5W+1H pattern: A perspective of systematic mapping studies and a case study on cloud software testing

- Changjiang Jia  
  City University of Hong Kong, and 
  National University of Defense Technology, Changsha, China
- Yan Cai  
  Chinese Academy of Sciences, Beijing, China
- Yuen Tak Yu  
  City University of Hong Kong
- T.H. Tse  
  The University of Hong Kong

Laboratory of Software Engineering and Methodology (LOSEM)  
Department of Computer Science  
City University of Hong Kong  

February 2015  
(Revision of version: August 2014)

This technical report documents the full details of a two-phase case study of a mapping study which adopts the 5W+1H pattern proposed in an article (of the same authors and title) published in:  

Copyright © 2014, 2015 by the authors. All rights reserved.

This report is presented by the contributing authors to ensure timely dissemination of scholarly and technical information on a noncommercial basis. Personal use of this material is permitted. Copyright and all rights therein are retained by the authors or by other copyright holders, despite the fact they have offered this information electronically. It is understood that all individuals copying this information will adhere to the terms and constraints invoked by each author’s copyright.

Permission to reprint, republish or further dissemination of this material for commercial, advertising or promotional purposes, or for creating new collective works for resale or redistribution to servers or lists, or to reuse any component of this work in other works must be obtained from the authors or other copyright holders.

Visit the website of LOSEM at http://losem.cs.cityu.edu.hk for email contact and queries on the use of this report.
This page is intentionally left blank.
5W+1H pattern: A perspective of systematic mapping studies and a case study on cloud software testing*

Changjiang Jia\textsuperscript{a,b}, Yan Cai\textsuperscript{c}, Yuen Tak Yu\textsuperscript{a}, T.H. Tse\textsuperscript{d,†}

\textsuperscript{a} Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Hong Kong
\textsuperscript{b} Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China
\textsuperscript{c} State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
\textsuperscript{d} Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong

ABSTRACT

A common type of study used by researchers to map out the landscape of a research topic is known as \textit{mapping study}. Such a study typically begins with an exploratory search on the possible ideas of the research topic, which is often done in an unsystematic manner. Hence, the activity of formulating research questions in mapping studies is ill-defined, rendering it difficult for researchers who are new to the topic. There is a need to guide them kicking off a mapping study of an unfamiliar domain. This report proposes a 5W+1H pattern to help investigators systematically examine a generic set of dimensions in a mapping study toward the formulation of research questions before identifying, reading, and analyzing sufficient articles of the topic. We have validated the feasibility of our proposal by conducting a case study of a mapping study on cloud software testing, that is, software testing for and on cloud computing platforms. The case study reveals that the 5W+1H pattern can lead investigators to define a set of systematic, generic, and complementary research questions, enabling them to kick off and expedite the mapping study process in a well-defined manner. We also share our experiences and lessons learned from our case study on the use of the 5W+1H pattern in mapping studies.

Keywords

5W+1H pattern; cloud software testing; systematic mapping study

Highlights

• Novices often find it difficult to kick off a mapping study or literature review
• A 5W+1H pattern is proposed to ease the difficulty of conducting literature surveys
• The 5W+1H pattern helps formulate initial research questions and structure reports
• A case study shows the applicability of the 5W+1H pattern and the lessons learned
• A three-year mapping study on cloud software testing research is presented

* This research is supported in part by the Early Career Scheme and the General Research Fund of the Research Grants Council of Hong Kong (project numbers 125113, 716612 and 717811) and the National Natural Science Foundation of China (project number 91418206).
† Corresponding author.

Email addresses: cjjia.cs@gmail.com (C. Jia), ycai.mail@gmail.com (Y. Cai), csytyu@cityu.edu.hk (Y.T. Yu), thtse@cs.hku.hk (T.H. Tse).
1. Introduction

Software testing consumes at least 30% of the whole software development budget [45]. Cloud computing [10] aims at providing a highly customizable and resourceful platform to deploy software [8]. The former is resource-hungry while the latter encompasses abundant computing resources. Does cloud computing solve or amplify the issues faced by software testing? For ease of presentation, we refer to the intersection area between Cloud computing and Software Testing as CST interface in this report.

Systematic literature review (SLR) [25][28] is a dominant approach to conducting survey on a research topic. It aims at producing an “engineering” approach with a well-defined methodology so that different investigators can produce survey results effectively and reliably. In particular, mapping study (MS) [4][6][7][21][26][37][38] is “a more ‘open’ form of an SLR, intended to ‘map out’ the research that has been undertaken rather than to answer a detailed research question” [7].

At the core of a typical SLR, including an MS, is a well-defined literature review protocol, which is expected to be conformed to by investigators when they conduct SLRs and MSS [37]. For instance, according to the protocol, a key activity in the planning phase of a typical SLR project [21] is to formulate a set of Research Questions (RQs) before identifying, reading, and analyzing articles of the topic. For an MS project, the set of RQs to be answered is shaped by an exploratory and yet unstructured search of some selected articles to frame a preliminary impression of the topic. Based on a limited subset of the articles studied, investigators may formulate a set of exploratory RQs without knowing for sure the RQs’ relevance to the topic under study. The set of RQs may evolve as more articles are reviewed by the investigators. However, when it comes to a topic with which the investigators are not truly familiar, formulating a comprehensive, coherent, probing, and reliable set of RQs on the topic before understanding a sufficient and unbiased set of relevant articles is indeed challenging, rendering it difficult to kick off an MS.

We observe that existing work on MS methodology [25][27][49][53] focuses on managing the processes of either literature search [25][49][53] or article categorization [27]. To the best of our knowledge, none of the existing work has provided a systematic way to explore the formulation of RQs in an MS. In this report, we address this problem.

We propose an architectural style based on the 5W+1H (Who, Why, What, Where, When, and How) model [19][36], which is widely used in the journalism domain, to define a high-dimensional design space in which the initial set of RQs can be systematically formulated. We validate the feasibility of our proposal via a two-phase case study of the MS of CST interface. While the empirical results revealed from our case study (presented in Section 3) are interesting and worth reporting on their own, they are not the primary focus of this document. Rather, we would like to report the evidence that this 5W+1H model-based architectural style can guide investigators to make systematic progress in surveying the state of the art of a research topic with which they are originally unfamiliar.

Our work consists of three parts. In the first part, we proposed a 5W+1H pattern1 to structure RQs and contrast the questions with the findings from the MS. This model-based pattern investigates a research topic from six different dimensions. Table 1 illustrates the pattern, which consists of six sections, one for each dimension (Who, Why, What, Where, When, and How). In each section, the pattern defines a placeholder for the RQ, a placeholder for the corresponding conjecture(s) relevant to the RQ to be “mapped out” (or verified) via the MS, a list of placeholders for the major findings that summarize the facts and statistics found, and a placeholder for assessment, which assesses the RQ and conjecture(s) based on the major findings.

In the second part, we applied the 5W+1H pattern to conduct a case study of an MS of CST-interface research. In a preliminary version [23] of this report, we presented a brief summary of the results obtained in Phase 1 of the case study, which was conducted in June 2012. In this first phase, we applied the 5W+1H pattern to generate an initial set of RQs. Then, we followed an existing MS protocol to conduct the MS. By applying the search keywords, the inclusion criterion and the exclusion criteria to three representative databases (ACM Digital Library [1], IEEE Xplore Digital Library [20] and Scopus abstract and citation database [40]), we collected 38 papers published in 2010 and 2011 (as listed in Appendix I). The results were summarized in a 5W+1H pattern to produce Table 1. Phase 1 provided us with preliminary knowledge of the state of the art of CST-interface research. To improve the RQs and classification scheme used in Phase 1, we progressed to Phase 2 of the case study in June 2013. Following the same 5W+1H pattern and MS protocol, we collected and analyzed 13 papers published in 2012 (as listed in Appendix II) on CST interface. We then integrated the results obtained in Phase 1 (summarized in Table 1) with the new and consolidated results obtained in Phase 2 (summarized in Table 2) using the same 5W+1H pattern. This technical report documents the full details of the MS of the complete case study.

In the third part, we reflected on the experiences from our two-phase case study on CST interface. We summarize the results of the case study according to the 5W+1H pattern and instantiate the pattern into contents in Table 1 (Phase 1) and Table 2 (Phase 2). We conclude from the complete case study that this 5W+1H pattern can lead us to define a set of generic and complementary RQs, which allow us to easily kick off and expedite the subsequent activities in an MS. This finding is encouraging.

To sum up, the main contribution of this report (which includes the results of Phase 1 of our case study conducted in June 2012 and presented in a preliminary version [23] of this report) is threefold. (1) It presents, to the best of our knowledge, the first work that defines an architectural style (the 5W+1H pattern) to guide the structuring of RQs for mapping studies. (2) It provides real-life evidence of the feasibility of using the proposed pattern via a two-phase case study of an MS of CST-interface research. (3) It reports our first-hand experiences and reflective lessons learned from applying the 5W+1H pattern to systematically study a new research topic, which we believe would provide insights as well as practical guides to other researchers who plan to conduct an MS on an unfamiliar topic.

---

1 Our work is inspired by design patterns in software design. A pattern is a template for a general solution to a common design problem. It is documented in seven sections, namely, intent, motivation, applicability, example, code listing, discussion, and assessment consequence.
Table 1. Summary of research questions, conjectures, major findings, and assessments in Phase 1 based on the 5W+1H pattern.

<table>
<thead>
<tr>
<th>RQ1: Who? — Authors and countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjecture C1: Plentiful recent papers have been published by diverse research groups and from different countries across the globe.</td>
</tr>
<tr>
<td>Major findings:</td>
</tr>
<tr>
<td>1. Our M5 in Phase 1 identified a primary set (PS) of 38 papers published in 2010–2011 on CST interface out of an initial set (IS) of 2949 records of papers on the broader area of “cloud testing” research (which includes, for instance, hardware or network testing).</td>
</tr>
<tr>
<td>2. Twenty-two papers were affiliated with China or USA. In each of the other 10 countries, only one research group published papers in the PS.</td>
</tr>
<tr>
<td>3. Some of the top 10 countries that published most cloud testing papers in the IS reported in Scopus contributed no paper to CST interface.</td>
</tr>
<tr>
<td>Assessment: CST interface was not widely researched during the period surveyed. Moreover, the author/country distributions of publications on CST interface differed substantially from those on the broader area of cloud testing. We could not find adequate literature evidence to support conjecture C1.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ2: Why? — Objectives of research</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjecture C2: Papers on multiple cloud service architectural layers (that is, Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS)) address the same kind of technical challenge with regard to the software testing topics.</td>
</tr>
<tr>
<td>Major findings:</td>
</tr>
<tr>
<td>1. Among the total 12 software testing topics identified from papers in the PS, four (1/3 of them) were studied in 14 papers (or 36.8% of all papers in the PS) on more than one cloud service architectural layer.</td>
</tr>
<tr>
<td>2. With regard to the 12 software testing topics and three cloud service architectural layers, only the research on 16 out of all the 36 combinations was reported in the PS, while that of the other 20 combinations was not explored.</td>
</tr>
<tr>
<td>3. Six out of the 38 papers in the PS were survey or viewpoint papers on issues such as testing tool features and opinions of practitioners.</td>
</tr>
<tr>
<td>Assessment: The first finding demonstrated literature evidence that was in line with conjecture C2. Moreover, more than half of the combinations of software testing topics and cloud service architectural layers was not studied, indicating that much opportunity on CST interface remained to be explored. Furthermore, the notably high proportion of survey and viewpoint papers on different aspects of CST interface seemed to indicate that each aspect was rich and “unclear” enough to warrant a separate study.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ3: What? — Research ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjecture C3: Testing research ideas for addressing the challenges in different cloud service architectural layers are very different.</td>
</tr>
<tr>
<td>Major findings:</td>
</tr>
<tr>
<td>1. Papers on IaaS proposed fault-based testing techniques to expose faults in the virtual machine implementations.</td>
</tr>
<tr>
<td>2. Papers on PaaS developed (a) techniques for testing SaaS applications to deal with failure-simulations or issues of nondeterminism in PaaS, (b) a methodology for benchmarking, and (c) strategies to lower testing costs by exploiting the elastic property of PaaS.</td>
</tr>
<tr>
<td>3. Papers on SaaS spanned over nine software testing topics. A diverse set of ideas was studied, including performance metrics across multiple cloud service architectural layers, usage-specific and general models of Testing as a Service (Taas), test parallelization, controllability and observability, test workloads, regression testing, the detection of vulnerability faults, and so on.</td>
</tr>
<tr>
<td>Assessment: The only idea common across different layers was fault-based testing. Hence, our data were consistent with conjecture C3. Research at a lower layer involved either randomized or fault-based testing, while many other systematic software testing approaches were not explored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ4: Where? — Patterns of papers at different cloud service architectural layers and types of publication venues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjecture C4: Every cloud service architectural layer receives good research attention. Also, consistent with the norm for computer science research, the majority of recent papers are published as research articles in conference proceedings, and yet there is a good presence of journal papers.</td>
</tr>
<tr>
<td>Major findings:</td>
</tr>
<tr>
<td>1. Papers on SaaS contributed to 57.9% of the PS, followed by PaaS (15.8%) and IaaS (10.5%). The rest (15.8%) were survey or viewpoint papers.</td>
</tr>
<tr>
<td>2. 92.1% of the articles were published in conference/workshop proceedings. Only a very low percentage (7.9%) of the articles were journal papers.</td>
</tr>
<tr>
<td>Assessment: Papers tended to focus on testing challenges at the upper layer (SaaS). Both the ratio (5.5 : 1) between SaaS and IaaS papers and the ratio (11.7 : 1) between conference/workshop and journal papers were notably high, indicating that CST-interface research was not yet mature. Moreover, most articles were published in conference/workshop proceedings. On the issue of publication venues, our data were consistent with conjecture C4. On the other issue, namely, research attention to different cloud service architectural layers, our data did not support conjecture C4.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ5: When? — Article citation immediacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjecture C5: Many papers are promptly cited by other papers.</td>
</tr>
<tr>
<td>Major findings:</td>
</tr>
<tr>
<td>1. Among all the 16 papers published in 2010, 43.8% of them received citations within the same year. The proportion of papers published in 2010 and cited in 2011 went up to 68.8%.</td>
</tr>
<tr>
<td>2. Among all the 38 papers in the PS, 39.5% of them were cited by papers within the PS.</td>
</tr>
<tr>
<td>Assessment: Many papers received prompt research attention in terms of citations, which was consistent with conjecture C5.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ6: How? — Article interrelevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjecture C6: Many papers on various software testing topics and cloud service architectural layers are interreferenced to evolve CST-interface research.</td>
</tr>
<tr>
<td>Major findings:</td>
</tr>
<tr>
<td>1. In terms of citation relationships within the PS, the three topic-layer combinations, robustness testing in the IaaS layer, testing parallelization in the SaaS layer, and integration testing in the SaaS layer, were the most cited ones in CST-interface research.</td>
</tr>
<tr>
<td>2. Three topic-layer combinations, fuzzing in the IaaS layer, migration testing in the SaaS layer, and log analysis in the SaaS layer, had no citation relationship with others within the PS.</td>
</tr>
<tr>
<td>3. Citation relations between papers at the same cloud service architectural layers are much more common than those across layers. Only 39.1% of all citation relationships within the PS involved pairs of papers on different software testing topics at different cloud service architectural layers.</td>
</tr>
<tr>
<td>4. Topics on IaaS and PaaS, as well as their intersections with SaaS, were not extensively explored to evolve CST-interface research.</td>
</tr>
<tr>
<td>Assessment: The findings indicated that interreferencing to evolve CST-interface research was not yet a mainstream practice. We did not find adequate evidence to support conjecture C6.</td>
</tr>
</tbody>
</table>
Table 2.
Summary of research questions, conjectures, major findings, and assessments in Phase 2 based on the 5W+1H pattern.

<table>
<thead>
<tr>
<th>RQ1: Who? — Authors and countries</th>
<th>Conjecture C1:</th>
<th>Major findings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Plentiful recent papers have been published by diverse research groups and from different countries across the globe.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. Our MS in Phase 2 identified a validating set (VS) of 13 papers published in 2012 on CST interface. Together with Phase 1, the combined paper set, denoted by PVS (= PS ∪ VS), included a total of 51 papers on CST interface published in the period 2010–2012.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Six papers in the VS were affiliated with China or USA. In each of the other seven countries, only one research group published one paper in the VS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Some of the top 10 countries that published most cloud testing papers reported in Scopus contributed no paper to CST interface.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Assessment</strong>: Compared with Phase 1, we found some new research groups and new countries that conducted research on CST interface. However, at the same time, some countries that published papers in the PS did not publish any paper in the VS. In absolute terms, even if we considered the PVS as a whole, we still found that CST interface had not been widely researched so far. We could not find adequate literature evidence to support conjecture C1. Moreover, similar to Phase 1, the author/country distributions of the VS differed significantly from those on cloud testing.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ2: Why? — Objectives of research</th>
<th>Conjecture C2:</th>
<th>Major findings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Papers on multiple cloud service architectural layers (that is, Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS)) address the same kind of technical challenge with regard to the software testing topics.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. Our MS classified the VS papers into seven testing topics, namely, six of the topics identified in Phase 1 together with one new topic. Two topics were studied at only one cloud service architectural layer in Phase 1 but at different new layers in Phase 2. Thus, treating the PVS as a whole, six software testing topics (involving 49% of the papers) were found in Phase 2 to be studied at more than one cloud service architectural layer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. With regard to the 13 software testing topics and three cloud service architectural layers, the VS papers studied two combinations that had not been explored by the PS papers. Treating the PVS as a whole, again 20 out of all the 39 combinations remained unexplored.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Three out of the 13 papers in the VS were survey or viewpoint papers on issues such as testing tool features and opinions of practitioners.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Assessment</strong>: The first finding shows that our data were in line with conjecture C2. In both Phase 1 and Phase 2, more than half of all combinations of software testing topics and cloud service architectural layers had not been explored. The proportions of survey and viewpoint papers were also high. Thus, the findings in Phase 2 were consistent with those found in Phase 1.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ3: What? — Research ideas</th>
<th>Conjecture C3:</th>
<th>Major findings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Testing research ideas for addressing the challenges in different cloud service architectural layers are very different.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. The only paper in the VS on IaaS proposed to build a lightweight cloud component model to scale up the testing of the integration between the deployed software and a large number of cloud components.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Papers in the VS on PaaS proposed (a) to create multiple testing instances and distribute them to different cloud nodes to speed up testing execution, (b) a methodology to handle the component dependencies, and (c) strategies to generate workloads for testing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Papers in the VS on SaaS spanned over four software testing topics. A diverse set of ideas had been studied, including generating statically-balanced testing workload distribution, speeding up regression testing with distributed cloud computation framework, identifying vulnerability points from outside of a cloud, and the development of cloud stubs to improve the structural coverage of unit testing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Assessment</strong>: There was no common idea across different layers among the papers in the VS. Our findings were consistent with conjecture C3.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ4: Where? — Patterns of papers at different cloud service architectural layers and types of publication venues</th>
<th>Conjecture C4:</th>
<th>Major findings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Every cloud service architectural layer receives good research attention. Also, consistent with the norm for computer science research, the majority of recent papers are published as research articles in conference proceedings, and yet there is a good presence of journal papers.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. Papers on IaaS, PaaS, and SaaS contributed to 9.8%, 21.6%, and 51.0% of the PVS, respectively. The rest (17.6%) were survey or viewpoint papers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. 76.9% of the articles in the VS were published in conference/workshop proceedings. Articles in journal papers only accounted for 23.1%.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Assessment</strong>: The lowest cloud service architectural layer (IaaS) still received the least attention from the research community. Most articles were published in conference/workshop proceedings. As in Phase 1, our data were consistent with conjecture C4 on the issue of publication venues, but did not support conjecture C4 on the issue regarding research attention to different cloud service architectural layers.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ5: When? — Article citation immediacy</th>
<th>Conjecture C5:</th>
<th>Major findings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Many papers are promptly cited by other papers.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. The proportion of papers published in 2010 receiving citations further increased from 68.8% in 2011 to 93.8% in 2012.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. 49.0% of the papers in the PVS were cited by papers within the PVS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Assessment</strong>: Again, many papers received prompt research attention in terms of citations, which was consistent with conjecture C5.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ6: How? — Article interrelevance</th>
<th>Conjecture C6:</th>
<th>Major findings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Many papers on various software testing topics and cloud service architectural layers are interreferenced to evolve CST-interface research.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. In terms of citation relationships within the PVS, the same three topic-layer combinations, robustness testing in the IaaS layer, testing parallelization in the SaaS layer, and integration testing in the SaaS layer, were the most cited ones in CST-interface research.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Two new topic-layer combinations, integration testing in the IaaS layer and unit testing in the SaaS layer, had no citation relationship with others within the PVS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Only 41.1% of all citation relationships in the PVS involved pairs of papers on different software testing topics at different cloud service architectural layers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. Topics on IaaS and PaaS, as well as their intersections with SaaS, were not explored extensively to evolve CST-interface research.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Assessment</strong>: Although the proportion of citation relationships across different software testing topics and different cloud service architectural layers increased slightly from 39.1% in Phase 1 (2010–2011) to 41.1% in all three years (2010–2012), the two percentages were moderate only, not high. Thus, our data did not provide strong support to conjecture C6.</td>
</tr>
</tbody>
</table>
The rest of this report is organized as follows. Section 2 revisits the 5W+1H model and proposes a 5W+1H pattern for applying the model to MS. Section 3 presents a case study of applying the 5W+1H pattern to structure RQs and report our findings of the MS, which are summarized in Table 1 and Table 2. Section 4 reports our experiences gained from the case study of applying the 5W+1H pattern to conduct an MS of CST-interface research. Section 5 discusses related work. Section 6 concludes the paper. Appendix I and Appendix II list the collection of papers that we examined in the two phases of the case study, respectively. Appendix III provides a brief summary of each testing topic studied by the papers we collected in the complete case study.

2. A 5W+1H pattern-based approach

Mapping study seeks to map out the state of research of a topic. In this section, we revisit the 5W+1H model and elaborate how we apply it to formulate a 5W+1H pattern to refine the free-form exploration in the planning phase of existing typical MS projects.

2.1. The 5W+1H model revisited

The term “5W+1H” is an abbreviation of six keywords: Who, Why, What, Where, When, and How. The 5W+1H model represents the majority needs of what people want to know about a news story. Kipling, an English writer, first mentioned the 5W+1H model in his book entitled Just So Stories [24] in 1902. Later, journalists widely applied this model to report news. From the perspective of journalists, to report a story, the readers should be supplied with essential information on six questions [19][36]:

1. Who performed the actions in the story (or who experienced the results)? [Actor]
2. Why did the actions occur? [Motivation]
3. What were the actions and what happened as a result of the actions? [Content]
4. Where did the actions take place? [Location]
5. When did the actions occur? [Time]
6. How did the actions connect to each other? [Causality]

On the other hand, an MS aims at synthesizing existing work on a research topic to obtain a comprehensive and objective understanding of the topic by mapping out the classifications of work done within the review scope of the MS. Thus, both journalists and MS investigators share the goal of seeking to understand and report certain activities (news events or research undertakings) comprehensively.

Typically, investigators conduct MSs by following a well-formed protocol [25][26] to collect existing literature and then analyze it to categorize findings based on a set of pre-proposed RQs. Thus, the understanding of an investigator when conducting an MS is largely determined by the pre-proposed RQs. Existing MS guidelines [6][7][21][30][37][38] suggest starting with an exploratory formulation of RQs by reading some selected articles relevant to a topic. For researchers knowledgeable in the domain, proposing a relevant set of coherent and probing RQs thereafter may not be difficult; however, this task can be challenging to investigators new to the research topic.

We recall that a goal of the MS methodology is to provide a well-defined protocol for one to follow so that different investigators can more or less produce similar results (so that the process can be engineering-oriented and repeatable). Simply asking investigators to explore some articles and developing an exploratory set of RQs without a set of concrete guidelines may be too abstract and unsystematic.

The 5W+1H model provides six dimensions to completely report events of interest. We propose that it can benefit MS investigators by relieving their challenges in defining the initial set of RQs and providing them with guides to perspectives that are not necessarily seen from other MSs in similar topics (such as service-based testing versus software testing). In the next section, we will elaborate our view on how to apply the 5W+1H model to MS.

2.2. Applying the 5W+1H model to a mapping study

Generally speaking, the purpose of writing or referencing (instead of publishing) an MS is to quickly understand the research state or progress of a topic as well as to identify the gaps or new problems for further research investigations. Our case study is going to show that the six dimensions of the 5W+1H model have the potential to guide an MS by defining RQs that could easily be missed (due to omission or negligence) or dismissed (due to bias or premature judgment) as “uninteresting”. We believe that it is unscientific to presume the lack of interests in certain RQs before soliciting objective grounds or empirical evidences to support the judgment. We also believe that it is unscientific to skip a whole dimension (say, due to the lack of interesting findings perceived by the investigators) in conducting the MS. This is because in either case, the result of the MS will be heavily and subjectively directed toward the selected dimensions and positive findings, resulting in biases in the publications.

Specifically, we adapt the 5W+1H model to the context of conducting an MS of a research topic and formulate a 5W+1H pattern that helps us define and focus on the initial RQs for studying the topic.

1. Who: the researchers
2. Why: the motivations and objectives of proposing the research problems
3. What: the research ideas and issues
4. Where: the locations of the research problems in terms of their positions in the topic context and venues of publication
5. When: the publication date
6. How: the interconnections among individual problems

Our MS case study was framed and guided by the above 5W+1H pattern. Specifically, we formulated a set of RQs for exploring CST-interface research from the six dimensions of the 5W+1H pattern. We extracted author information to report the researchers Who were active in research of the topic. We examined the motivations of the reviewed articles to understand Why the authors believed the research was necessary. We identified What software testing ideas, issues and topics were studied in the reviewed literature. Then, we located Where each paper was situated within a two-dimension classification scheme that integrated the software testing topics with the three-layered cloud service architectural structure [3][33]. (See the end of Section 3.1...
for a brief explanation of the layered structure.) We also studied the distribution of papers in different venues Where they were published. We counted the papers published and cited within the review period to see When the publications of CST-interface research appeared and received attention from the research community, respectively. Finally, we analyzed the citation relationships to explore How CST-interface research had interacted and evolved across different software testing topics and cloud service architectural layers.

3. Feasibility case study of applying the 5W+1H pattern: A mapping study of CST interface

In this section, we present a case study that validates the feasibility of our proposal. We follow existing MS guidelines [25][30][37] to organize the contents as follows: Section 3.1 presents a set of six RQs formulated by applying the 5W+1H pattern. Section 3.2 describes the process of identifying relevant articles in the literature. Section 3.3 outlines the quality assurance measures taken when we identified relevant articles for the study. Section 3.4 analyzes the collected papers and answers the RQs. Section 3.5 discusses the threats to validity of our case study.

3.1. The 5W+1H pattern

To portray a contemporary picture of CST-interface research, we adopted the 5W+1H pattern developed in Section 2.2 and then instantiated it into the context of studying the topic of CST-interface research to pre-propose one RQ for each of the six dimensions as follows:

RQ1: Who (which researchers or groups) were doing research in CST interface?
RQ2: Why were the research studies needed? That is, what research objectives were stated in the articles?
RQ3: What kinds of software testing research ideas were presented in the articles?
RQ4: Where were the articles published? Did the articles appear in typical types of publication venues? On which cloud service architectural layers [3] were the articles focused?
RQ5: When did the articles start to show impact? Were the articles immediately cited by other articles?
RQ6: How were the articles interreferenced among various software testing topics and cloud service architectural layers?

Using the 5W+1H pattern, we further formulated a conjecture for each RQ according to the common perception of computer scientists. We would like to assess to what extent the collected papers present evidence to support or refute the following conjectures:

C1: Plentiful recent papers have been published by diverse research groups and from different countries across the globe.
C2: Papers on multiple cloud service architectural layers address the same kind of technical challenge with regard to the software testing topics.
C3: Testing research ideas for addressing the challenges in different cloud service architectural layers are very different.
C4: Every cloud service architectural layer receives good research attention. Also, consistent with the norm for computer science research, the majority of recent papers are published as research articles in conference proceedings, and yet there is a good presence of journal papers.
C5: Many papers are promptly cited by other papers.
C6: Many papers on various software testing topics and cloud service architectural layers are interreferenced to evolve CST-interface research.

In formulating the above RQs and conjectures, we made use of a common key notion of cloud service architectural layers (or simply cloud layers) [3][33] in the domain of cloud computing that we recognized when reading the papers to determine whether they should be included in our collection. As mentioned in Section 1 and also seen from the above RQs and conjectures, this key notion was instrumental when we developed and refined a classification scheme to study the What and Where dimensions, as well as when we refined the interreferencing patterns in the How dimension to understand how the different software testing topics and cloud service architectural layers interacted to evolve CST-interface research.

To facilitate understanding and for the sake of self-completeness of this report, we briefly introduce here the generally accepted classification of three cloud service architectural layers of a cloud computing platform [3][33]. The three layers are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

- IaaS provides users of virtualized computing resources with configurable virtual machines. Examples of testing issues in the IaaS layer include testing whether the virtualized resources emulate their physical counterparts precisely [S19] and testing software robustness against hardware device failures simulated by configuring virtual machines [S12][S13].
- PaaS provides users with a virtual platform to build their own SaaS. Example components of such a platform include a guest operating system, storage, platform-based software, and application programming interfaces (APIs). Examples of testing issues include testing properties specific to the PaaS platform (due to the nondeterminism property of the MapReduce framework) [S7] and testing the behavior of software against various invocation sequences of cloud APIs [S15].
- SaaS provides users with usable software applications to operate or synthesize new applications on top of the SaaS. Examples of testing issues include testing parallelization on multiple cloud nodes [S3][S30], testing the composition of cloud services [S31][S33], and regression testing against the changes of synthesized cloud services [S6].

3.2. Paper identification process

In this section, we present the paper selection processes of our case study. We note that the main goal of the case study is to demonstrate the feasibility of adopting the 5W+1H pattern in an MS.
### 3.2.1. Phase 1: Identification of the primary set (PS) of papers

Phase 1 of the case study was performed in June 2012. Figure 1 depicts the paper search and selection process.

**Databases.** Our MS used three popular databases to identify the literature: ACM Digital Library (ACM DL) [1], IEEE Xplore Digital Library (IEEE Xplore) [20], and Scopus abstract and citation database (Scopus) [40].

**Search keywords for Phase 1.** We used two search keywords “Cloud” and “Testing” as the base and enumerated their popular variants to construct the following final compound keyword:

> “Cloud AND (Testing OR Analysis OR Test OR Analyzing OR Analyze)”

**Inclusion criterion (IC) and exclusion criteria (EC).** Following the practice of Kitchenham et al. [25], we specified an initial inclusion setting (IIS) by matching the search keywords with the abstract of a paper. We focused on peer-reviewed publications to ensure that the papers in our collection at least reached acceptable and publishable quality. To study the progress on CST-interface research, we included papers published in the two whole years, 2010 and 2011, just before Phase 1 was conducted.

- **Initial inclusion setting (IIS):** Search the (compound) keyword in the abstract of a paper that was published in a refereed journal or conference proceedings in 2010 or 2011 and is within the smallest domain that includes computer science.

Specifically, we set the publication venue as journals, proceedings, and transactions for ACM DL, journals & magazines and conference publications for IEEE Xplore, and journals and conference proceedings for Scopus. We set the topic domain as Computing & Processing (Hardware/Software) for IEEE Xplore and Computer Science for Scopus. We could not specify any topic domain for ACM DL as it provided no such option. We then refined the IIS in terms of actual search keywords to form the following inclusion criterion (IC):

- **IC:** Apply IIS using the compound keyword “(Cloud) AND (Testing OR Analysis OR Test OR Analyzing OR Analyze)”

By searching via the IC, we extracted an initial set (IS) of 2949 paper entries. We then applied the following three exclusion criteria (EC1–EC3) in stages to further refine the IS:

- **EC1:** Exclude a paper with fewer than four pages.
- **EC2:** Exclude a paper that mentions no issue on cloud computing or software testing in its abstract.
- **EC3:** Exclude a paper that mentions no issue on cloud computing or software testing in either the introduction or conclusion of the paper. Remove duplications due to multiple records that refer to the same paper.

We followed the practice of Kitchenham et al. [25] and applied EC1, which reduced the size of the IS to 2807. EC2 eliminated a large number of papers on irrelevant topics such as storage, hardware configuration, and network, thereby further reducing the number of records to 91. For EC3, we examined the papers’ introductions and conclusions in addition to the abstracts. To filter out duplicated entries, we first kept all the records from ACM DL. Then, for each record in the two subsequent databases (namely, IEEE Xplore followed by Scopus), we removed a record if it had already been found in a previous database. Figure 1 shows the number of records obtained successively via the IC and then EC1–EC3 in Phase 1. We then applied snowballing [25] to examine the reference lists of the selected papers to see whether we might have missed any important articles. In the last step, we identified one additional article ([S5]).

Note that the results of executing EC2 and EC3 are subjectively determined by the expertise of investigators. Thus, the two steps pose a threat to the validity of including papers in the MS. To minimize the validity, the investigators of the MS first reviewed the papers independently and then merged their results together to collaboratively resolve any inconsistency of decisions of whether a paper should be included. Finally, a paper was included in the MS only if all the investigators reached an agreement. Nevertheless, the unanimous decisions of the investigators could still be wrong. Thus, we add this potential problem as a threat to the validity of this MS.

We finally obtained 38 distinct papers, as listed in Appendix I. We refer to this collection of papers as the primary set (PS). We noted that only 1.3% (= 38 ÷ 2949) of the paper entries in the IS were related to both cloud computing and software testing and, hence, included in the PS.

### 3.2.2. Phase 2: Identification of the validating set (VS) of papers

Considering the validation purpose of Phase 2 instead of conducting a new MS, we chose to search papers in Scopus because of its largest literature coverage among the three databases used in Phase 1. With the prior understanding of CST-interface research obtained in Phase 1, we refined the compound search keyword to

> “(Cloud OR IaaS OR PaaS OR SaaS OR TaaS) AND (Testing OR Test)”,

Figure 1. The process of identifying the primary set (PS) of papers in Phase 1.
where \textit{IaaS}, \textit{PaaS}, and \textit{SaaS} are abbreviations of the three cloud service architectural layers, as explained in Section 3.1, and \textit{TaaS} is an abbreviation of the term \textit{Testing as a Service}, which refers to the deployment of testing in the form of a software service in the cloud. The four terms \textit{IaaS}, \textit{PaaS}, \textit{SaaS}, and \textit{TaaS} were added to refine the search keywords because they were so frequently found in cloud computing papers in Phase 1 that we would like to ensure no omission of papers collected in Phase 2 that used only these terms but not the word “cloud” in the abstract or paper title. On the other hand, we noted in Phase 1 that the word “analysis” and its variants were never found alone in papers on CST interface without the co-occurrence of the words “test” or “testing”. Accordingly, variants of the term “analysis” were omitted from the search keyword in Phase 2. A more detailed discussion of our process of improving the search keywords can be found later in Section 3.5.

Next, we set the search configuration in Scopus as follows: search for keywords in \textit{Article Title}, select document type \textit{ALL}, set the date range as \textit{published 2012 to 2012}, and select all subject areas. By executing this search query in June 2013, we extracted 43 papers from Scopus. We refer to this collection of 43 papers, contributed by researchers from 22 countries, as \textit{V-Scopus}. Considering the small size of V-Scopus, we directly read the abstract, introduction, and conclusion of each of the 43 papers, and finally obtained 13 papers published in 2012 that were relevant to CST-interface research, as listed in Appendix II. We refer to this set of 13 papers as the \textit{validating set (VS)}.

In Phase 2, we also needed to combine the two paper sets (PS and VS) to analyze the characteristics of all the papers published in the entire three-year period (2010–2012) of the two phases. We refer to the combined collection of 51 (= 38 + 13) papers in the PS and VS as the \textit{PVS}.

Note that we adopted a slightly different search configuration and a simplified paper selection process in Phase 2 mainly because its purpose was not to exactly replicate Phase 1 but to validate whether the paper classification scheme, analysis process, as well as the findings derived in Phase 1 could be successfully applied to the new set of literature. Despite these differences, we manually read each paper in full detail in Phase 2 to ensure that its quality was acceptable and generally comparable to those collected in Phase 1. More details of the quality assurance measures adopted in Phase 1 and Phase 2 are presented in the next section.

3.3. Quality assurance

We browsed the official website of the publication venue of each paper in the PVS to ascertain that it had been peer-reviewed. Also, by manually reading the full text of each paper in the PVS (in addition to the preceding stages of examining its abstract, introduction, and conclusion), we verified that the paper indeed satisfied the inclusion criterion but was not eliminated by the exclusion criteria, with only a few exceptional papers that we nevertheless decided to include in the MS for the following reasons.

First, we found that although ACM SIGOPS Operating System Review did not specify a paper review process, its paper entitled “Cloud9: A software testing service” [S5] was a published version of [13], which had been peer-reviewed earlier in a workshop with no proceedings. Hence, we decided to include [S5] in the PS. All the other papers in the PS were also verified to have been peer-reviewed, met the inclusion criterion, and were not eliminated by the exclusion criteria. Two of the VS papers [S40][S44] had fewer than four pages, which strictly speaking should have been eliminated according to the first exclusion criterion (EC1). Nevertheless, upon rigorous and thorough scrutiny, we found that the work [S40] was published as a follow-up of [S14] by the same research group, while the other work [S44] was a different version of the paper [S43] from the same research group. Being convinced that analyzing these two closely-related papers could better reveal the relationships among the evolved research ideas, we finally decided to include them in the VS. Every other paper in the VS consists of at least four pages.

As a triangulation check, we also used Google Scholar [18] to reexamine the IC by replacing the search on the abstract setting by a search in the “with all of the words” field in the advanced search menu, setting “where my words occur” to “anywhere in the article”, and using “return articles dated between 2010–2011”. We then scanned through the first 2000 returned records (out of an estimated 18,800 hits) and found two more articles [11][17] that reported certain viewpoints on the topic. Nonetheless, in order to maintain our review protocol, we chose to review them separately in Section 3.4.7 together with the other survey and viewpoint papers identified in the PVS. We did not use Google Scholar as the bibliographic database in our case study because it neither consistently indexed papers according to the publication years nor provided any option to search the abstracts. Searching only the keywords in the paper titles would miss many important references such as [S3][S4][S6][S8][S10][S14][S16][S19][S21][S24][S26][S29][S30][S32].

Further, we adopted our expert judgment to ensure that the PVS papers are of sufficient quality. The threats to validity of the case study are discussed in Section 3.5 of this report. In particular, there is no golden standard as to what constitutes a search string to locate articles from various databases. Although the choice of search strings may affect the outcome of an MS, it would not affect the validation results of the feasibility of pursuing an MS by using the 5W+1H pattern in our case study.

3.4. Data extraction and mapping process

In this section, we present the analysis of the papers in the PS followed by that of the papers in the VS for each of the six RQs and then affirm or reject the corresponding conjectures based on the collected evidences and findings. We report the six dimensions in the following order: the \textit{Who} dimension in Section 3.4.1, the \textit{Why} dimension in Section 3.4.2, the \textit{What} dimension in Section 3.4.3, the \textit{Where} dimension in Section 3.4.4, the \textit{When} dimension in Section 3.4.5, and the \textit{How} dimension in Section 3.4.6. In Section 3.4.7, we discuss the survey and viewpoint papers collected in the PVS together with the two viewpoint papers [11][17] separately identified from Google Scholar.

3.4.1. The \textit{Who} dimension: Analysis

We extracted the author information of each paper in the PVS in order to study the \textit{Who} dimension. When at least one author of a paper \textit{x} is affiliated with country \textit{y}, we simply say “the paper \textit{x} is affiliated with country \textit{y}” or “country \textit{y} published the paper \textit{x}”. Similarly, we write “the country \textit{y} in a paper set \textit{u}" to mean that at least one paper in the paper set \textit{u} is affiliated with country \textit{y}. We then explored the proportion of the authors’ contribution to CST-interface research in terms of paper counts of two attributes:
country and research group. Here, by research group we take the loose meaning of any group of researchers such that each member coauthored at least one paper (in the paper set under discussion) with at least one other member in the group.

The country attribute count reflects the country’s level of activity on CST-interface research. A high research group attribute count indicates that the researchers in the group are prolific in CST interface. For Phase 1, Table 3 and Table 4 show the counts of the PS papers by country and research group, respectively. For Phase 2, the corresponding counts of the VS papers are shown in Table 5 and Table 6.
For the purpose of comparison, in addition to the countries that published papers in the PS and the VS, Table 3 and Table 5 also show the numbers of papers affiliated with each of the top 10 countries that published the largest numbers of papers in the subsets, denoted by IS-Scopus and V-Scopus, of the initial sets obtained from Scopus in Phase 1 and Phase 2, respectively. Note that we did not analyze other papers in the initial sets because the other two databases do not provide an option to search for records by country.

We found that the majority of countries with publications appearing in IS-Scopus and V-Scopus did not publish any paper in the PS or the VS. Specifically, only 12 of all the 67 countries in IS-Scopus and nine of all the 22 countries in V-Scopus published papers in the PS and the VS, respectively. Moreover, out of the top 10 countries that published the largest numbers of papers in IS-Scopus and V-Scopus, some countries (Germany, South Korea, Taiwan, and France, shown in italics in Table 3, and UK, Belgium, France, and Italy, shown in italics in Table 5) did not publish any paper in the PS and the VS, respectively. In absolute terms, USA and China were the most prolific, publishing 18 and 10 papers in the PS for Phase 1, and four and three papers in the VS for Phase 2, respectively.

We also observed that in either phase, some countries had significantly higher proportions of their papers published specifically on CST-interface research than other countries. For instance, in terms of ratios, each of the countries Switzerland, Finland, and Pakistan (shown in boldface in Table 3) had at least 20% of its IS-Scopus papers included in the PS. Moreover, all (100%) of the papers published in V-Scopus by Algeria, Australia, Austria, Turkey, Estonia, and Finland (shown in boldface in Table 5) were actually included in the VS.

In Table 4 and Table 6, we categorized the authors’ research groups as either national (if all the authors were affiliated with the same country) or international (otherwise) for the papers in the PS and the VS, respectively. From Table 4, we found that the international category contains eight (21.1%) of the 38 papers in the PS. The strongest international tie appeared to be between China and USA, which jointly published five (62.5%) of the eight papers in this category. Interestingly, in the national category, we observed that for each country other than China and USA, the same research group published all the papers affiliated with that country.

Apparently the above patterns observed in the PS can generally be carried forward to the VS. For example, from Table 6, in the national category, the same research group published all the papers affiliated with each country other than China and USA, and only one research group (from USA) published more than one paper in the VS. However, only one paper was jointly published by China and USA in 2012, and the proportion of internationally-collaborated publications had significantly declined in 2012, compared with eight such papers published in 2010–2011.

Comparing Table 3 with Table 5 and comparing Table 4 with Table 6, we found that by the year 2012, CST interface had received attention from new research groups (such as GMU in USA) and new countries (such as Turkey).

[Assessment] Before performing the MS, we originally thought that CST interface must have received intensive attention from worldwide research communities. However, the evidences revealed different conclusions. Only 38 (or 1.3%) of the entries in IS could be classified as papers on CST interface and included in the PS, while the VS contained only 13 papers. Because of the small number of papers, the numbers of research groups and countries that published papers on CST interface were also small.

The country and author distributions in the PS and the VS were very different from those of IS-Scopus and V-Scopus, respectively. At the same time, we did find evidences that some papers in the VS were published by research groups that had not published any paper in the PS, showing that CST-interface research was spreading out, albeit slowly. As such, we did not find adequate evidence to support conjecture C1 in both Phase 1 and Phase 2. The above results are summarized in the Who sections of Table 1 and Table 2.

3.4.2. The Why dimension: Analysis

The Why dimension seeks to understand the research objectives and motivations that drive existing work. As we examined the papers to seek answers to the Why RQ, we found it natural to classify the papers according to the software testing topics that addressed the main research challenges. Also, as mentioned in Section 3.1, we came to recognize at the early stages that the key notion of cloud layer plays an important role in the cloud computing domain. As such, we classified the papers according to two dimensions, namely, software testing topic and cloud layer.

To ensure consistency and to reduce potential researcher bias in classifying the papers, we adopted the following protocol. First, two authors of this work independently classified each paper into a certain software testing topic and a cloud layer. Then, they came together to consolidate the results and attempted to resolve any conflict in the classification. Finally, a third author of this work inspected all the consolidated results for quality assurance and cast a “deciding vote” in case of unresolved conflicts. In our case study, the deciding vote turned out to be unnecessary.

Table 7 and Table 8 summarize the results of classifying the PS and VS papers in Phase 1 and Phase 2, respectively. In either table, each row shows a topic in software testing and its topic ID, and the columns entitled IaaS, PaaS, and SaaS represent the respective cloud layers. For each topic, we counted a paper (shown as the reference next to the name of the topic and cloud layer) toward a cloud layer if it addressed software testing issues due to that layer. The last row before the Total shows the survey and viewpoint papers in the paper collection of each phase. To answer the Why RQ more specifically, the rightmost column summarizes the main research ideas of the papers within the same software testing topic, while the challenges in each cloud layer will be summarized at the end of this section.

For the sake of comparison, we also present in Table 7 the proportion of papers of each topic within the PS and present in Table 8 the proportion of papers of each topic not only within the VS but also within the PV5 (= PS ∪ VS). Furthermore, in Table 8, to show the presence of the PS papers in the classification scheme for ease of comparison, we annotated each cell by an asterisk “*” for any topic-layer combination that contained at least one paper in the PS in Table 7.

The combined collection PVS contains a total of 51 (= 38 + 13) papers. Reviewing each of them here would distract readers from comprehending the big picture of our analysis. We therefore choose to include a brief review of each PVS paper in Appendix III for interested readers while we continue with our analysis as follows.
Table 7. Summary of main research ideas by software testing topics and cloud service architectural layers (Phase 1).

<table>
<thead>
<tr>
<th>Topic ID</th>
<th>Software testing topic (Cloud service architectural layer: [references])</th>
<th>No. of papers studying cloud layer</th>
<th>Proportion of papers in the PS</th>
<th>Summary of main research ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fuzzing ([IaaS: S19])</td>
<td>IaaS 1 PaaS 0 SaaS 0</td>
<td>0.026</td>
<td>Produce randomized test cases to trigger residual faults in the implementations of virtual machines via protocol-based fuzzing.</td>
</tr>
<tr>
<td>2</td>
<td>Robustness testing ([IaaS: S22][S12][S13]) ([PaaS: S11][S16])</td>
<td>IaaS 3 PaaS 2 SaaS 0</td>
<td>0.132</td>
<td>Generate customizable fault-based execution traces that are reachable by the applications, each execution trace simulating a failure combination of the underlying platforms.</td>
</tr>
<tr>
<td>3</td>
<td>Concurrency testing ([PaaS: S7])</td>
<td>IaaS 1 PaaS 0 SaaS 0</td>
<td>0.026</td>
<td>Generate test cases with respect to the nondeterministic behavior of an underlying infrastructure component reachable from the application.</td>
</tr>
<tr>
<td>4</td>
<td>Performance testing ([PaaS: S24]) ([SaaS: S10][S28][S32])</td>
<td>IaaS 1 PaaS 3 SaaS 0</td>
<td>0.105</td>
<td>Distribute workloads to reduce virtual machine rental costs. Design a model of system performance metrics that consider different architectural layers.</td>
</tr>
<tr>
<td>5</td>
<td>Testing strategy ([PaaS: S27]) ([SaaS: S35][S36])</td>
<td>IaaS 1 PaaS 2 SaaS 0</td>
<td>0.079</td>
<td>Propose models to use or organize Testing-as-a-Service.</td>
</tr>
<tr>
<td>6</td>
<td>Context sensitivity ([PaaS: S15]) ([SaaS: S4])</td>
<td>IaaS 1 PaaS 1 SaaS 0</td>
<td>0.053</td>
<td>Customize testing or Testing-as-a-Service with respect to specific usage scenarios.</td>
</tr>
<tr>
<td>7</td>
<td>Testing parallelization ([SaaS: S31][S32][S33][S34][S38])</td>
<td>IaaS 6 PaaS 0 SaaS 0</td>
<td>0.158</td>
<td>Parallelize a symbolic or concrete execution so that different fragments can be scheduled to run on a set of virtual machines.</td>
</tr>
<tr>
<td>8</td>
<td>Integration testing ([SaaS: S17][S18][S31][S33])</td>
<td>IaaS 4 PaaS 0 SaaS 0</td>
<td>0.105</td>
<td>Control the service discovery mechanism or provide a virtualized testing platform of a service to improve the test controllability and observability of the service or service composition.</td>
</tr>
<tr>
<td>9</td>
<td>Regression testing ([SaaS: S6][S14])</td>
<td>IaaS 2 PaaS 0 SaaS 0</td>
<td>0.053</td>
<td>Identify the changes among different versions of the same SaaS application to select and fix test cases.</td>
</tr>
<tr>
<td>10</td>
<td>Security testing ([SaaS: S29][S37])</td>
<td>IaaS 2 PaaS 0 SaaS 0</td>
<td>0.053</td>
<td>Expose vulnerability faults of an application due to the use of alternate API implementations or defective code accessible by the application.</td>
</tr>
<tr>
<td>11</td>
<td>Migration testing ([SaaS: S8])</td>
<td>IaaS 1 PaaS 0 SaaS 0</td>
<td>0.026</td>
<td>Translate between test requests induced by the same application on different platforms over the same test case and triage failures to ease failure diagnosis.</td>
</tr>
<tr>
<td>12</td>
<td>Log analysis ([SaaS: S21])</td>
<td>IaaS 1 PaaS 0 SaaS 0</td>
<td>0.026</td>
<td>Construct a generic log format to support different kinds of log analyses.</td>
</tr>
<tr>
<td></td>
<td><strong>Subtotal</strong></td>
<td>IaaS 4 PaaS 6 SaaS 22</td>
<td>0.842</td>
<td>Either apply a different classification scheme to survey existing cloud software testing work or report the industry’s understanding of cloud software testing issues.</td>
</tr>
</tbody>
</table>

Others: Survey paper [S1] and viewpoint papers [S9][S20][S23][S25][S26] 6 0.158

Table 7 classifies the set of 38 papers in the PS into 12 testing topics. Following the same classification scheme, Table 8 classifies the set of 13 papers in the VS into seven testing topics, in which six testing topics were identified in Phase 1 while one new topic was explored. Note also that, apart from the new topic unit testing studied by a paper in the VS in the SaaS layer, there were also some PS papers studying new topic-layer combinations, namely, fuzzing in the PaaS layer and integration testing in the IaaS layer (with paper counts shown in bold italics in Table 8) that had not been explored in the PS.

In Phase 1, with 12 identified software testing topics and three cloud layers, there should be in theory 36 topic-layer combinations potentially to be studied. As Table 7 shows, however, we only found papers that studied 16 of the 36 combinations, with 20 of them unexplored. In Phase 2, as one more software testing topic was identified, the number of topic-layer combinations increased to 39. As shown in Table 8, all the papers in the PVS studied only 19 combinations, leaving 20 combinations unexplored. Note that, apart from the new topic unit testing studied by a paper in the VS in the SaaS layer, there were also some VS papers studying new topic-layer combinations, namely, fuzzing in the PaaS layer and integration testing in the IaaS layer (with paper counts shown in bold italics in Table 8) that had not been explored in the PS.

A noticeably high percentage (15.8%) of papers in the PS was of survey and viewpoint types. In the VS, the proportion of survey and viewpoint papers was even higher (23.1%). These survey papers reviewed existing research work on CST interface using classification schemes that were different from ours, while the viewpoint papers reported the opinions of industry practitioners on cloud software testing. Later in Section 3.4.7, we will summarize all these papers and compare the survey papers with our work.
To answer the Why RQ, we actually examined the motivations to research on CST interface as documented in all the papers in the PVS. We summarized our findings and classified them into two broad categories: (1) applying existing techniques to test cloud components and cloud properties, and (2) enhancing existing testing techniques by the integration of cloud computing capabilities.

(1) Cloud: Program under test

Work in this category essentially perceived a cloud platform as a Program Under Test (PUT) and adopted existing techniques to test a cloud component or the cloud platform as a whole.

In the IaaS layer, a cloud platform provides virtual machines for tenants to deploy and execute their own software that are originally executed on physical machines. Hence, a testing issue is to ensure that the virtual machines can provide the behavior that simulates their physical counterparts when running the software applications on them [S19]. When testing the integration of the deployed software and cloud components, the traditional way of simulating cloud components within virtual machines would incur high costs as the number of cloud components grows. To address this challenge, Versteeg et al. [S48] built for each cloud component a lightweight model of only the interacted parts.

In the PaaS layer, the provided cloud services (such as the MapReduce framework) present the deployed software with certain generic criteria (such as nondeterminism in [S7]) against which the testing process needs to specifically assure.

In the SaaS layer, the integration of existing cloud services is an important part of building cloud software. Some existing work such as [S17][S18][S31][S33] investigated the issues that might arise from the interaction of cloud services. Cloud services may update their registered interfaces, which may affect the behavior of the client applications that use these interfaces.

Some other work in the SaaS layer focused on regression testing [S6][S14] to address the dynamic changes of cloud service interfaces. The states of cloud components may affect the behavior of deployed applications via the cloud interfaces. Some program paths of the applications are traversed only at some specific cloud states. To cover these paths, Zhang et al. [S51] proposed to construct stubs of cloud components to simulate their interface. The stub models could also support testing on a real cloud platform by generating an invocation sequence on the cloud platform giving rise to equivalent cloud states. Security risks to the cloud may come from the outside due to malicious invocations of cloud public interfaces. Zech et al. [S50] applied attack patterns to a model of public interfaces of the cloud under test to generate test cases.

To speed up testing under different levels of workloads, components to automate the generation of testing configurations to speed up the testing under different levels of workloads.

![Table 8](image)

<table>
<thead>
<tr>
<th>Topic ID</th>
<th>Software testing topic (Cloud service architectural layer: [references])</th>
<th>No. of papers studying cloud layer</th>
<th>Proportion of papers in the VS</th>
<th>Proportion of papers in the PVS</th>
<th>Summary of main research ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fuzzing (PaaS: [S43][S44])</td>
<td>* 2</td>
<td>0.154</td>
<td>0.059</td>
<td>Speed up the execution of a test suite by simultaneously running several instances of the software under test on different cloud nodes.</td>
</tr>
<tr>
<td>2</td>
<td>Robustness testing</td>
<td>*</td>
<td>0.098</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Concurrency testing</td>
<td>*</td>
<td>0.020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Performance testing (PaaS: [S42][S47] [S49])</td>
<td>*3</td>
<td>0.231</td>
<td>0.137</td>
<td>Modularize the dependency complexities among distributed software components to automate the generation of testing configurations to speed up the testing under different levels of workloads.</td>
</tr>
<tr>
<td>5</td>
<td>Testing strategy</td>
<td>*</td>
<td>0.059</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Context sensitivity</td>
<td>*</td>
<td>0.039</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Testing parallelization (SaaS: [S39])</td>
<td>*1</td>
<td>0.077</td>
<td>0.137</td>
<td>Build lightweight cloud components to support integration testing of deployed software when the number of interacted cloud components is very large.</td>
</tr>
<tr>
<td>8</td>
<td>Integration testing (IaaS: [S48])</td>
<td>1</td>
<td>0.077</td>
<td>0.098</td>
<td>Build lightweight cloud components to support integration testing of deployed software when the number of interacted cloud components is very large.</td>
</tr>
<tr>
<td>9</td>
<td>Regression testing (SaaS: [S40])</td>
<td>*1</td>
<td>0.077</td>
<td>0.059</td>
<td>Apply cloud services (such as BigTable and MapReduce) to scale up the data storage and parallelize the regression testing executions.</td>
</tr>
<tr>
<td>10</td>
<td>Security testing (SaaS: [S50])</td>
<td>*1</td>
<td>0.077</td>
<td>0.059</td>
<td>Identify vulnerability points of a cloud computing environment by invoking the cloud public interface with malicious inputs.</td>
</tr>
<tr>
<td>11</td>
<td>Migration testing</td>
<td>*</td>
<td>0.020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Log analysis</td>
<td>*</td>
<td>0.020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 (new)</td>
<td>Unit testing (SaaS: [S51])</td>
<td>1</td>
<td>0.077</td>
<td>0.020</td>
<td>Build a simulated cloud environment that meets the cloud interface specifications to create cloud states for covering specific paths of the application units under test.</td>
</tr>
</tbody>
</table>

Note: Each cell decorated by an asterisk (*) represents a topic-layer combination studied by at least one paper in the PS.
Cloud computing provides the deployed software services with many useful properties (such as elastic resource provision and flexible service access) while encountering new challenges (such as security). Existing testing techniques could be enhanced to take advantage of cloud properties to address these new challenges. For example, robustness testing aims at validating software performance against various failure scenarios, which often incur high costs. Some existing work [S2][S11][S12][S13][S16] utilized the cloud to simulate different failure scenario combinations to test the deployed software. Performance testing against various workloads also incurs high cost due to workload generation. Some existing work [S10][S24][S28][S32] utilized the cloud to explore the optimized workload generation strategy to meet performance testing requirements. Some other work [S3][S5][S22][S30][S34][S38] proposed to parallelize the testing execution traces into multiple cloud nodes simultaneously so as to speed up the testing process. Another interesting issue is service migration between two deployed platforms [S8]. Software deployed on a cloud platform may use the public interfaces provided by other cloud platforms, which may increase their vulnerability to security. Some existing work [S29][S37] focused on testing these security problems. Some work [S21] attempted to scale up the analysis of large log files generated and utilized in the testing process. A major motivation of using cloud properties was to improve testing performance, typically by either parallelizing the execution of multiple testing instances simultaneously [S43][S44] or by distributing partitions of the same test execution into different cloud nodes [S39]. For example, some existing cloud services (such as BigTable and MapReduce) were designed to handle large datasets and to parallelize computations. Thus, Huang et al. [S40] implemented their regression testing technique with cloud services in order to speed up its performance. Computing cloud could also act as a platform to accelerate performance testing activities executed under different scenarios [S42][S47][S49].

[Assessment] More than half of all the topic-layer combinations in both Table 7 and Table 8 were not studied in any paper in the PS and the VS, respectively. There is a surprisingly high ratio of survey and viewpoint papers, which suggests that much research opportunity on CST interface remained to be explored, and many aspects could be rich (but unclear) enough to warrant further investigations. We also found that 36.8% of the papers in the PS as well as 49.0% of the papers in the PVS exploring the same testing topics in more than one cloud layer. This finding is in line with conjecture C2 that papers on multiple cloud layers address issues in the same testing topics. The above results are summarized in the Why sections of Table 1 and Table 2.  

3.4.3. The What dimension: Analysis  
The What dimension seeks to identify the kinds of research ideas in existing work for addressing the challenges in different cloud layers. Table 7 shows that each of Topics 2, 4, 7, and 8 (namely, robustness testing, performance testing, testing parallelization, and integration testing, respectively) contributed to at least 10% of the papers in the PS, and these four topics together contributed to half of the PS papers. Topics 4, 7, and 8 were also studied in papers in the VS, as shown in Table 8. When considering all the papers in the PVS together, the former four topics (2, 4, 7, and 8) remained to be the most popular ones in CST-interface research, each contributing to at least 9.8% of the papers in the PVS. In the IaaS layer, the idea of fault-based testing was researched in the generation of faulty virtual machines [S19] and various failure scenarios [S2][S12][S13] by papers in the PS. Papers in the VS proposed to build a lightweight model of cloud components to speed up integration testing for dealing with a large number of integrated cloud components [S48]. In the PaaS layer, the idea of fault-based testing was also researched via the generation of multiple-failure scenarios for application testing execution [S11] by papers in the PS. Papers in the PS also developed techniques for testing SaaS applications to deal with failure-simulation issues [S16], nondeterminism issues [S7], or different invocation sequences of cloud service APIs [S15] in the PaaS layer. Methodology for benchmarking [S24] and strategies to lower testing costs by exploiting the elastic property of PaaS [S27] were also investigated. On the other hand, papers in the VS aimed at (1) speeding up testing by creating multiple testing instances in cloud nodes and executing them simultaneously [S43][S44], (2) generating different performance scenarios (workload patterns) to test application behavior [S47][S49], or (3) developing a methodology to handle component dependencies [S42]. In contrast, even though the majority (51%) of the papers in the PVS focused on testing challenges in SaaS, fault-based testing work were not studied in the SaaS layer by any paper in the PVS. Papers in the PS investigated a diverse set of ideas that include performance measurement [S32], metrics across different cloud service architectural layers [S10], usage-specific [S4] or general models of TauS [S35], testing task management [S36], test parallelization (either by parallelizing the testing execution states [S3][S5][S30] or by utilizing the existing cloud services and architecture [S22][S34][S38]), optimizing the testing of web service composition [S31][S33], improving controllability and observability [S17][S18], reducing costs of testing [S28], identifying software changes for regression testing [S6][S14], detection of vulnerability faults [S29][S37], migration testing of applications from physical to virtualized platform [S8], and the analysis of huge data in log files [S21]. On the other hand, papers in the VS studied (1) the partitioning of a test execution process and distributing these partitions to achieve statically-balanced workloads [S39], (2) ways to accelerate regression testing through the support of a distributed cloud computation framework [S40], (3) ways to identify vulnerability points from outside of a cloud to test the risks of cloud platforms being attacked by malicious users [S50], and (4) how to build stubs of cloud components to improve the structural coverage of unit testing, such as covering program paths of the deployed application unit with more diverse cloud states [S51].  

[Assessment] While there was only one common research idea (namely, fault-based testing) across multiple cloud layers in Phase 1, none was found in Phase 2. Thus, testing research ideas varied widely across different cloud layers, which was consistent with conjecture C3. In the lower cloud layers, the research ideas mainly involved either randomized or fault-based testing, and many other systematic testing approaches were not explored. The above results are summarized in the What sections of Table 1 and Table 2.
3.4.4. The Where dimension: Analysis

Regarding the Where dimension, we studied two aspects: (1) the cloud layers in which the papers were more focused, and (2) the publication venues of the papers.

For the first aspect, from Table 7 and Table 8, the three layers IaaS, PaaS, and SaaS, respectively, accounted for two, five, and nine topics identified in Phase 1, and 3, 6, and 10 topics identified in the combined two phases. The three layers were studied by 5, 11, 26 papers, respectively. SaaS was thus the layer most studied by the papers published in the combined period (2010–2012) of the two phases.

In terms of proportions, 9.8%, 21.6%, and 51.0% of the papers in the PVS addressed issues in the IaaS, PaaS, and SaaS layers, respectively. The rest (17.6%) were survey or viewpoint papers. As such, we could not find literature evidence to show that testing in the IaaS and PaaS layers received more attention than that in the SaaS layer.

For the second aspect, Table 9 and Table 10 summarize the distributions of papers in the PS and the VS, respectively, which appeared in different types of publication venues. Table 9 shows that 7.9%, 78.9%, and 13.2% of the papers in the PS were disseminated in journals, conferences, and workshops, respectively. For comparison purpose, we also computed the statistics of papers in IS-Scopus and found that 77.4% and 22.6% of them were published in proceedings (of conferences and workshops) and journals, respectively. Thus, the PS had a much smaller percentage of journal papers than IS-Scopus (7.9% versus 22.6%). That is, a relatively high percentage of journal papers in IS-Scopus were eliminated by our exclusion criteria (EC1–EC3) as irrelevant to CST interface. In other words, papers on CST interface were more likely than those on other topics to be published in conference/workshop proceedings than journals.

Table 10 shows a similarly biased distribution, in which 23.1%, 61.5%, and 15.4% of the papers in the VS were disseminated in journals, conferences, and workshops, respectively. Similar to what was observed in Phase 1, more than half of the papers were published in proceedings of conferences and workshops.

[Assessment] There seemed to be a tendency for more testing papers to address challenges in an upper layer than a lower layer. When considering the PVS as a whole, papers that studied SaaS issues still outnumbered those studying PaaS and IaaS by a wide margin. Thus, even though there might be a slight upward trend of research attention in the lower layers, the attention in SaaS was still dominant. Therefore, our findings did not support the first part of conjecture C4 regarding research attention to different cloud layers. In terms of publication venues, the findings were consistent with conjecture C4 that the majority of recent papers are published as research articles in conference/workshop proceedings with a good presence of journal papers. The above results are summarized in the Where sections of Table 1 and Table 2.

3.4.5. The When dimension: Analysis

The When dimension concerns with when the research work was published and cited by other papers.

To reduce verbosity, we refer to a paper published in year \( y \) as a year-\( y \) paper. Also, when a paper \( P \) cites a paper \( Q \), we call \( P \) the citing paper and \( Q \) the cited paper, and we say that \( Q \) receives a citation (from \( P \)) and there is a citation relationship between \( P \) and \( Q \).

<table>
<thead>
<tr>
<th>Table 9. Publication venues and types of the PS papers (Phase 1).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of</strong></td>
</tr>
<tr>
<td><strong>research</strong></td>
</tr>
<tr>
<td><strong>articles</strong></td>
</tr>
<tr>
<td><strong>Journal</strong></td>
</tr>
<tr>
<td><strong>Proceedings</strong></td>
</tr>
<tr>
<td><strong>Conference</strong></td>
</tr>
<tr>
<td><strong>Workshop</strong></td>
</tr>
<tr>
<td><strong>Total</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 10. Publication venues and types of the VS papers (Phase 2).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of</strong></td>
</tr>
<tr>
<td><strong>research</strong></td>
</tr>
<tr>
<td><strong>articles</strong></td>
</tr>
<tr>
<td><strong>Journal</strong></td>
</tr>
<tr>
<td><strong>Proceedings</strong></td>
</tr>
<tr>
<td><strong>Conference</strong></td>
</tr>
<tr>
<td><strong>Workshop</strong></td>
</tr>
<tr>
<td><strong>Total</strong></td>
</tr>
</tbody>
</table>

Table 11 and Table 12 summarize the paper citation counts by publication years and by paper sets, respectively. In Table 11, the number in the intersection of each row year \( A \) and column year \( B \) represents the number of year-\( B \) papers receiving citations from at least one year-\( A \) paper. The bottom row of Table 11 presents the numbers of papers published in the respective year columns. Similarly, the rightmost column of Table 12 shows the numbers of publications in the respective paper set rows. In the same table, the number in the intersection of each (PS or PVS) paper set row and each year \( B \) column represents the number of year-\( B \) papers receiving citations from at least one paper in the paper set.

In Table 11, we see that among the 16 papers published in 2010, seven of them were cited by year-2010 papers. In Table 12, we find that the cumulative count of year-2010 papers being cited increased from seven in 2010 to 11 in 2011 and 15 in 2012 as more of the year-2010 papers received citations by year-2011 and year-2012 papers, respectively. Hence, the proportion of year-2010 papers that were cited increased from 43.8% (= 7/16) in 2010 to 68.8% (= 11/16) in 2011 and 93.8% (= 15/16) in 2012. That is, within just two years of publication, 93.8% of all the year-2010 papers received citations from other papers, showing that they were very quickly referenced by researchers in the community.

In the column 2011 of Table 11, none of the 22 papers published in 2011 was cited by those published in 2010, which was of no surprise. Four papers published in 2011 were cited by those published in the same year. Two of these four cited papers were further cited by papers published in 2012, while five other year-2011 papers were cited by those published in 2012. Thus, within the entire three-year period of 2010–2012, nine (40.9%) of the 22 papers published in 2011 were referenced by other papers (see the intersection of the PVS row and the 2011 column of Table 12).
Table 11. Paper citation counts by publication years.

<table>
<thead>
<tr>
<th>Paper set</th>
<th>Publication year of citing paper</th>
<th>Number of cited papers (cited by papers published in the row year) that were published in</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2010</td>
</tr>
<tr>
<td>PVS</td>
<td>2010</td>
<td>7</td>
</tr>
<tr>
<td>PS</td>
<td>2011</td>
<td>11</td>
</tr>
<tr>
<td>VS</td>
<td>2012</td>
<td>15</td>
</tr>
<tr>
<td>Total number of publications in the year</td>
<td>16</td>
<td>22</td>
</tr>
</tbody>
</table>

Table 12. Paper citation counts by paper sets.

<table>
<thead>
<tr>
<th>Citing paper set (Publication period)</th>
<th>Number of cited papers (cited by papers in the paper set row) that were published in</th>
<th>Number of publications in the paper set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2010</td>
<td>2011</td>
</tr>
<tr>
<td>PS (2010–2011)</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>PVS (2010–2012)</td>
<td>15</td>
<td>9</td>
</tr>
</tbody>
</table>

For the 13 papers published in 2012, we found (again unsurprisingly) that none of them was cited by either those published in 2010 or those published in 2011. Only one of the 13 papers in the VS was cited by those published in the same year 2012.

With the data in Table 12, considering the PS paper set surveyed in Phase 1, a total of 15 = 11 + 4 (or 39.5%) of its 38 papers were cited by papers in the PS. Considering the PVS as whole, a total of 25 = 15 + 9 + 1 (or 49.0%) of its 51 papers were cited by other PVS papers.

We also counted the number of citation relationships per paper. There were 13 citation relationships among year-2010 papers, 49 among the PS papers and 96 among the PVS papers, respectively. Thus, during the periods of one year (2010), two years (2010–2011), and three years (2010–2012), the numbers of citation relationships per paper were 0.81 (= 13/16), 1.29 (= 49/38), and 1.88 (= 96/51), respectively. These ratios indicated that papers on CST interface received progressively increasing attention during the survey period.

[Assessment] Among all the 16 papers published in 2010, 43.8% of them received citations within the same year. The proportion of year-2010 papers receiving citations went up to 68.8% in 2011 and 93.8% in 2012, showing that they were promptly recognized by researchers. Moreover, 39.5% of the PS papers were cited by other PS papers, while 49.0% of the PVS papers were cited by other PVS papers. Thus, our data were consistent with conjecture C5 that many papers are promptly cited. The above results are summarized in the When sections of Table 1 and Table 2.

3.4.6. The How dimension: Analysis

The How dimension concerns with how the research evolved among various software testing topics and cloud layers. We conducted a refined analysis of the citation relationships received by each topic-layer combination in each phase, as depicted in Figure 2 and Figure 3, respectively.

In Figure 2, the two “boxes” with the labels “2010: 16 papers” and “2011: 22 papers” depict the data for year-2010 and year-2011 papers. We will call them the year-2010 box and year-2011 box, respectively. Papers addressing testing topics in different cloud layers are shown in different shapes, namely, oval, rectangle, and stadium (that is, rectangle with two oval sides) for SaaS, PaaS, and IaaS, respectively. Inside each shape is a list of references of papers on the same topic in the same layer. For instance, the text “S19” inside the stadium shape near the top right corner of the year-2010 box refers to Martignoni et al. [S19], which dealt with the fuzzing topic in the IaaS layer.

When the referenced paper in a shape is published by researchers who have authored more than one paper, we outline the shape with a double border (see, for example, [S4], which was published in 2010 and studied the topic context sensitivity in the SaaS layer). Otherwise, we outline the shape with a single border (see, for instance, [S19], whose authors have published only one paper in the PS). A directed edge from a shape $x$ to another shape $y$ indicates that the papers referenced in $x$ collectively cited the papers referenced in $y$. When the number of citations in the PS made by the papers in $x$ exceeds one, the number is labeled at the left side of $x$. Similarly, if the number of citations in the PS received by the papers in $y$ exceeds one, the number is labeled at the right side of $y$. Survey and viewpoint papers are excluded from our analysis here and their citation numbers are omitted from the figure, as by nature they tend to cite many papers.

(1) Analysis in Phase 1

Number of citations received. The number of citations received by (the papers of) a topic-layer combination reflects its importance to the research development of the subject. In Phase 1, the topic-layer combination robustness testing in the IaaS layer received the most (7) citations, followed by testing parallelization in the SaaS layer (which received five citations) and integration testing in the SaaS layer (which also received five citations). These three topic-layer combinations already accounted for 17 = 7 + 5 + 5 (or 73.9%) of all the 23 citation relationships among the PS papers and, hence, were the most cited ones in CST interface research during the period 2010–2011. In contrast, each other topic-layer combination received at most two citations. In particular, fuzzing in the IaaS layer, migration testing in the SaaS layer, and log analysis in the SaaS layer had no citation relationship with others.

Citation relationships by layer. C6 conjectures that many papers on various software testing topics and cloud service architectural layers are interreferenced to evolve CST-interface research. The idea is that it is certainly not surprising to find a paper citing other papers that addressed the same topic or issues in the same layer. In contrast, it would be much more interesting when a citation involves interreferencing among papers across different topics or different layers, which is more likely to provoke new ideas that help evolve the research to maturity. To assess conjecture C6, we classified each citation relationship according to whether the testing topics and cloud layers addressed by the two papers in the relationship are the same or different.
Table 13 shows the distribution of the citation relationships among papers in the PS across the three cloud layers found in Phase 1. For instance, the number 2 at the intersection of the SaaS row and the IaaS column means there were two citation relationships in the PS such that the citing paper was concerned with testing issues in the SaaS layer while the cited paper was in the IaaS layer.

From Table 13, we find that there were 10, 1, and 3 intraSaaS, intraPaaS, and intraIaaS citation relationships among papers in the PS, respectively, giving a total of 14 intralayer citation relationships, which outnumbered the other nine (or 39.1%) interlayer ones out of a total of 23. In other words, citations among papers in the same layer were much more common than those across different layers. In particular, the bottom row of Table 13 shows that the total (and percentage) of citations received by papers in the SaaS, PaaS, and IaaS layers were 15 (65.2%), 1 (4.3%), and 7 (30.4%), respectively. Evidently, papers in the SaaS layer were cited far more often than those in the PaaS and IaaS layers.

<table>
<thead>
<tr>
<th>Citing papers at layer</th>
<th>Number of cited papers at layer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SaaS</td>
</tr>
<tr>
<td>SaaS</td>
<td>10</td>
</tr>
<tr>
<td>PaaS</td>
<td>4</td>
</tr>
<tr>
<td>IaaS</td>
<td>1</td>
</tr>
<tr>
<td><strong>Subtotal</strong></td>
<td><strong>15</strong></td>
</tr>
<tr>
<td>(Percentage of all citations)</td>
<td>(65.2%)</td>
</tr>
</tbody>
</table>

Figure 2. Citation relationships of the PS papers across years, testing topics, and cloud service layers (Phase 1).
**Interreferencing.** We further counted the number of citation relationships among papers of the same and different topics in the same and different layers. We found seven relationships among papers of the same topic and layer, none involving the same topic in different layers, seven addressing different topics in the same layer, and nine (or 39.1%) of the total of 23 citation relationships were concerned with interreferences across different topics in different layers. The last percentage was not high, indicating that interreferencing was not a mainstream practice. Thus, we did not find adequate evidence to support conjecture C6 in Phase 1.

(2) **Analysis in Phase 2**

In Phase 2, we extended the analysis from citation relationships among papers in the PS to all citation relationships among papers in the PVS. Using the same legend as in Figure 2, the extended analysis results are depicted in Figure 3, in which the focus of analysis was on the rightmost (year-2012) box. Note that there are new directed edges from the year-2012 box to the two (year-2010 and year-2011) boxes on the left (which were shown in Figure 2 before). Specifically, there were 11 citation relationships from the VS papers to the PS papers.

![Figure 3. Citation relationships of the PVS papers across years, testing topics, and cloud service layers (Phase 2).](image-url)
We observed that two topic-layer combinations (integration testing in the IaaS layer [S48] and unit testing in the SaaS layer [S51]) in the year-2012 box were "isolated", having no citation relationships with any other papers in the PVS. Two of the other five topic-layer combinations (testing parallelization in the SaaS layer [S39] and regression testing in the SaaS layer [S40]) in the year-2012 box had cross-year citation relationships with papers sharing the same topic in the same cloud layer. One combination (security testing in the SaaS layer [S50]) in the year-2012 box had cross-year citations associated with other topics in the same cloud layer. The remaining two topic-layer combinations (fuzzing in the PaaS layer [S43] and performance testing in the PaaS layer [S42][S49]) contributed to five cross-year citation relationships with papers on different testing topics in different cloud layers. Finally, two papers [S44][S47] cited no other papers in the PVS.

Overall, within the entire period covered by our case study, there were a total of 34 (= 23 + 11) citation relationships among papers in the PVS. Only 14 = 9 + 5 (or 41.1%) of all these citation relationships were among papers of different testing topics across different cloud layers. Hence, interreferencing among different testing topics and across different cloud layers was still not very common in Phase 2.

[Assessment] In Phase 1, we found that the distribution of citation relationships among the PS papers in different cloud layers was rather uneven. Most citations were related to testing topics in the SaaS layer while very few citations were made in the other two layers. The proportion of citations across different testing topics in different cloud layers was only 39.1%. Thus, interreferencing across different topics in different layers had not been the mainstream practice. Our findings in Phase 1 did not provide evidence to support conjecture C6.

In Phase 2, among the seven topic-layer combinations studied by the VS papers, two had no citation relationships with other papers at all. In fact, less than half of all the citation relationships from the VS papers were made to different testing topics across different cloud layers. When considering all the papers in the PVS, citations across different testing topics and layers accounted only to 41.1%, a very small increase from the percentage (39.1%) found in Phase 1. Again, interreferencing among different topics and layers had not been the mainstream practice by the year 2012. Thus, our data in both of the two phases did not support conjecture C6. The above results are summarized in the How sections of Table 1 and Table 2.

3.4.7. Survey and viewpoint papers

Both the PS and VS paper sets contained some papers that were survey in nature [S1][S41][S45] or presented the viewpoints of the authors or their interviewees [S9][S20][S23][S25][S26][S46]. Moreover, toward the end of Section 3.3, we mentioned two viewpoint papers [11][17] on CST interface identified from Google Scholar. We concisely review these papers in this section.

(1) Survey papers

Bai et al. [S1] classified existing cloud software testing tools into four categories: (1) simulations, (2) service mocking, (3) test job parallelization, and (4) environment virtualization. The work in category (1) attempted to simulate testing conditions (such as failure scenarios [S2][S12]) in a cloud. The work in category (2) addressed the dependency issues between cloud-based software and its external parties. The work in category (3), such as [S5][S22], aimed at improving the efficiency and reducing the cost of the testing process. The work in category (4) constructed a virtualized test environment and conditions for domain-specific software testing.

Bai et al. [S1] also proposed three different perspectives (namely, testing objectives, testing activities, and tool architecture) to classify 13 representative cloud software testing tools. The perspective of testing objective considered the purpose of applying the testing tools (such as performance testing and fault-tolerant testing). The perspective of testing activities considered how the tools support different activities of the testing process (such as test resource management and parallel testing execution). The perspective of tool architecture considered whether the tools were built on a cloud platform and scalable.

Incki et al. [S41] surveyed the cloud software testing topic from the perspective of software testing as a service, which means either testing cloud-resident applications or providing testing software as a service in the cloud. They classified their collected papers into two categories: one according to the type of application (such as mobile applications or cloud applications), and the other according to the characteristics of the provided testing service (including testing level, testing type, contributions to the testing process, and the cloud delivery model). Regarding the testing level, they found that system testing and acceptance testing had received the most and the least research attention, respectively. For the testing type, they found that functional testing and security testing had received the most and the least research attention, respectively. For contributions to the testing process, they found that test execution automation, test case generation, test framework, and test evaluation received comparable degrees of attention. For the cloud delivery model, they found that the SaaS layer received the most research attention while the IaaS layer received the least attention, which was consistent with our findings.

Priyanka et al. [S45] followed the guidelines of [25] to report an MS on cloud-based software testing techniques. They proposed an RQ for each of five aspects: (1) the motivation of migrating testing to the cloud, (2) applications suitable for online testing, (3) the proposed testing approaches, (4) testing as a service, and (5) cloud testing providers. They collected 82 research papers by using search keywords that were different from ours in five databases without specifying a limit to the paper publication time. They classified the papers into four categories: cloud based testing (38 papers), automated test case generation (9 papers), testing frameworks (23 papers), and cloud application development (12 papers). Similar to what we did in our case study, their work also investigated the distributions of publication time and venues. They concluded that cloud-based software testing was hot but still immature. However, Priyanka et al. [S45] did not explicitly answer the RQs put forward in their paper, nor did they document the methodology of classifying their 82 papers. As such, we could not further compare their survey results with ours.

(2) Viewpoint papers

Fang and Xiong [S9] considered three aspects to be very important to cloud software testing: (1) ubiquitous test middleware to manage cloud resources and testing tasks, (2) multiple test scenario convergence, and (3) co-simulation of different components. Other authors [S23][S25][S26] opined that testing tasks with little interdependency would be suitable for execution in the cloud.
Mohammad and Mcheick [S20] considered that the lack of a test environment rendered it difficult to compare different service composition methods. They outlined the differences among traditional testing, service-oriented testing, and cloud service testing, and posed several open questions for further study.

Parveen and Tilley [S23] argued that applications suitable for testing in the cloud should have three characteristics: (1) the test cases are independent of one another, (2) a self-contained and identifiable operational environment, and (3) a programmatically interface suitable for automated testing. They also argued that the computing cloud could greatly benefit the following types of testing: unit testing, high volume automated testing, and performance testing under different circumstances.

Riungu et al. [S25][S26] summarized the issues mentioned by 11 practitioners and came up with two main conclusions. First, using a computing cloud would benefit testing tasks that are loosely related (such as independent test cases in unit testing, high-volume automated testing, and performance testing). Second, cloud software testing management (such as user contexts and test process management) was an important issue for the use of cloud software testing.

Kalliosaari et al. [S46] interviewed 15 organizations in the industry and selected eight of them to study their adoption of cloud software testing. They analyzed the interview comments in terms of three aspects of the effect of cloud computing on software testing. On the effectiveness aspect, they found that cloud computing offered pay-for-use testing resources with flexible configurations in parallel, which reduced testing cost, shortened testing time, and obtained more realistic test results. On the second aspect, they found that testing service delivery and support had been enhanced with flexible access to testing tools, better developer-tester communication, and faster regression testing of new requirements. The third aspect was on the challenges that prevented the adoption of cloud for testing. The interoperability among different clouds and security-related issues were the top two challenges discussed at the interviews. Finally, they proposed a roadmap for industry organizations to deploy their testing businesses to the cloud.

Chhabra et al. [11] urged for more research effort to six software engineering aspects. Three of them were taken from the web services domain: service selection, testing criteria for cloud interactions, and the availability of source code in third-party software components. From the testing perspective, these three aspects were explored by Tsi et al. [S31], Chan et al. [S8], and King et al. [S17][S18], respectively. The other three aspects were due to the distributed nature of nodes in a computing cloud: a query-oriented programming pattern (such as MapReduce), a distributed software execution model, and a loosely-coupled design to deal with issues in cloud integration. The paper [11] was affiliated with India and cited by one paper [S45] in our case study.

Gao et al. [17] classified cloud software testing into seven categories: (1) service functional testing, (2) integration testing, (3) API and connectivity testing, (4) performance and scalability testing, (5) security testing, (6) interoperability and compatibility testing, and (7) regression testing. Their work did not explore the dimension of cloud layers as what we have done in our MS. Some of their categories, however, are common with our identified topics: integration testing, performance testing, security testing, and regression testing. This work [17] was the result of a collaboration among researchers from USA and China, and was cited by two papers [S1][S10] in the PS and one paper [S45] in the VS.

[Assessment] Survey and viewpoint papers together constituted 15.8% and 23.1% of the papers in the PS and the VS, respectively. These proportions were quite high for a new research area. It would be interesting to compare the three survey papers on cloud software testing [S1][S41][S45] with our work. Viewed from the 5W+1H model, the paper [S1] considered existing cloud software testing tools from three dimensions (Why, What, and Where) only whereas the other two papers [S41][S45] considered only two dimensions (What and Where) and five dimensions (Who, Why, What, Where, and When), respectively. Thus, every survey paper missed certain dimensions. Thus, our proposal of using the 5W+1H pattern to structure RQs and report MS results can guide researchers to generate a more comprehensive set of perspectives for reviewing a research topic.

3.5 Threats to validity

Construct threats to validity. We measured the relationships among the papers by the number of citations. Other metrics might reveal a different picture of relationships. We used our expert judgment to exclude papers that were considered irrelevant to the case study. We minimized this threat across different investigators. However, given the same paper, an outside investigator may hold a different view from ours in deciding whether the paper should be excluded from the final paper set.

Internal threats to validity. This study used only three databases for paper search in Phase 1 and then the largest one among these three databases in Phase 2. Recently, there are other open-access paper archival repositories. Including them in the study might introduce other articles and, hence, reveal different paper statistics. Moreover, there was invariably a time delay for the inclusion of papers in the databases. For instance, the proceedings of a conference held in December 2012 might not have appeared in the databases at the time of paper extraction in our case study. To address this threat, one might conduct several rounds of paper extractions until the dataset is stable, but this would take an indefinite amount of time and much more effort to track the evolution.

In this work, we performed two phases of search for papers published in two different time periods and could validate the consistency of findings between the two phases.

In this study, we checked the consistency only among ourselves in classifying the collected papers into testing topics and cloud layers. Other researchers may classify the papers differently. However, considering the small number of papers with conflicting classifications among ourselves, we argue that our findings were robust enough to possibly different classifications. Similarly, assessing the relevance of the papers to CST interface could be prone to bias or errors. To ensure impartiality and accuracy, the task was performed independently by more than one of us. Our chosen keywords and their variants certainly influenced the set of articles generated for analysis. Our focus was software testing for cloud computing. Initially, we extended the concept of testing to include “analysis” in formulating the search keyword, although on completion of Phase 1 we found it actually unnecessary. We did not include “software” in the keyword because, to our knowledge, a paper might not mention “software” or “program” in its abstract. This omission might have introduced ambiguity. To mitigate this risk, we read the included papers to ensure their true relevance to software testing. In Phase 1, we did not loosen our criteria on the term “cloud” because at that time we could not identify what other terms might have been used with a meaning similar to “cloud”.

CityU-CS-LOSEM-TR-1403 Page 19
After gaining the experience from Phase 1, we included additional keywords “IaaS”, “PaaS”, “SaaS”, and “TaaS” in Phase 2. We had once considered the terms “utility” or “grid”, but they were usually used to refer to non-cloud articles (such as papers on grid-based application testing) that were related to some classical topics rather than the emerging topic of our interest. Finally, we only included papers that had been peer-reviewed. As such, other work like technical reports, white papers, or patents might have been excluded.

**External threats to validity.** Our results are not necessarily generalizable to other topics or the use of alternative inclusion and exclusion criteria. To generalize the 5W+1H pattern, one may extend the questions in each dimension. It would be interesting to compare our findings with further work that reviews research papers their research objectives (Moreover, researchers have to understand from the collected such basic data. It would be interesting to compare our findings with further work that reviews research challenges in other topics.

4. Experience in applying the 5W+1H pattern to a mapping study on CST interface

In this section, we share our experiences with readers who may have the need to report MSs of other research topics. Section 4.1 discusses our understanding of mapping the six dimensions to report an MS. Section 4.2 discusses our recommendations when adopting the mappings.

4.1. Enhancing an MS with the six dimensions of the 5W+1H pattern

This section presents our experience in mapping the six dimensions of the 5W+1H pattern to an MS of a research topic. We adopted a particular orientation in mapping these six dimensions in our case study. We had not been aware of any other possible mappings until we completed this study. Section 4.1.1 presents the direct mapping of the six dimensions. Section 4.1.2 discusses the extension of the direct mapping in a hierarchical manner. Finally, Section 4.1.3 discusses the relationships and integration of individual dimensions. Finally, Section 4.1.4 suggests ways to enhance the cost-effectiveness of performing an MS.

4.1.1. Six dimensions to structure a mapping study

Our view is that researchers who are new to a particular research topic can greatly benefit by being informed of the representative researchers (Who) and important publication venues (Where) so that they can quickly assess the existing research progress (When) of the topic. It is not difficult to gather this basic information by collecting a set of published papers on the topic and extracting such basic data.

Moreover, researchers have to understand from the collected papers their research objectives (Why), research problems (What), proposed solutions, and relationships with other work in the same area (How). Simply browsing a paper from these three dimensions already helps the researcher appreciate the rationales of the research topic, relate different work, and trace their progress. However, this task is very time-consuming and it is nontrivial to obtain clear and consistent results. It is imperative to have several researchers working together to avoid individual bias and to resolve any inconsistencies among different judgments of the same paper.

The *Who* dimension in journalism seeks to identify the actors involved with news events. When studying a research topic, this dimension naturally suggests identifying the involved researchers and their research groups. We adopted this mapping in our case study of the MS of CST interface. However, we believe that the *Who* dimension may also guide us to explore more information than just the basics of individual researchers. This is because new research work is usually built from existing ones, whereas a news event may occur without necessarily bearing any relationship to other news events (unless it is a developing story that lasts for a period of time, such as a presidential election). Thus, we can trace the research relationships by viewing the references of a paper from the perspective of exploring relationships among different researchers. By doing so, given a set of papers, we can build a researcher citation relationship network to connect different researchers together. (Indeed, we have seen social or professional network sites [35] developed for similar purposes as well.) Different researchers in the network may work on various topics with different backgrounds. With the understanding of individual researchers and their relationships around a research topic, we may further study how these researchers identify research problems of the same topic from different perspectives. The understanding of citation relationships around a research topic may shed light on how scholars are inspired about the same topic and consider how a new piece of work fits into the context of existing ones. We believe that inexperienced researchers can benefit a lot by learning the research methodologies of others.

The *Why* dimension in journalism is meant to reveal the motivations behind news events. In an MS, we have interpreted it to mean an investigation of the expected research objectives of each paper and the significance of the identified research challenges. The motivation to drive a new piece of research work often comes from noting or addressing the limitations of existing work. The same limitations of existing work may be addressed by different strategies, approaches, or solutions at different levels. Thus, we can group different pieces of existing work by their motivation and then compare one group with another in order to appreciate the best work in terms of the level of success in resolving the limitations. In our case study, we only explored the research objectives of the papers under the same identified research topics. We did not compare different individual pieces of work to explore their relationships and differences in terms of motivations. We believe that researchers would encounter this problem when their understanding of a new topic is insufficient. Conducting an MS of a topic, in fact, aims at improving the researchers’ understanding of the topic. Thus, we recommend adopting an iterative process to drive the study of the *Why* dimension. Through improvements in the understanding of a topic after several iterations of an MS, researchers can obtain a clearer “motivation map” of the topic.

The *What* dimension in journalism usually aims at reporting the content of news events. In an MS, when reading a research paper, scholars are usually interested in finding answers to three questions: (1) what are the research problems that the paper identifies, (2) what is the innovative idea that the paper utilizes to address the identified problems, and (3) what are the possible limitations of the solutions proposed in the paper? The answers to these three questions, however, are not always explicitly presented in a paper. Researchers need to critically read a paper to search for the answers. This step often consumes the greatest effort. Moreover, to assure the validity of an MS, investigators need to collaborate with one another to resolve the possible differences in understanding the same paper.
The *Where* dimension in journalism mainly intends to report the geographical locations where news events happen. When studying a research topic, this dimension naturally suggests identifying the venues *where* the topic-related literature was published. Some people may argue that a good way of knowing the mainstream publication venues of a research topic is to ask the experienced researchers in the topic domain. From our own experience, most papers on an *emerging* topic, including even some of the influential papers, do not always appear in the mainstream publication venues commonly considered to be the best in the field. For example, the majority of papers reviewed in this study did not appear in top mainstream conferences or journals of the software engineering or services computing domain.

The *Where* dimension can also suggest that the investigator (as what we have done) generate a topic-specific structure to organize the research items (problems, solutions, limitations, and so on) identified by the *What* dimension. Such a structure provides new researchers with an overall understanding of the subtopics within the research topic under study.

The *When* dimension in journalism seeks to report the time when a news event occurs. In an MS, this dimension may suggest gathering data to demonstrate the popularity of a topic to the research community. For example, our case study adopted the number of papers published each year as an indicator of the degree of research attention from the community, but other metrics may also be used for this purpose. However, we argue that the *When* dimension alone may not reveal too much information about a research topic. The reason is that the time of publishing a research study just serves to tag the progress at a specific temporal phase. Hence, in our case study, we integrated the *When* and *How* dimensions to investigate how a paper evolved from previous work.

The *How* dimension in journalism describes the causality relationships among events in news reports. In an MS, this dimension can be taken to reveal the causality relationships among objects identified in the *What*, *Where*, and *When* dimensions. For example, our case study explored the citation relationships among the identified software testing topics, across different cloud service architectural layers, as well as among different years of publication. We believe that the *How* dimension can reveal more findings when viewing the causality relationships among different dimensions of the classification scheme. For example, we classified the collected papers using a two-dimensional scheme involving combinations of software testing topics and cloud service architectural layers, as well as along the temporal axis, as depicted in Figure 3. These dimensions and combinations can provide a useful context to enrich and deepen the understanding of relationships among papers.

### 4.1.2. Hierarchical mapping of the six dimensions

Section 4.1.1 above presents our understanding of directly mapping each dimension to structure RQs and report results of an MS. Such a direct mapping can generate a top-level research map of an investigated topic. However, due to insufficient topic-specific knowledge, the general research map can only provide researchers with an initial understanding rather than clearly identified research problems. From our experience in the case study, we think that the direct mapping can be further enhanced to provide more information for researchers to follow. For example, we identified 12 software testing topics in CST-interface research in Phase 1 of the case study. Software engineering researchers are well aware that there are plenty of testing topics, as software testing is a broad branch of the discipline. However, even with the limited number of identified testing topics, we still found it challenging to pinpoint the solid research problems that would directly motivate further study. It would be even more difficult for services computing researchers who are not familiar with software testing topics. Suppose that we are interested in two particular topics: concurrency testing and testing parallelization (Topics 3 and 7 in Table 7 and Table 8). We noted that the findings in this study could not provide sufficient understanding on the two topics. We analyzed the reason and found that the two topics were not cloud-specific but had been researched for many years on the testing of software executed on desktops and servers. Our case study only focused on exploring the two topics in the cloud domain and ignored the related progress in other domains. Thus, without an informed comparison with the research progress of the two topics from other domains, we could not precisely pinpoint the effect of the cloud computing domain on the two topics.

Therefore, we suggest hierarchically applying the 5W+1H pattern and iteratively investigating a topic. In the first iteration, we may identify some subproblems of the topic. In the next iteration, we may then set the selected subproblems as the objectives of the next-stage MS and repeat the same procedure.

A hierarchical mapping style may provide researchers with more knowledge when viewing a research topic from multiple domains. Take the integration testing topic (Topic 8 in Table 7 and Table 8) in our case study as an example. The first iteration of the direct mapping study identified papers that considered integration testing in the cloud context. We expect that by performing a second iteration of direct mapping to the integration testing topic, we would be able to obtain papers that treated integration testing from the perspective of other domains, such as web services. With two iterations, we could compare the perspectives of the web services domain with those of the cloud domain on the integration testing topic. This comparison may benefit our understanding of an existing topic from multiple application domains.

### 4.1.3. Completeness of report: Dimensions integration

The 5W+1H model is considered sufficient to completely report news, but we found it insufficient to completely and precisely report an MS of CST-interface research by simply mapping the 5W+1H pattern to organize the RQs and report results. This is because in order to adequately comprehend a piece of research work, including its values and limitations, the investigator has to simultaneously identify its motivation, problem, and solution, which are concerned with several dimensions rather than individual ones separately. Thus, it is essential to integrate some dimensions to review the research work.

In our case study, for instance, we tried to integrate the dimensions of *What*, *Why*, and *Where* together to review the research content of individual work. Initially, we simply studied the papers one by one to extract its research problem (*What*) and motivation (*Why*). In so doing, we soon found that we could not effectively relate each paper with other papers to comprehend the big picture. This limitation posed a challenge to us in synthesizing the common research motivations and issues. To address this challenge, we propose to supplement the 5W+1H pattern with topic-specific properties. In our case study, we adopted the generally accepted three-layer cloud service architectural model to refine the 5W+1H pattern. Since different classifications could lead to different styles in integrating multiple dimensions, investigators need to propose their own specific dimension integration mechanism to serve their research purposes.
4.1.4. Effort and cost effectiveness

Performing an MS requires great effort in each step of collecting and analyzing existing work on a topic. In our case study, with the input search string, we automatically retrieved almost up to 3000 records from the digital databases in the two phases. The first tedious task we did was to manually download each item-indexed paper and browse it to verify against the inclusion and exclusion criteria. The second time-consuming task was to critically and rigorously review each of the collected papers in the PVS to extract and analyze information for answering the research questions.

Based on our experiences, we have two suggestions to improve the cost-effectiveness in performing an MS. First, part of the paper identification process may be automated. For example, the first exclusion criterion (EC1) is to exclude a paper with fewer than 4 pages. The number of pages may be determined from the page numbers of the indexed papers. Thus, this step can largely be automatic.

The second suggestion is on the step of information extraction from papers and data analysis for answering the research questions. In our case study, four of the six dimensions under study are related to the metadata of the papers: the Who dimension is concerned with the author and affiliation data, the Where dimension is concerned with the publication venue data, whereas the When and How dimensions are concerned with the data of the publication dates and the papers’ references. While we had collected these data by manually reading the papers in our case study, it is possible to extract these kinds of basic information automatically. On the other hand, some other information (such as the researchers’ motivations in the Why dimension and the research problems in the What dimension) must rely on the investigators’ manual effort in critically and carefully reviewing the papers. Take the two-dimension classification scheme in our case study as an example. To map the papers into the scheme correctly, all investigators individually classified each paper first and then gathered together to address any conflicts by face-to-face discussions. We found this procedure effective and relatively efficient in avoiding potential errors in producing the mapping results.

Staples and Niazi [43] suggested that defining narrow research questions is critical to controlling the effort of performing a SLR. However, defining narrow research questions itself requires nontrivial effort and good prior understanding of the topic. Once again it begs for the question of how to kick off the process by novices who are initially unfamiliar with the domain under study. Our proposal of applying the 5W+1H pattern is precisely one way out of such a dilemma.

4.2. Use of the 5W+1H pattern in an MS

We have proposed a framework of applying the 5W+1H pattern to structure RQs and report results of an MS of a research topic. Based on the case study of an MS of CST-interface research, our experience was that this six-dimensional mapping mechanism was indeed helpful for researchers to comprehensively approach a new topic for understanding. However, we also believe that this general exploratory framework may work well under certain, but not necessarily all, MS scenarios. For some other MS scenarios, investigators may be able to use the 5W+1H pattern to define more effective RQs and investigation procedures to explore a topic. Specifically, we may classify MSs into two types. In the first type, investigators define specific RQs to study a well-formed topic (such as concurrency bug detection and regression testing of web services), while in the second type, investigators aim at exploring the research progress of an emerging and unfamiliar topic (such as CST interface in our case study). For MSs of the first type, the investigated topic has a well-formed definition and researchers often know relatively clearly what problems they would like to focus on. Thus, it is better to design RQs focusing on the interested aspects rather than adopting the general 5W+1H pattern to start the generic procedure of MSs. For MSs of the second type, investigators are challenged by defining unbiased RQs to cover an unfamiliar topic. Thus, it seems to be a good idea to adopt our proposed framework of applying the 5W+1H pattern in order to kick off and expedite the process of the MSs, as the framework contains some easy-to-follow concrete guidelines. After obtaining a deeper understanding of the research topic, investigators may then treat the MS as the first type and further explore the topic.

5. Related work

In this section, we briefly review existing uses of the 5W+1H model in the software engineering and services computing domains as well as some recent studies on cloud software testing.

5.1. The use of 5W+1H model in software engineering and services computing

To enhance the rigor of the SLR process, Kitchenham et al. [29] propose to apply evidence-based concepts from medical research to the software engineering domain. Kitchenham and Charters [28] then define a general framework for an SLR on software engineering topics. Kitchenham and other colleagues [30] note that a manual search process might miss some relevant papers and, thus, propose to adopt an automated search process. She and her coauthors [7] also find that more guidelines are needed for an MS to be effective. Budgen et al. [7] and Petersen et al. [37], for example, have refined the MS process. However, the initial part of an MS process, which involves the selection of articles to start the exploratory search to formulate an initial impression on a topic, is still ill-defined [4][14]. Our work contributes to this part of the MS process.

The result of an automated search process of an SLR or MS is largely determined by the search string in various databases. There are other suggestions to paper searching, such as reference-based search strategies [42]. Webster and Watson [47] propose an incremental three-step paper searching approach called snowballing. This approach starts with searching papers from some generally-accepted high quality publication venues (that is, journals or proceedings) as the first step. In the second step, it includes papers in the reference lists of those collected in the first step. In the third step, it searches the databases (such as ISI Web of Science) to find other papers that cite the papers collected in the previous two steps. The snowballing approach has been applied to an SLR of the software engineering domain [48] and an MS in the services computing domain [21].

Existing SLRs and MSs typically follow a similar reporting scheme in organizing their papers. However, SLR and MS in services computing and software engineering have only emerged
for less than 10 years. It is not desirable to restrict oneself to the use of only one form of reporting scheme. Our paper proposes a 5W+1H pattern (see Table 1 and Table 2) as an alternative way to consolidate and report the results of an MS. To the best of our knowledge, there has been no existing work suggesting the application of the 5W+1H pattern to structure RQs and report results of an MS.

The 5W+1H model, though, has been used for other purposes in software engineering and services computing. Chung et al. [12] apply the model to the re-documentation of a given legacy system with UML visual models. They map the six dimensions of the model with topic-specific contexts as follows: the role of software developers (Who), the benefits of doing re-documentation (Why), the use of UML elements in various views by different roles (What), the different views, such as the use case view, of a legacy system (Where), different phases in the software development process (When), and the process of constructing the other dimension elements and building relationships (How).

Context-aware applications rely on the captured context information to maintain their performance. Existing context modeling techniques are specific to certain information (such as location), leading to a tight cohesion between contexts and applications. Jang et al. [22] use the 5W+1H model to build unified user-centric contextual information to be shared among several applications. The six dimensions would completely cover the complicated context. Yang et al. [51] use the model to build a conceptual modeling framework to analyze domain concepts and relationships from the six aspects. None of them has applied the model to conduct an SLR or MS in the field, nor did they formulate the structure in each dimension as what we have done in Table 1 and Table 2.

Literature reviews in the services computing domain are very popular. A number of excellent surveys were recently published, covering the quality of service assessment [31], service composition methods [34], execution simulation [42], and transaction control [44]. To the best of our knowledge, in spite of their excellent contributions, none of these surveys deals with the testing phase in the lifecycle of cloud applications and infrastructure, nor do they report their findings by following certain common patterns. The lack of a common way to summarize report findings shows the merit of the 5W+1H pattern that we have proposed in our work. For instance, our pattern requires a formulation of conjectures and then a validation of the conjectures instead of merely reporting what were observed but leaving the overall judgment on the reported observations to the readers.

The use of the 5W+1H pattern alleviates a problem of MSs conducted by inexperienced researchers. Some researchers may also find it difficult to formulate the inclusion and exclusion criteria after defining the search string. There are empirical studies reporting experiments to alleviate this difficulty. For instance, Skoghund and Runeson [41] propose to start an SLR (including an MS) with a set of “take-off” papers and then follow the references of these papers to locate cited and citing articles and expand the set of papers iteratively. This independent approach further supports our comprehensive proposal to apply the 5W+1H pattern to MS. Incidentally, based on recent observations from four SLRs in an academic setting, Lavallée et al. [32] independently echo the proposal of an iterative approach for formulating research questions, searching and selecting the paper sets for review, as well as performing other tasks of the SLR to ensure the completeness and repeatability of the reviews conducted by novices. Our experiences, discussed in Section 4.2, likewise call for hierarchically applying the 5W+1H pattern to iteratively identify subproblems and objectives of the research topic under study and to deepen the investigator’s understanding of the topic in multiple application domains.

5.2. Cloud software testing

In our case study, we have applied the 5W+1H pattern in surveying primary articles published between 2010 and 2012. It is natural that the research field has further evolved since then. In this section, we briefly review selected and representative work in cloud software testing published after the period under our case study. Like the majority of papers that have been summarized in the RQ4 of Table 1, the work reviewed here also focuses on addressing the testing challenge in the upper layer. These papers cite the work in the PS. As they were only published recently, we do not analyze their citations.

Yan et al. [50] propose WS-TaaS, which aims to address the service load testing problem (Topic 4) by building a SaaS application on top of PlanetLab. The research idea is to develop a new heuristic algorithm for test task scheduling while achieving geographical distribution diversity in terms of service invocation requests.

Portillo-Dominguez et al. [39] propose an automated load testing SaaS to test web applications interacting with an expert system. The idea is to periodically collect certain samples from the applications under test, and then feed these samples to the expert system instances to compute outputs. The work is interesting in that it no longer treats an application under test as a generic application. Rather, it specializes in one component as an expert system and uses this specialization to make the work different from a general-purpose load testing SaaS. To further ease the deployment activity, the same research group [46] proposes a domain-specific language to specify the deployment process and requirements as well as to generate installation scripts.

Batarseh et al. [5] propose CATCR, a test case reduction approach to test applications in the cloud by considering geographical context information. The idea is to study the local importance of each test case in its geographical site in the cloud and then select test cases for execution according to the local importance in each site. The paper does not explicitly state the cloud service architectural layer that CATCR is built on and applies to.

Gambi et al. [16] propose a testing methodology to automatically generate robustness test cases for detecting any violation of the elastic properties of software systems through a model-based approach. The idea is to construct an elastic model in the form of a labeled transition system, assess the model-based scaling behavior, and then refine a test case to breach such scaled behavior. The work only presents a conceptual overview of the idea.

We observe that efforts to organize specialized workshops or conference tracks on cloud software testing are gaining momentum. For instance, the first International Workshop on Testing the Cloud (TTC ’13)2 was held in July 2013. Moreover, in the series of International Symposia on Service-Oriented System Engineering (SOSE)3, there are now sessions on cloud software testing. A full review of all the papers published in these venues would be beyond the scope of this report.

---

2 http://issta2013.inf.usi.ch/workshop4/
We have also scanned Scopus with the keywords “cloud computing” and “software testing” that appeared in Article Title, Abstract, or Keywords in August 2014. However, there are still a very limited number of journal publications, indicating that there is much room for the field to progress further toward maturity. This observation is consistent with our finding on our PVS dataset presented in the case study.

6. Conclusion

It is challenging to perform mapping studies on a research topic that investigators are unfamiliar with. This paper has presented the first work that proposes to adopt a 5W+1H pattern to kick off the exploratory step in mapping studies. The 5W+1H (Who, Why, What, Where, When, and How) model has been widely used in journalism to report news. Based on the 5W+1H model, we have developed an architectural style and pattern through which the initial set of exploratory research questions can be systematically formulated in six coherent and complementary dimensions: researchers (Who), motivations and objectives (Why), research ideas, problems, and solutions (What), locations in the research map and publication venues (Where), publication dates and article citation immediacy (When), and relationships among individual studies (How).

To validate the feasibility of our proposal, we have conducted a mapping study on CST interface, that is, the intersection area between cloud computing and software testing. The two-phase case study investigated the state of CST interface published in a three-year period. The process and results of the case study have provided evidence that our proposal indeed helps investigators kick off a mapping study on an unfamiliar topic.

Another major contribution of this work is the reporting of our first-hand experiences and reflective lessons learned from applying the 5W+1H pattern to systematically conduct a mapping study in an unfamiliar area. We have discussed the mapping of each dimension to the corresponding research contents, followed by extension to hierarchical and iterative applications of the pattern to deepen the understanding of a topic. To conclude, we postulate with substantiated evidence from our case study that the 5W+1H pattern can equip investigators with a generic framework to systematically study a new research topic at the initial exploratory phase.

References


Changjiang Jia is a PhD candidate in the Department of Computer Science at City University of Hong Kong. He received his BEng and MEng degrees from National University of Defense Technology, China. His research interests are program analysis, test sampling, concurrency bug detection and failure diagnosis. His research results have been reported in IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Services Computing, International Conference on Quality Software, and IEEE International Conference on Web Services.

Yan Cai received the PhD degree from the Department of Computer Science at City University of Hong Kong. He is an associate research professor in the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China. His current research interest is concurrency bug detection and reproduction in large-scale multithreaded and distributed systems. His research results have been reported in IEEE Transactions on Software Engineering, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Services Computing, Software: Practice and Experience, International Conference on Software Engineering, ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, IEEE International Conference on Web Services, International Symposium on Software Reliability Engineering, and others.

Yuen Tak Yu received the PhD degree from the University of Melbourne, Australia. He is an associate professor in the Department of Computer Science at City University of Hong Kong. His research interests include software engineering, software testing, computers in education and e-commerce. His publications have appeared in scholarly journals, including ACM Transactions on Software Engineering and Methodology, IEEE Transactions on Software Engineering, Journal of Systems and Software, Information and Software Technology, Software: Practice and Experience, Computers and Education, Information Research, and so on, as well as in leading international conferences such as ICSE, ACM SIGSOFT/FSE, COMPSAC, ISSRE, ICCE, and others. He was a past chairman of the IEEE Hong Kong Section Computer Society Chapter.

T.H. Tse received the PhD degree from the London School of Economics and was a visiting fellow at the University of Oxford. He is an honorary professor in computer science at The University of Hong Kong after retiring from the full professorship in July 2014. His current research interest is in program testing, debugging, and analysis. He is the steering committee co-chair of QRS and an editorial board member of the Journal of Systems and Software, Software Testing, Verification and Reliability, Software: Practice and Experience, and the Journal of Universal Computer Science. He also served on the search committee for the editor-in-chief of the IEEE Transactions on Software Engineering in 2013. He is a fellow of the British Computer Society, a fellow of the Institute for the Management of Information Systems, a fellow of the Institute of Mathematics and Its Applications, and a fellow of the Hong Kong Institution of Engineers. He was awarded an MBE by The Queen of the United Kingdom.
Appendix I. List of papers in the PS (Phase 1)


Appendix II. List of papers in the VS (Phase 2)


Appendix III. Review of papers in the PVS

This appendix briefly reviews the papers in the PVS by cloud layers, phases of the case study, and testing topics.

III.1. Testing with respect to challenges in IaaS

We identified four PS papers and one VS paper that addressed challenges in IaaS (see Table 7 and Table 8). The PS papers studied two topics: (1) fuzzing, and (2) robustness testing, while the VS paper studied the topic: (8) integration testing.

[Topic 1: Fuzzing, Phase 1] To test whether a virtualized infrastructure precisely emulates its physical counterpart without fault, Martignoni et al. [S19] applied protocol-based fuzzing to generate randomized test cases, execute both the virtual machine and the corresponding physical system over the same test cases with respect to the network protocol applicable to communicate with the systems, and compare the behavior of the two systems.

[Topic 2: Robustness testing, Phase 1] Banzai et al. [S2] proposed the D-Cloud framework to support robustness testing of distributed applications. D-Cloud generated a set of faulty virtual machines, each designed to simulate certain scenarios of device failures. In a test execution, D-cloud coordinated the activations of faults among the machines so that a distributed (fault-tolerant) application running on them could be tested under the simulated failure scenarios. The same research group then extended D-Cloud [S12][S13] to support the detection of customizable hardware failures by modeling the behavior of customized hardware devices and generating an emulator for every such device linkable to the standard virtual machine without having to modify the source code of the latter.

[Summary on IaaS for Phase 1] The reviewed work aimed either at exposing faults in the IaaS layer by testing whether it precisely emulates its physical counterpart [S19] or to inject faults into the IaaS layer and test the robustness of the distributed software running on it [S2][S12][S13].

[Topic 8: Integration testing, Phase 2] Versteeg et al. [S48] proposed a framework called Kaluta to address the scalability challenge in the IaaS layer of integrating thousands of cloud components with software deployed on the cloud. The authors viewed that the challenge had been ignored, while many existing studies mainly focused on generating scalable client requests from the user domain. A simple approach that initialized virtual machines to simulate cloud components may work but incurs high costs. Versteeg et al. [S48] proposed to model only the interacting parts (called endpoints) rather than all the cloud components in order to reach a lower cost (possibly with a reduced precision) when testing the integration relationships. Kaluta modeled three levels of each cloud component: (1) protocol modeling to describe the syntax of communication messages, (2) behavior modeling to provide the logic to process the communication messages, and (3) data modeling to populate these messages with real data values. Kaluta could then simultaneously execute many endpoints to emulate the integration between the software under test and the thousands of emulated cloud components. Their experiments showed that Kaluta could handle the scale of 10,000 endpoints simultaneously on a single physical host.

[Summary on IaaS for Phase 2] The reviewed work [S48] attempted to address the challenges in the integration of deployed software when the number of integrated cloud component is very large.

III.2. Testing with respect to challenges in PaaS

We identified six PS papers and five VS papers that addressed challenges in PaaS (see Table 7 and Table 8). The PS papers studied five topics: (2) robustness testing, (3) concurrency testing, (4) performance testing, (5) testing strategy, and (6) context sensitivity, while the VS papers studied two topics: (1) fuzzing, and (4) performance testing.

[Topic 2: Robustness testing, Phase 1] Gunawi et al. [S11] modeled platform failure scenarios as a set of prioritized failure patterns. They proposed to activate each sequence of prioritized failure patterns in a test execution and produce different traces to test different such sequences. They prioritized the scenarios using heuristics so that multiple and dissimilar platform failure scenarios within the same test execution could be tested first, which addressed the challenge of exponential explosion of multiple failures should a brute-force approach be used. They developed a relational language for developers to specify failure recovery behavior of the application so that the correctness of test results after failure recovery could be verified. The same research group [S16] further proposed a programmable approach to multiple failure injection, which allowed developers to express their own policies for pruning the explorations of different combinations of platform failure scenarios.

[Topic 3: Concurrency testing, Phase 1] The output of a MapReduce platform [15] can be nondeterministic due to parallelism in its computations, but the programming model of MapReduce provides no explicit control mechanism to specify determinism. Csallner et al. [S7] deemed that each MapReduce-specific API calling sequence in the same execution should yield deterministic values. They symbolically executed the program to look for possible nondeterministic values returned to a call site, and generated a test case if a violation was found.

[Topic 4: Performance testing, Phase 1] The large-scale table stores (such as BigTable [9] and HBase [2]) are becoming increasingly complicated due to the inclusion of optimization features for various properties. To better understand the performance of the table stores, Patil et al. [S24] developed YCSB++ on top of the Yahoo Cloud Service Benchmark (YCSB) and presented a methodology for benchmark testing of a selected number of advanced functionality (such as weak consistency, bulk insertions, table pre-splitting, server-side filtering, and fine-grained access control) of cloud-based system storage components.

[Topic 5: Testing strategy, Phase 1] Robinson and Ragusa [S27] proposed criteria to assess the needs of using a cloud platform to conduct software testing. They measured the test requirements of an application with respect to seven high level criteria in each of five usage patterns (proposed by the authors) and recommended to conduct testing with the one whose weighted sum of the measured values was the highest. The five PaaS usage patterns ranged from hosting the application and its test environment in a single cloud site to allowing a multisite environment with test infrastructure distributed to different sites.

[Topic 6: Context sensitivity, Phase 1] Jenkins et al. [S15] investigated methods to cost-effectively test the APIs of a cloud platform (such as the Google App Engine) with respect to an application. They exploited the heuristics that only some but not all APIs were used in a typical application. To assure the part of the platform relevant to an application, they proposed to construct harnesses to test the API invocation sequences that the execution traces of the application could include.
[Summary on PaaS for Phase 1] The reviewed testing research on PaaS fell into three categories.

The first category included papers on robustness testing [S11] [S16], concurrency testing [S7], and context sensitivity [S15]. The traditional mainstream approach to software testing focused on testing of the logics of an application and assumed that the platform on which the application runs is correct. Papers in this category explicitly challenged such an assumption. Their common idea was to test with the overt aim of simulating or detecting failures originated from the application’s underlying PaaS when it runs on a cloud platform.

The second category included the paper on benchmarking [S24]. It considered the issues arising from the differences among PaaS configuration instances. The paper [S27] in the third category suggested reducing testing costs by restricting the usage scope of the PaaS with respect to an application.

[Topic 1: Fuzzing, Phase 2] Mahmood et al. [S43] studied the security testing issues of Android applications with fuzzing techniques in the PaaS layer. Traditional fuzzing techniques on security testing randomly generate test cases and then execute them to expose potential vulnerabilities, but test cases generated with randomness are often ineffective in exposing security flaws and executing a large number of test cases may take a long time. Mahmood et al. [S43] proposed to improve the effectiveness of security by generating test cases that achieve higher code coverage. To speed up test execution, they proposed to deploy the same testing environment to multiple cloud nodes. The same research group presented a framework and the implementation details of their technique for testing Android applications [S44]. Specifically, the framework first identified the I/O interfaces of an Android application. Then, it used a revised version of the symbolic execution engine called Java Pathfinder to systematically generate test cases that could achieve good code coverage. This framework also initialized multiple Android emulators in cloud nodes to parallelize the test execution.

[Topic 4: Performance testing, Phase 2] Papers in the VS addressed two types of performance testing: configuration testing [S42] and load testing [S47][S49].

Owing to the complicated dependencies among different components in the same distributed software and the environmental changes in cloud platforms, it is challenging to automatically generate code to test deployed distributed software under a multitude of different configurations in the PaaS layer. Jayasinghe et al. [S42] proposed a multistage approach that generated an XML-based intermediate file to isolate complexities among different stages. They also applied aspect-oriented programming to modularize the effects of cloud platform environments. Their framework, called Expertus, was capable of testing the order of 10,000 different configurations.

Yan et al. [S49] focused on generating loads with a geographically distributed property, which had the potential to better simulate the characteristics of real world scenarios. They also improved load testing by automatically distributing the testing tasks into multiple cloud computing nodes. Vasar et al. [S47], however, mainly focused on how web applications maintained their QoS under heavy workload. They developed a workload prediction model by applying queuing theory to historical workload data. Given the average service time, the model could compute the expected number of cloud virtualized servers needed to maintain the QoS for the incoming workload.
Ciortea et al. [S5] developed Cloud9 to parallelize the dynamic symbolic program execution into a set of nodes in a cloud. Starting from a single node, Cloud9 gradually partitioned the symbolic execution tree into isolated parts and assigned them to different nodes dynamically (if possible) with a load balancing strategy to prevent excessive workload on any node (Bucur et al. [S3]). To reduce redundant state explorations, Cloud9 also used execution context abstractions to identify the state of each node.

Staats and Pasareanu [S30] proposed a technique called Simple Static Partitioning to split an execution tree into subtrees to be executed on a distributed worker node. The path constraint of the original tree to reach the root node of each subtree is treated as a precondition to ensure symbolic exploration of states reachable from the root node of the whole tree.

Oriol and Ullah [S22] extended the York Extensible Testing Infrastructure (YETI) [S2] to generate and execute random (concrete) test cases in parallel on MapReduce, and aggregate the faulty execution traces before presenting the latter to testers.

Layered approaches have been used to organize tests to run in a massively parallel manner. Wu et al. [S34] organized all the test activities at test control nodes that shield the next layer of nodes (that is, test center nodes) from the testers. The test center nodes in turn would partition the assigned testing activities into jobs and distribute them to different worker nodes for execution. Zhang et al. [S38] also proposed a three-layer approach. The user management layer would receive a test request from a tester and pass it as a task to the system control layer, which would then be distributed to different testing nodes of the bottom layer.

[Topic 8: Integration testing, Phase 1] Integration testing aims at detecting failures due to mismatches among the interactions of the components in an application. When conducted in an open environment, integration testing should further address the challenges due to the use of dynamic service-oriented architecture that allows an application to dynamically discover, bind, and communicate with previously unknown services at runtime or change its registered interface. Testing the composition of web services that can themselves evolve independently is challenging. Typically, the composition and testing of web services are organized into separate steps, which may not be optimal. Vengattaraman et al. [S33] proposed a conceptual model of the software under test as a composition of (web) services controlled by a service manager. The service manager would first put a subset of the composition into a service container. Other implementations of web services that were discovered to form a target service composition with this subset of web services would then be added to the container. This approach can assemble possible variants of the original service compositions to test their subsets.

Tsai et al. [S31] addressed the same problem from another perspective. They considered the composition of services as a hierarchical process, where each intermediate node represented a service sub-composition. If no control at an upper level in the hierarchy could govern the selection of a particular set of service implementations to be tested at a lower level, the latter might select a sub-composition irrelevant to the testing at the upper level. Tsai et al. [S31] proposed a framework for testers to specify a service implementation to be used in a composition at service discovery. Their testing for a target service composition had been extended to include the testing of service composition candidates. They also proposed to execute test cases of different levels in parallel by using a MapReduce-like framework.

King et al. [S17] and King and Ganti [S18] associated each SaaS application with a test environment consisting of a test data repository, a program repository, as well as the PaaS and IaaS where the test cases were run. They proposed the notion of Test Support as a Service (TSaaS) to allow other users to perform integration tests in this environment (after being customized to fit the requirement in a TSaaS setup request), thereby achieving better controllability and observability for testing.

All the papers on this topic [S17][S18][S31][S33] considered the control aspect of the execution of integration tests, while King et al. [S17] and King and Ganti [S18] further addressed the test observability issue.

[Topic 9: Regression testing, Phase 1] Regression testing reuses existing test cases to safeguard a modified application from unintended changes in artifacts. To address the challenges due to dynamic changes in registered interfaces, Cooray et al. [S6] monitored the web service repository during a round of testing an atomic web service to spot any change (such as addition, deletion, or revision of parameter types of an operation signature) in its web service interface description (such as WSDL document), and fixed the existing test input data to cater for changes in the remaining part of the current round of testing.

Huang et al. [S14] studied safe regression test case selection as a service for developers. They pointed out that many existing effective regression testing techniques were limited by source code availability and scalability issues. To lower the test cost and improve the efficiency, they proposed to analyze the builds of two (consecutive) versions of an application to identify the change points and then select test cases that traverse these change points. They also built a prototype cloud regression testing tool and validated its effectiveness in selected projects of software vendors.

Both of the above two papers studied the issue of change identification for regression tests. Cooray et al. [S6] further explored the dimension of test case repair, whereas Huang et al. [S14] examined efficient execution runtime profiling.

[Topic 10: Security testing, Phase 1] Two papers studied the issue of security testing of SaaS applications from two opposing perspectives: normal execution [S29] and abnormal execution (such as use of malicious code) [S37]. Srivastava et al. [S29] exploited the differences in multiple independent implementations of the same Java class library API to detect potential security violations. Zech [S37] generated negative security test cases (such as malicious code usage) based on a risk model that represented the past vulnerabilities of the SaaS application.

[Topic 11: Migration testing, Phase 1] An application may reveal failures when ported from a physical platform to a virtual one in the cloud. Despite their similarity, the two platforms may differ in some specific information (such as references to resources and secure network connection in the environment) in a test case or in the output (say, temporal information) of the application, resulting in false positives from the execution or output comparison.

Ding et al. [S8] proposed to address these problems by Splitter, which served as an intermediary between a client and an application on the original platform. For each test case, Splitter intercepted the messages between the client and the current version of the application, translated the client request by mapping the environment-specific contents (such as cookies, URL parameters, POST message, and JavaScript contents generated by the application on a platform) from the original platform to a target
platform, and forwarded the translated request to the same application on the target platform. It then compared the returned messages from these two applications on the original and the target platforms via a multilevel similarity-based approach to effectively triage failures, thereby facilitating developers to diagnose the migration problems.

[Topic 12: Log analysis, Phase 1] The log files of long-lived applications may contain a huge amount of data. Nagappan [S21] mentioned the development of a framework that could abstract the contents of log files into a generic format on which different log analysis techniques (such as fault diagnosis and operational profiling) could be built, but no other details were presented in the paper.

[Summary on SaaS for Phase 1] Compared with those on the IaaS and PaaS layers, research studies on the SaaS layer were distinctively more diverse. Three categories of SaaS studies could be identified.

The first category was concerned with nonfunctional testing. It included the development of multilayer performance metric model [S10][S32] and methods to detect vulnerability faults in applications in normal executions due to different library implementations [S29] as well as abnormal executions through model-based risk analysis [S37]. Compared with the large number of emergent properties studied in traditional testing, those explored for cloud software testing in the reviewed period had been very limited, which might indicate much opportunity for further research.

The second category was concerned with test cost reduction. This could be done by optimizing the amount of resources required to support testing [S28], reducing the tests needed [S6][S14], and enabling test case reuse [S8]. The tradeoff between effectiveness and cost reduction, however, was not adequately studied.

The third category was concerned with test model development. It included the development of new models such as TaaS [S4][S35] [S36], new methods to improve testing parallelization approaches [S5][S22][S30][S34][S38], and new ideas to improve test controllability and observability [S17][S18][S31][S33]. There was also a study that proposed to build a generic data format to ease dynamic analyses [S21].

[Topic 7: Testing parallelization, Phase 2] Many authors have proposed ideas (such as [S3][S30]) to partition the symbolic execution based testing task and distribute it in multiple cloud nodes for parallel processing. All such techniques rely on forward path exploration to dynamically partition the workload. This strategy would lead to unbalanced partitions among different computation nodes.

To address the partition balancing problem, Aleb and Kechid [S39] proposed to statically model the program to produce a static balanced partition. They identified the path constraint of each program variable by iteratively formulating the conditional and loop constraints of possible occurrences of each variable. By doing so, they modeled the whole program as a set of static variables with path constraints. Then, they distributed this variable set evenly in terms of the number of estimated paths to a set of distributed worker nodes in a computing cloud. A special worker node called coordinator took charge of organizing the results of other worker nodes. They reported a case study to validate the effectiveness of their static program modeling approach in reaching a static well-balanced partitioning.

[Topic 9: Regression testing, Phase 2] To address the issue of source code unavailability in regression testing, Huang et al. [S14] had proposed earlier to identify the changes in the builds between two consecutive versions of the same program and then generate test cases to traverse the change points. They stored the testing data and the code of the builds in a relational database, which formed a performance bottleneck when the testing trace files grew to millions of function invocations.

To overcome the limitations due to the performance bottleneck, in their later work [S40], Huang and other researchers redesigned the data model and testing execution to fit the BigTable and MapReduce frameworks, respectively. This new design enabled the testing service to identify the changes of a new build against each uploaded build instead of only the consecutive build.

[Topic 10: Security testing, Phase 2] Existing work on security testing mainly relied on cloud providers to achieve the security guarantee of lower cloud service architectural layers (such as virtual machine security in the IaaS layer). However, the deployed software in a cloud may introduce new security risks due to other public service interfaces, which might be invoked by malicious users.

Zech et al. [S50] proposed to build a system model for the Cloud Under Test (CUT) by using UML2 to describe the public interfaces of the CUT and then generate a risk analysis model by applying certain attack patterns to the system model. The framework would then generate test case descriptions and execute them against the CUT. The CUT itself might evolve its state when new application services were deployed. To handle this dynamic property of the CUT, the framework adopted a change-driven testing approach to build an adapted model to react to such changes. The system vulnerability database was also updated using feedback from the test outputs. By doing so, the framework enabled an iteration of security tests to use the accumulated knowledge of previous rounds to handle the evolving CUT more effectively.

[Topic 13: Unit testing, Phase 2] It is challenging to conduct unit testing of cloud-based software applications because some specific program paths are only traversed under certain combinations of test inputs and cloud states. Existing solutions mainly relied on building cloud stubs manually to simulate the interaction between the cloud units and external components. However, this process of building cloud stubs usually incurs huge manual effort.

To reduce human effort, Zhang et al. [S51] proposed to build a simulated cloud model that satisfied the interface requirements of the cloud to deploy the software application under test. This model could generate test inputs and cloud states to test the application unit. By converting each cloud state into a sequence of cloud API invocations, the same state could be reconstructed to simulate the real environment.

[Summary on SaaS for Phase 2] Research studies of the SaaS layer can be classified into two categories. The first category was concerned with utilizing the abundant cloud resources [S39] and existing cloud services [S40] to speed up test execution. The second category addressed the testing issues introduced by the interfaces of cloud components, such as the problem of security vulnerability due to malicious invocation of cloud interfaces [S50] and the problem to achieve testing coverage of application units that interacted with cloud interfaces [S51].