of the 3rd International Conference on Software Reuse: Advances in Software Reusability, pp. 126–138 (1994)
48. Puder, A., Antebi, O.: Cross-compiling android applications to ios and windows phone 7. Mobile Networks and Applications **18**(1), 3–21 (2013). URL http://dx.doi.org/10.1007/s11036-012-0374-2
49. Rahul, R., Tolety, S.B.: A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In: 2012 Annual IEEE India Conference, pp. 625–629 (2012)
50. Rajasekar, M.B., Kumar, M.B., Manoharan, R., Somasundaram, M., S.P.Karthikeyan: Portability of mobile applications using phonegap: A case study. In: International Conference on Software Engineering and Mobile Application Modelling and Development, pp. 1–6 (2012)
51. Rogness, N., Case, S.: An assessment of design and implementation trade-offs and their impact on mobile applications. In: Midwest Instruction and Computing Symposium (2003). URL http://www.micsymposium.org/mics_2003/Rogness.pdf
52. Rowley, D.: The business of application portability. StandardView **4**(2), 80–87 (1996). URL http://doi.acm.org/10.1145/234999.235000
53. Salonen, V.: Automatic portability testing. Master's thesis (2012). URL https://jyx.jyu.fi/dspace/handle/123456789/40043
54. Sanz-Rodriguez, J., Dodero, J.M., Sanchez-Alonso, S.: Metrics-based evaluation of learning object reusability. Software Quality Journal **19**(1), 121–140 (2011). URL http://dx.doi.org/10.1007/s11219-010-9108-5
55. Seacord, R.C.: User interface management systems and application portability. Computer **23**(10), 73–75 (1990). URL http://www.computer.org/csdl/mags/co/1990/10/rx073-abs.html
56. Shinjo, Y., Pu, C.: Achieving efficiency and portability in systems software: a case study on posix-compliant multithreaded programs. IEEE Transactions on Software Engineering **31**(9), 785–800 (2005)
57. Shixiang, G., Tao, Z.: Portability of dalvik in ios. In: 2012 International Conference on Computer Science Service System, pp. 531–537 (2012)
58. Silakov, D.V., Khoroshilov, A.V.: Ensuring portability of software. Programming and Computer Software **37**(1), 41–47 (2011). URL http://dx.doi.org/10.1134/S0361768811010051
59. Smith, G., Smith, R., Wardhani, A.: Software reuse across robotic platforms: limiting the effects of diversity. In: Proceedings. of the Australian Software Engineering Conference, pp. 252–261 (2005)
60. Smith, R., Smith, G., Wardani, A.: Software reuse in robotics: Enabling portability in the face of diversity. In: IEEE Conference on Robotics, Automation and Mechatronics, vol. 2, pp. 933–938 (2004)
61. Somishetty, S.: Porting a web application to hibernate environment. Master's thesis (2007). URL http://hdl.handle.net/10450/5381
62. Spriestersbach, A., Springer, T.: Quality attributes in mobile web application development. In: F. Bomarius, H. Iida (eds.) Product Focused Software Process Improvement, *Lecture Notes in Computer Science*, vol. 3009, pp. 120–130. Springer Berlin Heidelberg (2004). URL http://dx.doi.org/10.1007/978-3-540-24659-6_9
63. Stewart, B.M.: Portability study of android and ios. Master's thesis (2012). URL http://hdl.handle.net/10450/13060
64. Syer, M.D., Nagappan, M., Adams, B., Hassan, A.E.: Studying the relationship between source code quality and mobile platform dependence. Software Quality Journal pp. 1–24 (2014). URL http://dx.doi.org/10.1007/s11219-014-9238-2
65. Takeda, K., Wolton, I., Nicole, D.: Software portability and maintenance. In: R. Allan, M. Guest, A. Simpson, D. Henty, D. Nicole (eds.) High-Performance Computing, pp. 119–125. Springer US (1999). URL http://dx.doi.org/10.1007/978-1-4615-4873-7_13
66. Tanaka, M.: A study of portability problems and evaluation. In: Conference on Software Maintenance, pp. 90–95 (1992)
67. Tanner, P.: Software portability: Still an open issue? StandardView **4**(2), 88–93 (1996). URL http://doi.acm.org/10.1145/234999.235001
68. Teixeira, C.A., Sardinha, E.D.: Reuse in implementations of middleware for digital tv. In: Proceedings of the 18th Brazilian Symposium on Multimedia and the Web, pp. 99–102. ACM, New York, NY, USA (2012). URL http://doi.acm.org/10.1145/2382636.2382659
69. Vuletic, M., Pozzi, L., Ienne, P.: Programming transparency and portable hardware interfacing: towards general-purpose reconfigurable computing. In: Proceedings. of the 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, pp. 339–351 (2004)
70. Washizaki, H., Yamamoto, H., Fukazawa, Y.: A metrics suite for measuring reusability of software components. In: Proceedings of the 9th International Software Metrics Symposium, pp. 211–223 (2003)
71. Wasserman, A.I.: Software engineering issues for mobile application development. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, pp. 397–400. ACM, New York, NY, USA (2010). URL http://doi.acm.org/10.1145/1882362.1882443

A Appendix

A.1 Sources reviewed for RQ1

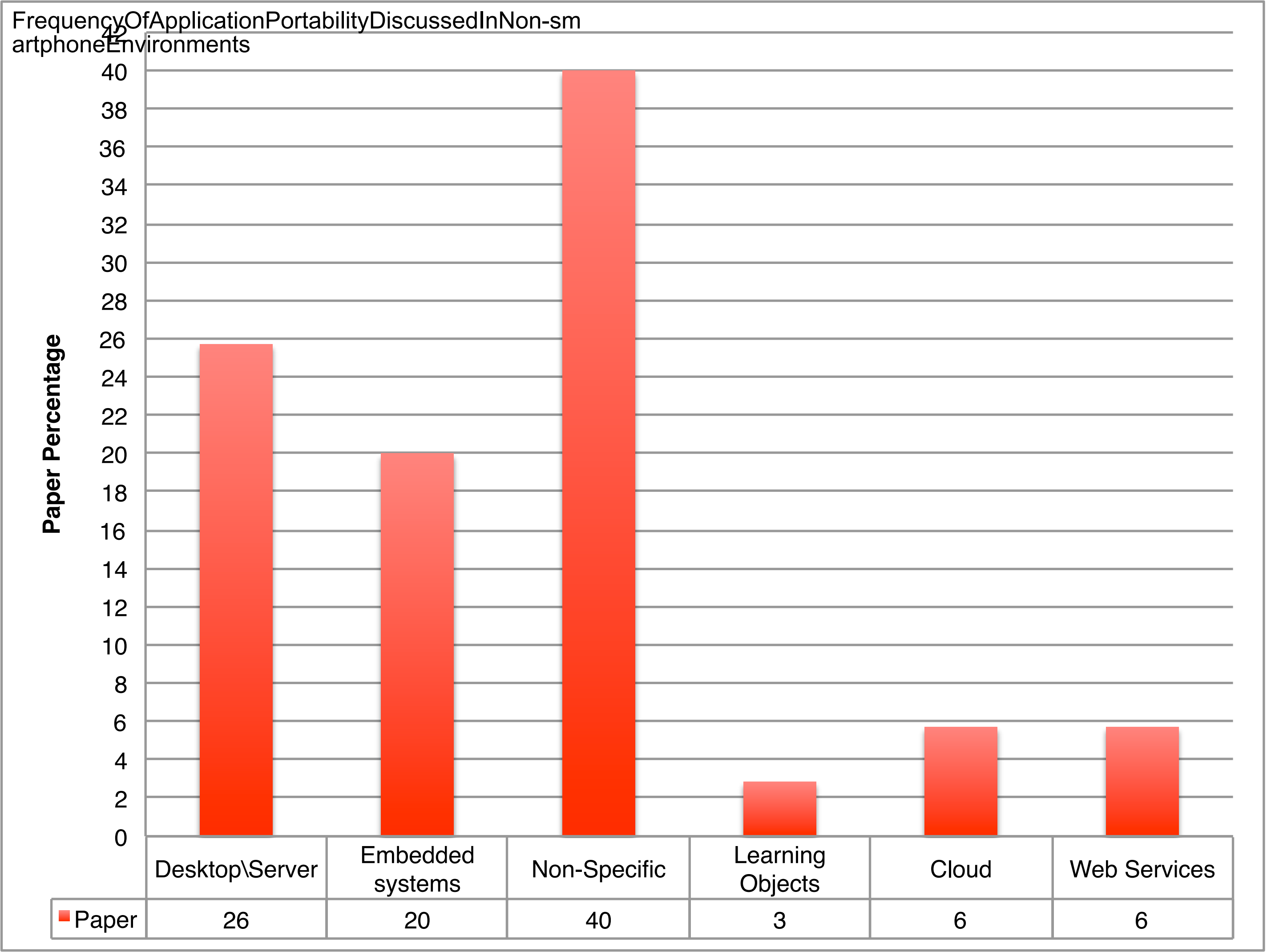
1. Aguiar, A., Hessel, F.: Current techniques and future trends in embedded system's virtualization. Software: Practice and Experience **42**(7), 917–944 (2012). URL http://dx.doi.org/10.1002/spe.1156.
2. Franz, M., Kistler, T.: Slim binaries. Commun. ACM **40**(12), 87–94 (1997). URL http://doi.acm.org/10.1145/265563.265576.
3. Hasan, Y., Zaidi, M., Haider, N., W.U.Hasan, I.Amin: Smart phones application development using html5 and related technologies: A tradeoff between cost and quality. International Journal of Computer Science Issues **9**(3), 455 (2012). URL http://goo.gl/W9KZxD.
4. Mooney, J.D.: Strategies for supporting application portability. Computer **23**(11), 59–70 (1990).
5. Puder, A., Antebi, O.: Cross-compiling android applications to ios and windows phone 7. Mobile Networks and Applications **18**(1), 3–21 (2013). URL http://dx.doi.org/10.1007/s11036-012-0374-2.
6. Ohrt, J., Turau, V.: Cross-platform development tools for smartphone applications. Computer **45**(9), 72–79 (2012).
7. Shinjo, Y., Pu, C.: Achieving efficiency and portability in systems software: a case study on posix-compliant multithreaded programs. IEEE Transactions on Software Engineering **31**(9), 785–800 (2005).
8. Silakov, D.V., Khoroshilov, A.V.: Ensuring portability of software. Programming and Computer Software **37**(1), 41–47 (2011). URL http://dx.doi.org/10.1134/S0361768811010051.
9. Seacord, R.C.: User interface management systems and application portability. Computer **23**(10), 73–75 (1990). URL http://www.computer.org/csdl/mags/co/1990/10/rx073-abs.html.
10. Petcu, D., Macariu, G., Panica, S., Crciun, C.: Portable cloud applicationsfrom theory to practice. Future Generation Computer Systems **29**(6), 1417 – 1430 (2013). URL http://www.sciencedirect.com/science/article/pii/S0167739X12000210.
11. Franz, M.S.: Code-generation on-the-fly: A key to portable software. Ph.D. thesis, Swiss Federal Institute of Technology (1994). URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.1424.
12. Jansson, E., Jonasson, N.: Speedflirt for android, a study of porting a mobile application and its effects on user experience. Master's thesis, Ume University, Department of Applied Physics and Electronics (2011).
13. Johansson, A., Svensson, J.: Techniques for software portability in mobile development. Master's thesis (2009). URL http://goo.gl/VFCDuR.
14. Mathew, J.: Cross-platform application development on symbian. Master's thesis (2010). URL http://theseus17-kk.lib.helsinki.fi/handle/10024/14566.
15. Salonen, V.: Automatic portability testing. Master's thesis (2012). URL https://jyx.jyu.fi/dspace/handle/123456789/40043.
16. Stewart, B.M.: Portability study of android and ios. Master's thesis (2012). URL http://hdl.handle.net/10450/13060.
17. Somishetty, S.: Porting a web application to hibernate environment. Master's thesis (2007). URL http://hdl.handle.net/10450/5381.
18. Rajasekar, M.B., Kumar, M.B., Manoharan, R., Somasundaram, M., S.P.Karthikeyan: Portability of mobile applications using phonegap: A case study. In: International Conference on Software Engineering and Mobile Application Modelling and Development, pp. 1–6 (2012).
19. Burkhart, H., Gutzwiller, S.: Software reuse and portability of parallel programs. In: Proceedings of the 29th Hawaii International Conference on System Sciences, vol. 2, pp. 289–298 (1995).
20. Chen, F., Zhou, H., Li, J., Liu, R., Yang, H., Li, H., Guo, H., Wang, Y.: An ontology-based approach to portable embedded system development. In: Proceedings of the 21st International Conference on Software Engineering & Knowledge Engineering, pp. 1–3. Boston, Massachusetts, USA (2009). URL https://www.dora.dmu.ac.uk/handle/2086/9322.
21. Corral, L., Janes, A., Remencius, T.: Potential advantages and disadvantages of multiplatform development frameworksa vision on mobile environments. In: Procedia Computer Science, vol. 10, pp. 1202 – 1207 (2012). URL http://www.sciencedirect.com/science/article/pii/S1877050912005303.
22. Rogness, N., Case, S.: An assessment of design and implementation trade-offs and their impact on mobile applications. In: Midwest Instruction and Computing Symposium (2003). URL http://www.micsymposium.org/mics_2003/Rogness.pdf.
23. Post, E.: Designing and implementing computer simulation models for portability and reuse. In: Centre for Advanced Computational Solutions (2003). URL http://hdl.handle.net/10182/5406.
24. Gonidis, F., Paraskakis, I., Simons, A.J.H., Kourtesis, D.: Cloud application portability: An initial view. In: Proceedings of the 6th Balkan Conference in Informatics, pp. 275–282. ACM, New York, NY, USA (2013). URL http://doi.acm.org/10.1145/2490257.2490290.
25. Janka, R.: A new development framework based on efficient middleware for real-time embedded heterogeneous multicomputers. In: Proceedings of IEEE Conference on Engineering of Computer-based Systems, pp. 261–268. IEEE Computer Society, Washington, DC, USA (1999). URL http://dl.acm.org/citation.cfm?id=1867775.1867810.
26. Long, D.T., Page, D.C.: Supporting portable applications with jain slee. In: Australasian Telecommunication Networks and Applications Conference, pp. 454–459 (2007).
27. Mosbeck, R.A., Reeve, L.C., Thedens, J.R.: Software portability in open architectures. In: 20th Conference Digital Avionics Systems, vol. 2 (2001).
28. Papanikolaou, K., Mavromoustakos, S.: Critical success factors for the development of mobile learning applications. In: Proceedings of the 24th International Conference on Internet and Multimedia Systems and Applications, pp. 19–24. ACTA Press, Anaheim, CA, USA (2006). URL http://dl.acm.org/citation.cfm?id=1169167.1169171.
29. Rahul, R., Tolety, S.B.: A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In: 2012 Annual IEEE India Conference, pp. 625–629 (2012).
30. Shixiang, G., Tao, Z.: Portability of dalvik in ios. In: 2012 International Conference on Computer Science Service System, pp. 531–537 (2012).
31. Smith, R., Smith, G., Wardani, A.: Software reuse in robotics: Enabling portability in the face of diversity. In: IEEE Conference on Robotics, Automation and Mechatronics, vol. 2, pp. 933–938 (2004).
32. Smith, G., Smith, R., Wardhani, A.: Software reuse across robotic platforms: limiting the effects of diversity. In: Proceedings. of the Australian Software Engineering Conference, pp. 252–261 (2005).
33. Teixeira, C.A., Sardinha, E.D.: Reuse in implementations of middleware for digital tv. In: Proceedings of the 18th Brazilian Symposium on Multimedia and the Web, pp. 99–102. ACM, New York, NY, USA (2012). URL http://doi.acm.org/10.1145/2382636.2382659.

34. Vuletic, M., Pozzi, L., Ienne, P.: Programming transparency and portable hardware interfacing: towards general-purpose reconfigurable computing. In: Proceedings. of the 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, pp. 339–351 (2004).
35. Wasserman, A.I.: Software engineering issues for mobile application development. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, pp. 397–400. ACM, New York, NY, USA (2010). URL http://doi.acm.org/10.1145/1882362.1882443.
36. Marinho, E.H., Resende, R.F.: Quality factors in development best practices for mobile applications. In: B. Murgante, O. Gervasi, S. Misra, N. Nedjah, A. Rocha, D. Taniar, B. Apduhan (eds.) Computational Science and Its Applications 2012, *Lecture Notes in Computer Science*, vol. 7336, pp. 632–645. Springer Berlin Heidelberg (2012). URL http://dx.doi.org/10.1007/978-3-642-31128-4_47.
37. Paal, S., Kammller, R., Freisleben, B.: A cross-platform application environment for nomadic desktop computing. In: M. Weske, P. Liggesmeyer (eds.) Object-Oriented and Internet-Based Technologies, *Lecture Notes in Computer Science*, vol. 3263, pp. 185–200. Springer Berlin Heidelberg (2004). URL http://dx.doi.org/10.1007/978-3-540-30196-7_14.
38. Spriestersbach, A., Springer, T.: Quality attributes in mobile web application development. In: F. Bomarius, H. Iida (eds.) Product Focused Software Process Improvement, *Lecture Notes in Computer Science*, vol. 3009, pp. 120–130. Springer Berlin Heidelberg (2004). URL http://dx.doi.org/10.1007/978-3-540-24659-6_9.
39. Takeda, K., Wolton, I., Nicole, D.: Software portability and maintenance. In: R. Allan, M. Guest, A. Simpson, D. Henty, D. Nicole (eds.) High-Performance Computing, pp. 119–125. Springer US (1999). URL http://dx.doi.org/10.1007/978-1-4615-4873-7_13.
40. Corral, L., Janes, A., Remencius, T.: Potential advantages and disadvantages of multiplatform development frameworksa vision on mobile environments. In: Procedia Computer Science, vol. 10, pp. 1202 – 1207 (2012). URL http://www.sciencedirect.com/science/article/pii/S1877050912005303.
41. Moore, N., Conti, A., Leeser, M., Cordes, B., King, L.S.: An extensible framework for application portability between reconfigurable supercomputing architectures (2007). URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.92.7392.

A.2 Sources reviewed for RQ2

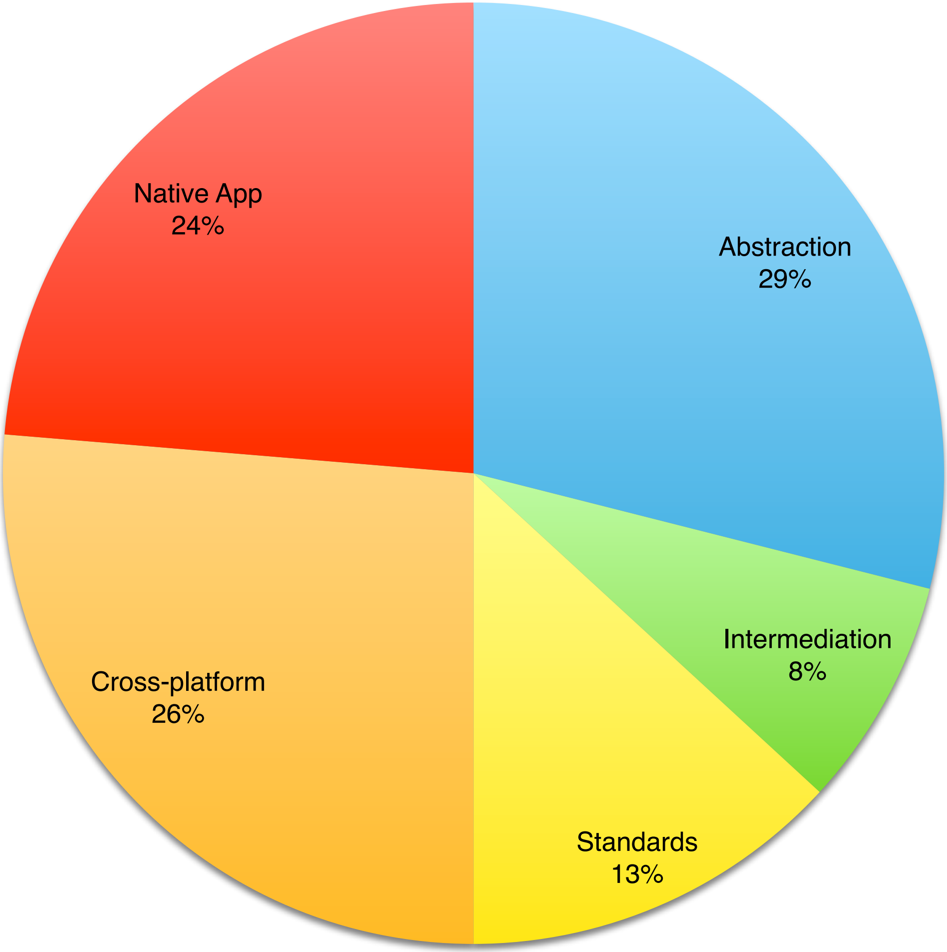
S1. Tanaka, M.: A study of portability problems and evaluation. In: Conference on Software Maintenance, pp. 90–95 (1992)
S2. Mooney, J.D.: Issues in the specification and measurement of software portability. http://www.csee.wvu.edu/~jdm/research/portability/reports/TR_93-6.html (1993)
S3. Poulin, J.S.: Measuring software reusability. In: Proceedings of the 3rd International Conference on Software Reuse: Advances in Software Reusability, pp. 126–138 (1994)
S4. Hakuta, M., Ohminami, M.: A study of software portability evaluation. Journal of Systems and Software **38**(2), 145 – 154 (1997). URL http://www.sciencedirect.com/science/article/pii/S0164121296001185
S5. Washizaki, H., Yamamoto, H., Fukazawa, Y.: A metrics suite for measuring reusability of software components. In: Proceedings of the 9th International Software Metrics Symposium, pp. 211–223 (2003)
S6. Matinlassi, M.: Evaluating the portability and maintainability of software product family architecture: terminal software case study. In: Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture 2004, pp. 295–298 (2004)
S7. Jönsson, P.: Developer-oriented quality attributes and evaluation methods. In: Software quality attributes and trade-offs. Blekinge Engineering Software Qualities Research Center (2005). URL http://goo.gl/wpbVsR
S8. Gafni, R.: Framework for quality metrics in mobile-wireless information systems. Interdisciplinary Journal of Information, Knowledge & Management **3**, 23 – 38 (2008). URL http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=31626699&site=ehost-live
S9. Sanz-Rodriguez, J., Dodero, J.M., Sanchez-Alonso, S.: Metrics-based evaluation of learning object reusability. Software Quality Journal **19**(1), 121–140 (2011). URL http://dx.doi.org/10.1007/s11219-010-9108-5
S10. Lenhard, J., Wirtz, G.: Measuring the portability of executable service-oriented processes. In: 17th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pp. 117–126 (2013)
S11. AbuTalib, F., Giannacopoulos, D., Abran, A.: Designing a measurement method for the portability non-functional requirement. In: 23rd International Workshop on Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement, pp. 38–43 (2013)

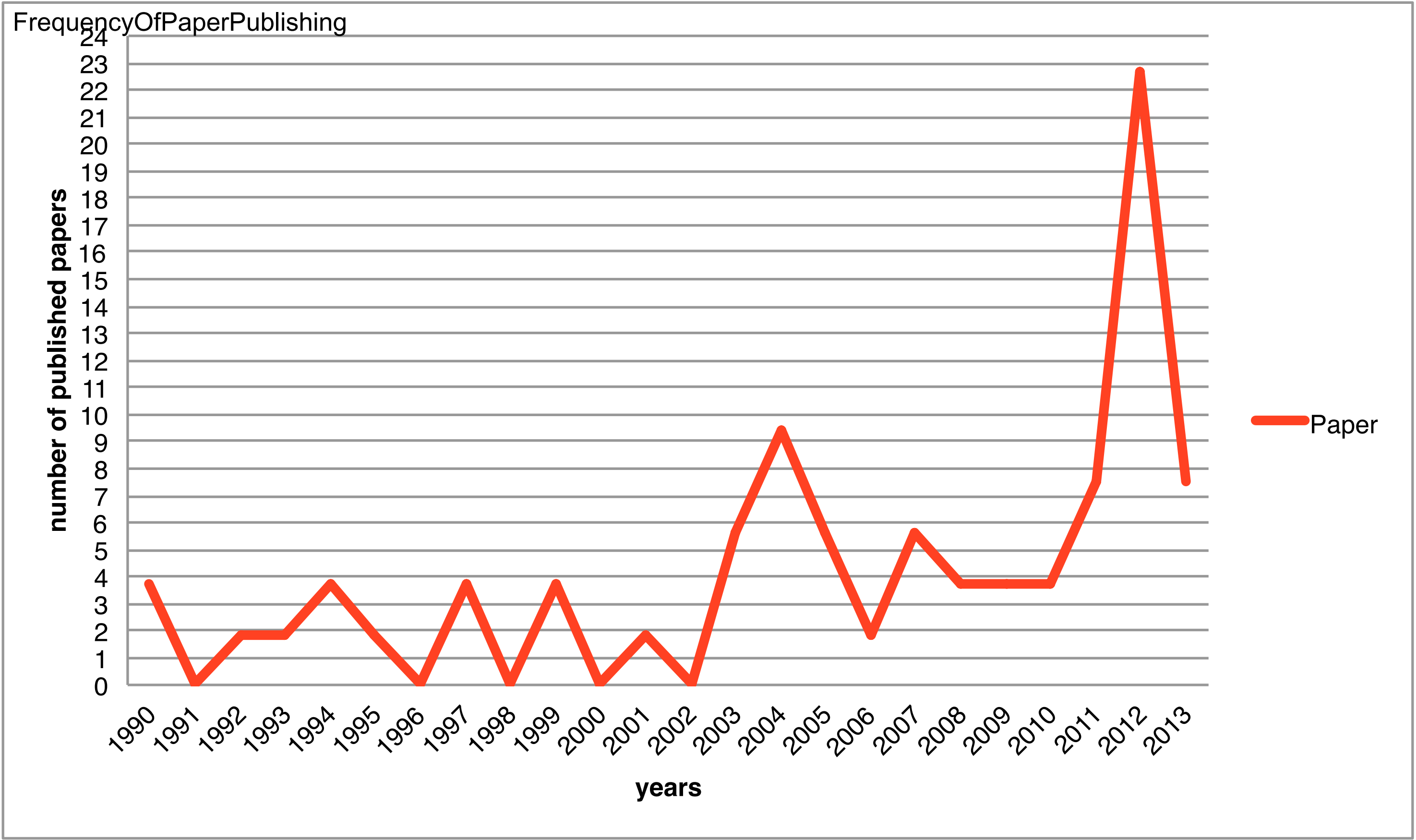
FrequencyOfApplicationPortabilityDiscussedInNon-smartphoneEnvironments

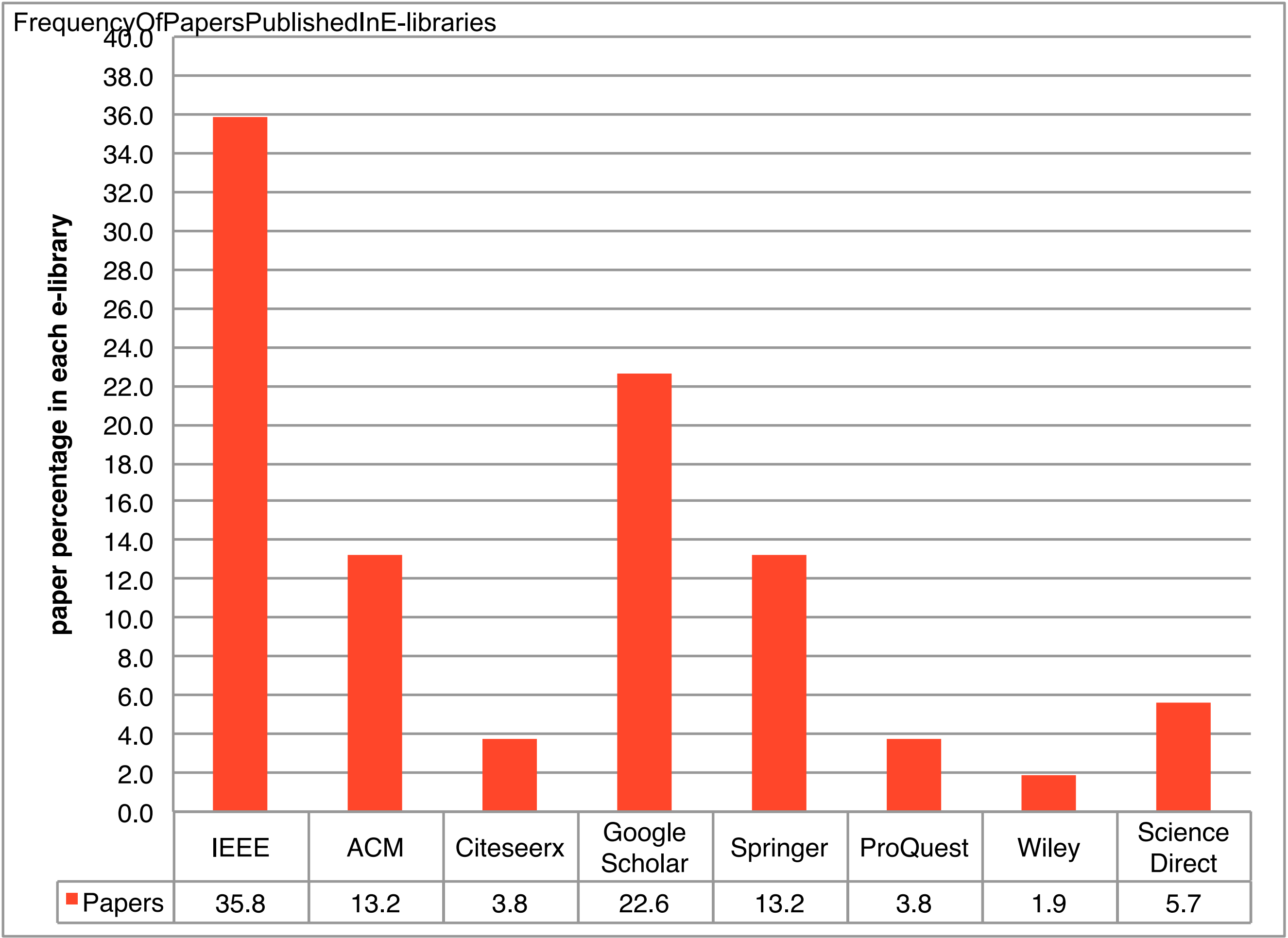


	Desktop\Server	Embedded systems	Non-Specific	Learning Objects	Cloud	Web Services
🟥 Paper	26	20	40	3	6	6

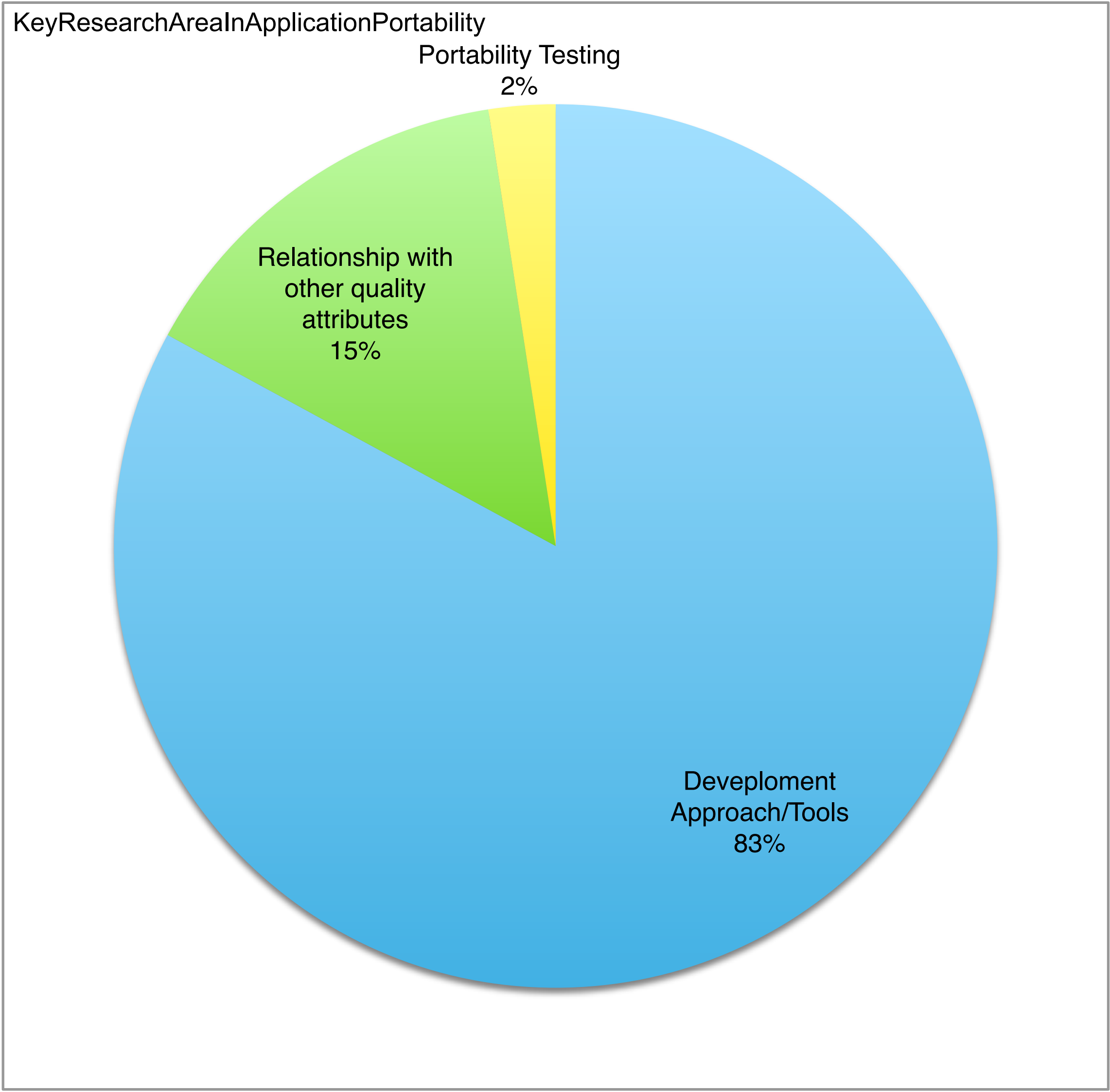
FrequencyOfDevelopmentApproaches



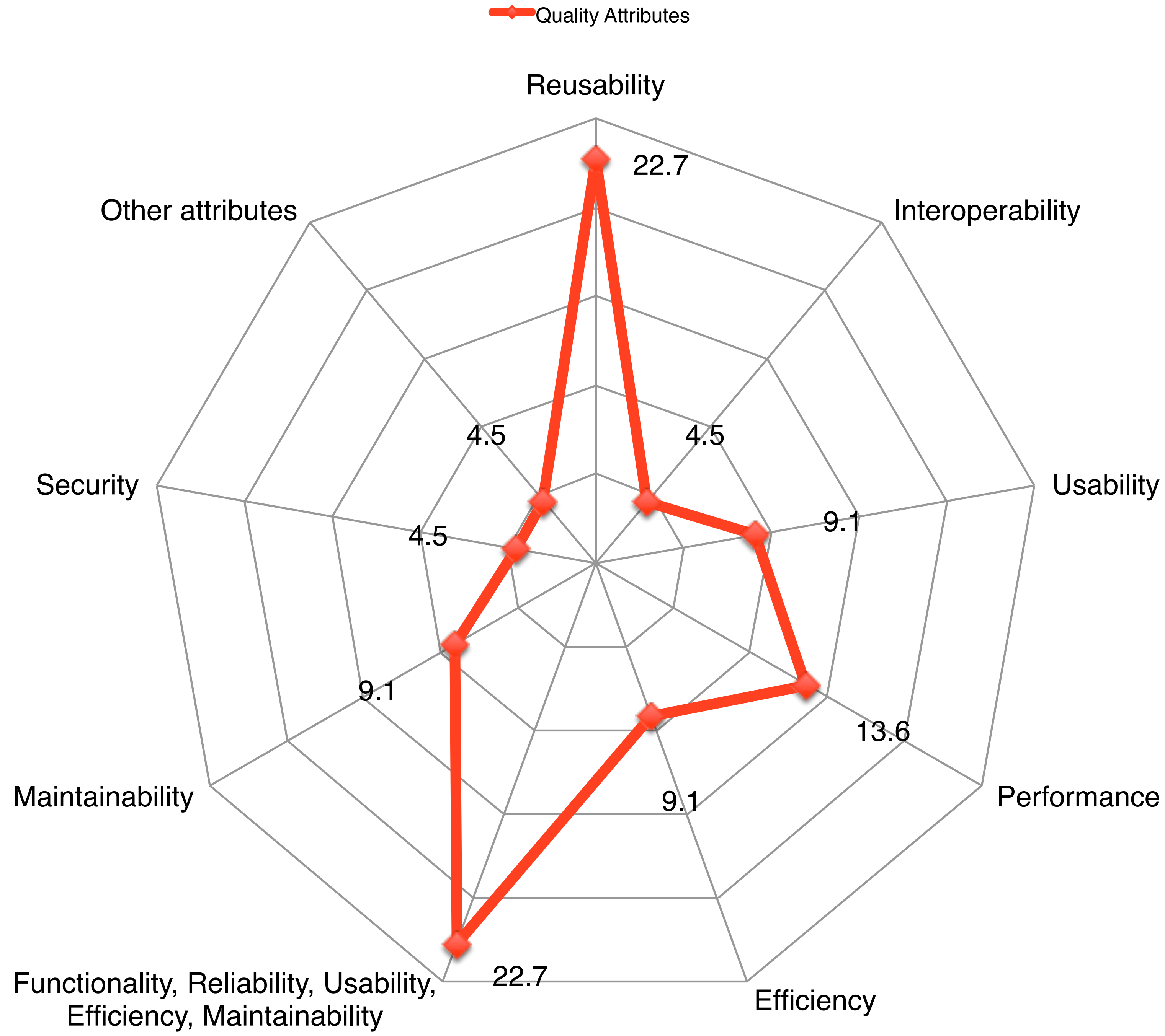

FrequencyOfPaperPublishing
number of published papers
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
years
Paper



	IEEE	ACM	Citeseerx	Google Scholar	Springer	ProQuest	Wiley	Science Direct
🟥 Papers	35.8	13.2	3.8	22.6	13.2	3.8	1.9	5.7


KeyResearchAreaInApplicationPortability
Portability Testing
2%
Relationship with
other quality
attributes
15%
Deveploment
Approach/Tools
83%

SoftwareQualityAttributesDiscussedWithSoftwarePort ability


Quality Attributes
Reusability
Interoperability
Usability
Performance
Efficiency
Functionality, Reliability, Usability, Efficiency, Maintainability
Maintainability
Security
Other attributes