Vertical Test Reuse for Embedded Systems: A Systematic Mapping Study

Daniel Flemström, Daniel Sundmark and Wasif Afzal
School of Innovation, Design and Engineering
Mälardalen University, Västerås, Sweden
{daniel.flemstrom | daniel.sundmark | wasif.afzal}@mdh.se

***Abstract*—Vertical test reuse refers to the the reuse of test cases or other test artifacts over different integration levels in the software or system engineering process. Vertical test reuse has previously been proposed for reducing test effort and improving test effectiveness, particularly for embedded system development. The goal of this study is to provide an overview of the state of the art in the field of vertical test reuse for embedded system development. For this purpose, a systematic mapping study has been performed, identifying 11 papers on vertical test reuse for embedded systems. The primary result from the mapping is a classification of published work on vertical test reuse in the embedded system domain, covering motivations for reuse, reuse techniques, test levels and reusable test artifacts considered, and to what extent the effects of reuse have been evaluated.**

I. Introduction

Test and verification activities are widely known to account for a major part of the development cost for software. Since testing of embedded systems pose additional challenges compared to testing of regular software [1], the same statement can be safely made regarding such systems as well. Moreover, as the complexity and size of embedded software increases, the cost and effort for testing naturally follow. In this situation, there is a need for reducing the test activity effort while still ensuring that the same, or even an increased quality assurance is attained by these activities.

The above argumentation can be more specifically illustrated with recent trends in the development of embedded systems in the automotive industry, where an increasing number of features rely on complex interaction between different subsystems. Many of these features are also autonomous, and should function with little or no driver intervention. This growth of complex features must not compromise vehicle safety, and occurs in parallel with the introduction of new development standards such as ISO26262 [2], posing numerous requirements on testing, verification, and documentation. Consequently, cost-effective methods for quality assurance are needed.

For traditional software development, reduction of test effort (often with maintained level of quality assurance as an implicitly or explicitly stated constraint) is a widely studied field [3], [4]. Reduction of test effort commonly focuses on reducing time and cost for regression testing, as it is a frequently occurring activity in contemporary software development. Regression testing traditionally concerns reuse of test cases over different software versions, primarily with the purpose of establishing that software changes do not inadvertently also affect parts of the software not intended to be changed.

While not explicitly focusing test reuse, Runeson and Engström [5] describe software product line testing as a three-dimensional regression testing problem. The regression testing dimensions are *versions, variants* and *levels*. Here, regression testing between versions has been thoroughly studied [6], [7], and regression testing between product variants has gained quite a bit of recent attention [8], [9], [10]. However, the level dimension of regression testing has received limited attention, and in contrast to test reuse between versions and variants, no summarizing literature surveys exist on this topic. We call this dimension of regression testing as *vertical test reuse*. By vertical test reuse, we refer to reuse of test cases or other test artifacts between different levels of integration. For example, test cases designed for component- or unit-level testing can (with or without modifications) be reused at the subsystem integration or system test levels.

In this paper, we provide an overview of the body of knowledge within vertical test reuse in development of embedded systems. We do this by means of a systematic mapping study. Our results indicate that relatively few papers have been published on vertical test reuse for embedded systems, and that most of these papers have been published within the last five years. In these published papers, a number of different approaches for vertical test reuse of embedded systems have been proposed, and the primary application domain has been the automotive sector. However, the actual gains or drawbacks of using vertical test reuse in real-life industrial projects are sparsely evaluated and therefore poorly understood.

The rest of the paper is organized as follows. Section 2 describes the process we followed while conducting this systematic mapping study. Section 3 presents our study results, answering our stated research questions. Validity threats are given in Section 4 while the paper is concluded in Section 5.

II. The systematic mapping process

According to Kitchenham and Charters [11], a systematic mapping study gives a broad review of primary studies in a research area. Further, guidelines for undertaking systematic mapping studies in software engineering have been proposed by Petersen et al. [12]. Based on these two sources, this section describes the process followed in conducting the mapping study.

Data base	Papers	Papers 2005-2015
IEEE Xplore	1776	1024
Scopus	1898	1307
WoS	1334	858
Total	5008	3189

TABLE I
DATABASES SEARCHED AND RESULTS.

A. Research Questions

The purpose of this study is to identify available evidence of research within vertical test reuse for embedded systems. We are particularly interested in research trends with respect to time, publication fora and research methods used. We also seek to identify available approaches of vertical test reuse, and how their impact (efficiency and effectiveness) is measured. We, therefore, pose the following research questions:

RQ1: *What is the distribution of primary studies on vertical test reuse over time, fora of publication and research type?*

RQ2: *What are the justifications given for vertical test reuse, items of test reuse and proposed approaches?*

RQ3: *What is the extent of evaluation of the identified vertical test reuse approaches?*

The above research questions partly reflect the PICOC criteria for structuring research questions as mentioned in [11]. We do not restrict our research questions in terms of 'comparison', but have the following elements:

- Population: software.
- Intervention: testing, verification and validation.
- Outcomes: reuse approaches.
- Context: embedded systems.

B. Search strategy

After defining the research questions, the search for relevant primary studies starts with designing a search string to be used in well-known electronic databases. We started piloting our search strategy using two databases: IEEE Xplore and Scopus. After a number of trial searches, adjusting for both scope and relevancy, we finalized the following search term: `Software AND (test* OR validation OR verification) AND reuse)`. We used the above search string in three databases: IEEE Xplore, Scopus and Web of Science (WoS). Other relevant databases (such as the ACM Digital Library, ScienceDirect and SpringerLink) are covered by the two citation databases we searched (Scopus and WoS) and were therefore not individually searched.

C. Selecting studies

The search in IEEE Xplore, Scopus and WoS resulted in a total of 5008 references (including duplicates). Initially, we did not restrict the search in terms of year of publication, rather used the default year setting. Later, we realized that it will be useful to narrow-down on papers published in and after year 2005. This decision was based on the fact that a recent literature survey on approaches for reducing test effort [4] covers very few papers published in and after year 2005. Secondly, the papers covered prior to 2005 could be excluded using the same criteria as for the collected papers, as described later in this section. Therefore restricting to year 2005 and later was considered to present a more recent and relevant state-of-the-art. However, to ensure completeness of search, a snowballing step was added to explicitly look for evidence prior to year 2005.

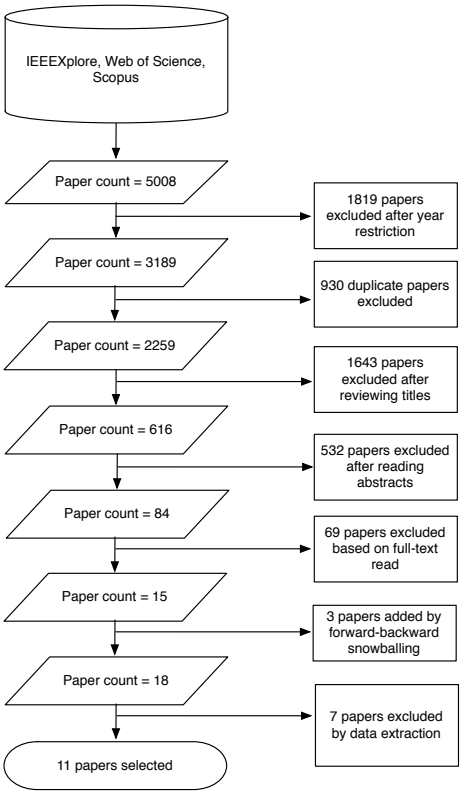

Fig. 1. Study selection process.

Restricting the year of publication to year 2005 and later gave us 3189 papers (Table I) and were further reduced to 2259 papers after duplicate removal.

After duplicate removal, the next step in study selection was to exclude papers by reading titles that were obviously not relevant. We also removed references that represented conference titles and standards. The abstracts of the remaining 616 references were read in detail to reduce the set of references to 84. During this, and the subsequent full-text inspection phase, we aimed to identify studies that relate to test reuse between different test levels in embedded systems development. Consequently, papers meeting the following following criteria were excluded in the screening process:

- Papers that did not relate to software engineering/ computer science.
- Papers that did not relate to the embedded systems domain.
- Papers that did not relate to software testing.

Authors	Ref	Year	Title	Forum	Research Type
Pérez and Kaiser	[13]	2010	Bottom-up Reuse for Multi-Level Testing.	Journal	Evaluation Research
Benz	[14]	2007	Combining Test Case Generation for Component and Integration Testing	Workshop	Solution Proposal
Schätz and Pfaller	[15]	2010	Integrating Component Tests to System Tests	Workshop	Solution Proposal
Pérez and Kaiser	[16]	2009	Integrating Test Levels for Embedded Systems	Conference	Solution Proposal
Pérez and Kaiser	[17]	2010	Multi-Level Test Models for Embedded Systems	Conference	Solution Proposal
Rodrigues et al.	[18]	2011	Refactoring VeriSC Testbenches to Improve the Functional Verification during the Integration Phase	Conference	Evaluation Research
Pérez and Kaiser	[19]	2009	Reusing Component Test Cases for Integration Testing of Retarding Embedded System Components	Conference	Solution Proposal
Pérez and Kaiser	[20]	2010	Top-Down Reuse for Multi-level Testing	Conference	Solution Proposal
Asaithambi and Jarzabek	[21]	2013	Towards Test Case Reuse: A Study of Redundancies in Android Platform Test Libraries	Conference	Solution Proposal
Choi and Bunse	[22]	2011	Design Verification in Model-Based μ-Controller Development using an Abstract Component	Journal	Evaluation Research
Schultz et al.	[23]	2002	Multilevel Testing for Design Verification of Embedded Systems	Journal	Solution Proposal

TABLE II
LIST OF INCLUDED PAPERS.

- Papers that did not relate to software reuse.

We also excluded papers focusing on regression testing. Although regression testing does represent a form of reuse, we do not include such papers because they focus on reuse in different versions of the same product or variants of the product. In contrast, our scope is limited to reuse between different test levels, regardless of version or variant of a software.

The remaining 84 papers were then read in full-text, further reducing the number of papers meeting the inclusion criteria to 15. In order to make sure we do not miss any relevant papers prior to year 2005, we added a forward-backward snowballing stage [24] for the 15 references. The references of these 15 papers were merged and we applied our exclusion criteria on this set. The only difference now is that we included papers older than 10 years to catch relevant papers prior to year 2005. We read the titles and if required, read the abstracts and the full-text. Further, for each reference of the 15 papers, we checked their reference lists and also used Google Scholar to find the citing papers. The step added 3 more relevant papers not identified in the initial search. The remaining 18 papers were used to perform the data extraction. The data extraction further revealed 7 papers that were irrelevant. Ultimately, we were left with 11 references as our set of primary studies. The study selection process is shown in its entirety in Figure 1.

D. Study quality assessment

No quality assessment of included primary studies was conducted after the study selection phase. The reason for this choice was that we wanted to be as inclusive as possible with respect to the vertical test reuse approaches.

E. Data extraction

In order to extract relevant data from the set of primary studies, a data extraction form was created. The form was designed in such a way that the extracted data would correspond to the information needed to address the research questions. Specifically, we extracted the following information from the included primary studies:

- Year of publication.
- Forum of publication.
- Research method used.
- Reusable test items, levels and dimensions.
- Selection criteria for reusing items.
- Arguments in favor of reusing items.
- Approaches of test reuse.
- Means of evaluation and measures of effectiveness and efficiency.
- Practical implications and limitations.

III. RESULTS

This section presents the results of the systematic mapping on vertical test reuse. First we present the list included papers, and provide information on how different authors and groups have contributed to this field over the last 13 years. This is followed by the motivation for vertical test reuse stated in the included papers. In Section III-C we classify each proposed reuse approach into one of three conceptual categories. The final section covers how the proposed reuse methods have been evaluated, particularly related to the aforementioned motivation for vertical test reuse.

A. Published work on vertical test reuse for embedded systems

Table II shows a complete list of the primary studies identified in the search process. Most of the papers (6 out of 11) are conference papers while three are journal articles. To categorize the research type, we make use of a classification scheme proposed by Wieringa et al. [25]. A majority of the papers (8 out of 11) are solution proposals (i.e., a presentation of an approach or method to solve a problem, often accompanied by a small proof-of-concept example or supporting arguments), while the rest are evaluation research (i.e., a study where a technique is implemented in practice and its drawbacks or benefits are evaluated). Consequently, a majority of the examples in the primary studies, although

relevant and from the automotive domain, were rather small (less than 10 components). Table III shows the distribution of primary studies with respect to year of publication. 2010 saw the highest number of publications, however this does not truly reflect increased activity within the area since it was mostly two distict groups of researchers publishing in year 2010. In order to visualize this, the studies from the same authors were given a darker color in Table III, and the number of duplicate authors were put within parentheses in column 2. The most activity was found between years 2008 and 2011 and we found nothing published after year 2013.

Year	Count	Published Papers
2002	1	[23]
2007	1	[14]
2009	2(1)	[16][19]
2010	4(2)	[13][15][17][20]
2011	2	[18][22]
2013	1	[21]
	11	

TABLE III
PAPERS PUBLISHED ON VERTICAL TEST REUSE PER YEAR.

B. Motivation for reuse, items of reuse, and test levels involved

Each method for vertical test reuse proposed in the literature is designed with a particular purpose in mind. In this section we present findings from the primary studies concerning the motivations we found for vertical test reuse, what test items are reused and at what test levels. Table IV presents the different purposes of vertical test reuse, as stated in the included primary studies. In general, four **main motivations** for vertical test reuse is possible to identify in the included studies:

- **Avoiding rework for similar test cases**
  Many of our primary studies argue that a significant number of test cases share commonalities with respect to different test levels [13], [16], [17], [18], [19], [20], [21]. A reason for this is that different test levels are performed by different departments, while much of the test behavior is similar. Therefore, there is an opportunity to reduce rework when testing at different test levels by leveraging on such commonalities. Details on such approaches are presented in section III-C.
- **Increasing quality of test cases** Pérez and Kaiser [13],[19] argue that the organizational separation between the people performing tests in the different test levels is problematic. This paves way for the argument that vertical test reuse might increase collaboration across test levels and thus yield better test case quality. A few of our primary studies argue that the separation of test levels prevents synergies and collaborations, thus missing out on opportunities to improve test case quality.
- **Avoiding costly creation of new environments** In contemporary embedded system development, subcontracting of components is commonly used. However, before integrating third party components into a system there may be a need for internal testing on the component level [15]. Since this sometimes requires new, potentially tailor-made testing environments for the third party components, the cost for creating these new environments on the component level cannot be justified. In Section III-C we describe an approach found in [15] that addresses this problem.
- **Reducing test case complexity**
  The number of possible test cases increases rapidly as the state space increases [22], [14]. Benz [14] argues that raising the abstraction using interaction modeling and component model reuse has a positive effect on the size of the state space and that this would indirectly reduce the complexity. Other studies that indirectly or directly claim complexity reduction as a reason for resuse are [14], [17], [20], [21], [23]. Section III-C covers the approaches found for reducing test case complexity in the included papers.

Scanning the included primary studies for information regarding which particular **test items** that were considered for vertical test reuse, we found that the following items were subject to reuse:

- Abstract test cases or test models.
- Concrete test cases.
- Test Environment.
- Component specification/component model.

Table V describes the distribution of items reused with respect to the primary studies. We found that the most common item of reuse are the abstract test cases or test models [13], [16], [17], [18], [19], [20]. An abstract test case is not directly executable on a specific test level. Instead, it constitutes a higher level textual description or a model that describes the test behavior and is used for generating test cases that can be executed with or without adaptations to the environment or the test object.

Another common reuse item is the concrete test case code [13], [15], [16], [19], [20], [21]. Concrete test cases can be executed directly at one or more specific test levels with or without adaptations to the environment or test object. For the sake of this mapping, it is possible to argue that a number of studies (i.e., [13], [16], [19], [20]) propose reuse of both abstract and concrete test cases, since the approaches reuse test behavior between several tests levels. A test case may be reused from the test level where it is executable to a level where it is more abstract, since the direct dependencies to the test object and environment on that level has been abstracted away. However, it may also be reused to yet another level, which means that an abstract test case is reused. For some applications (e.g., [18]), test bench components was one of the candidates for reuse. In this case, the testing environment consists of a number of test bench components that are used for testing at sub levels of the *integration* test level. Here, each level of composition, when integrating components, requires substantial effort for writing new test bench components. The concept of test benches are further described in Section III-C2.

	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]
Avoiding rework for similar test cases	x			x	x	x	x	x	x		
Increase in quality of test cases	x						x				
Avoiding creating new costly component environments			x								
Reduction in test case complexity		x			x			x	x	x	x

TABLE IV
PURPOSE OF VERTICAL TEST REUSE.

	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]
Abstract test cases or test model	x			x	x	x	x	x			x
Concrete test cases	x		x	x			x	x	x		
Component specification or component model		x								x	
Test bench components						x					

TABLE V
TEST ITEMS SUBJECT TO REUSE.

Benz [14] propose reuse of the component specification / model itself for testing purposes, together with formal descriptions of the different activities and relations between activities such as starting or stopping a playback on a car entertainment system.

It should be noted that all test items are not suitable for reuse on all test levels, since the test levels address different abstraction levels and different aspects of testing. Not all papers discuss in what situations reuse was most efficient, however a few ideas are presented by Pérez and Kaiser [17]. Here, abstract test cases are divided into disjoint sets, each set suitable for one level of integration. This approach may also be applicable for other papers using the same underlying approach such as [13], [16], [19], [20]

Another approach to determine which items to reuse, is to use set theory to deduce which items to include (see, e.g., Rodrigues et al. [18]). There are however very few details on how the items and their relations should be specified. Asaithambi and Jarzabek [21] use a clone detection tool in order to determine which test cases, or parts of test cases, that are most beneficial to reuse.

Different primary studies have different focus in terms of between what **test levels** reuse can be applied. The different levels discussed in the papers were: *system level, integration level* and *unit/component test level* as depicted in Table VI. The table describes the test levels of interest in our primary studies together with the distribution of studies addressing each level. For example, the mark 'x' in 'Integration level(s) reuse' and another in 'Unit/component level reuse' denotes that the paper discuss reuse between these two levels. In the special case of 'x' only at 'Integration level(s) reuse', reuse is addressed between different sub-levels of the integration test level[1]. The most general approaches (e.g. [13], [17], [20], [22], [23]) focus on reuse between all levels. In embedded systems, it is common to have several levels within the integration test level where reuse can be applied. This helps to explain why a number of primary studies (e.g. [14], [16], [19]) are focusing on this level. Reusing test items from unit or component level to integration level(s) is addressed in [18], [21].

It should be noted that unit-, component-, integration-, and system-level testing are not uniquely defined concepts and may have slightly different interpretations in different domains, contexts and companies. The above summary on test level focus rests upon the terminology used in the included primary studies.

C. Approaches to Vertical Test Reuse

On a conceptual level, we found three classes of reuse approaches proposed in the primary studies: *adapter-*, *formal methods-* and *clone mining*-based approaches. Within these classes, we found that the different primary studies slightly differ in their details. In the forthcoming sections we will describe each approach and the variations found in primary studies within each approach.

1) Adapter Based Approaches: The first class of approaches, adapter-based approaches, represents the largest number of primary studies [13], [16], [17], [18], [19], [20]. The approach is based on the assumption that each test level may have a different level of abstraction and may also differ in the way that the test object can be accessed. In the approaches found, the test cases were either written entirely at the component level (*bottom-up reuse*) or the system level (*top-down reuse*). At the level where the test cases were written, the test cases may be concrete, i.e directly executable. However at other test levels, adapters are needed to overcome the differences with respect to level of abstraction as well as interface issues. The proposed solution is thus to create test cases that are concrete on the level where they were written and more abstract on the other levels. On these other levels the test case code only communicates with the test environment, and the object under test, through adapters, tailor made for each test level. The approach is described in a few different flavors (*multi-level test cases, bottom up-* and *top down* reuse) which will be described below. The first variant

[1]The integration level often consists of different sub-levels, such as component integration and software-hardware integration.

	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]
System level reuse	x		x		x			x		x	x
Integration level(s) reuse	x	x		x	x	x	x	x	x	x	x
Unit/component level reuse	x		x		x	x		x	x	x	x

TABLE VI
REUSE ACROSS TEST LEVELS IN PRIMARY STUDIES.

of the adapter-based approaches is *multi-level test cases* [16] (with a model based version presented in [17]). For each test level, there will be a set of test cases that are specific to this level but also a set of test cases that may be directly reused between different levels. The approach is to abstract away the test environment dependencies on each level to form a level-independent test core interface. The test teams use this interface to interact with the test object while focusing on test case behavior. Such an abstract test case consists of a test stimulation part, a set of test parameters and an evaluation part. Each of these parts use the test core interface to interact with the environment and the test object. For each test level, there are adapters that convert the test core interface to the required level of abstraction. In cases where some test objects introduce delays at higher test levels where more components are integrated. Pérez and Kaiser [19] suggest that the input and output adapters can be complemented with delay balance and delay compensation components to preserve the order of test object stimuli. The purpose of these compensation components is to preserve the order of the event sequences that are at risk to be out of order if a signal is delayed on one level, but not on another (e.g due to functional simulations of some components). Depending on which level the test cases are written at, the *bottom-up* approach [13] allows for writing all test cases at the component level (and then reused in higher test levels using the aforementioned adapters on each level), while the *top-down* approach [20] is designed for writing all test cases at the system level.

2) Formal Methods: Rodrigues et al. [18] also use adapters, but combine it with *formal methods*. In their approach test bench components are reused using adapters and intelligent partitioning of test cases and reference models. A set of test benches in this methodology can be mapped to the test level *integration level* in table VI but divided into *sub levels* where components are gradually integrated and tested on each sub level. Adapters are created for each in and out interface of a test object. A test object in this sense can be one individual component (at one sub level) or any number of integrated components (at another sub level). In practice, this means that testing an individual component A with two in interfaces, I_{a1} and I_{a2}, require two adapters. Similar to the other adapter based approach, the test source is abstract with respect to the test level, since the test cases only use the adapters to interact with the test object. The same stimuli can thus be fed to the test object *and* a reference model that describes the expected behavior. The major difference from the other adapter based approach is the partitioning of the test cases and adapters. While other adapter approaches such as [17] and [13] partition with respect to the V-model test level, this approach partitions the test source into disjunct sets, each set testing one interface. Furthermore, the reference model, that is used as an oracle, is also divided accordingly. After selecting a coverage criteria, set theory is applied at particular test (sub) levels, thus inferring which interfaces, adapters, tests and parts of the reference model that satisfies the desired criteria.

There is also an approach that uses hierarchical task models [14]. These models describe, in a formal way, the possible interactions between the user and the system using the ConcurTaskTree (CTT) notation [26], thus trying to reduce the state space for testing the system on lower levels. Such tasks and their subtasks also describe how different functions relate from a users point of view and also include possible timing constraints. There are also formal models for each components. Each task has pre and post conditions on required underlying component models which makes it possible to generate test cases using different coverage criteria.

Another formal approach is to use abstract components with the MARMOT model as in [22]. These abstract components and their interactions are formally specified using the notation of π-calculus and labeled transition systems. These models are refined during the whole development process and the way they are specified allows formal verification throughout the entire software development process. Test reuse is thus performed indirectly by reusing the specification of the abstract components, since that specification is used for the verification activities.

Schulz et al [23] uses formal descriptions together with an abstract system model to be able to generate consistent test scenarios (derived from specification) from the system level down to integrated-implementation level.

For third party components, where creating a component level testing environment is a problem, formal methods has been used to translate component tests so they can be performed using only the system level interfaces, thus making it possible to run (some of the) component tests in the system level test environment [15]. The idea is to describe the components and all interactions between the components and the environment as well as state transitions in a formal way. Using these formal specifications, test cases are generated. In addition, an analysis is performed to identify which behaviors can be satisfactorily tested using the system level interfaces.

3) Clone Mining: The last class of vertical test reuse is *clone mining*. In this class, we found several studies that were dismissed because they did not fall in the embedded domain. However, one study [21] focusing on embedded systems was found. The purpose of clone mining is to show how (and how

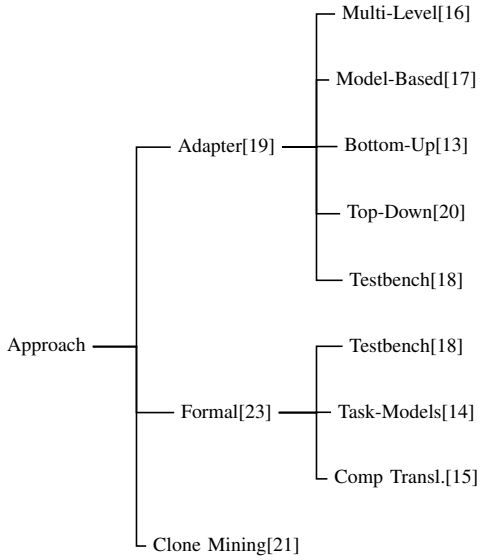

Fig. 2. Approaches to reuse.

much) test case code is really duplicated in a software project rather than proposing a new reuse method. An existing set of test cases were scanned for clones or partial clones. A clone in this sense would mean test cases that either are equal line by line or that the test case code could be reused by abstracting the differences into parameters, allowing the original test cases to be generated from the one common source file. In the study they found both test cases that were identical but they also found other variants which were classified as *identical test clones, parametric test clones, reordered test clones* (same steps but in different order) and *intertwined test clone*. The parameterized test cases found were successfully rewritten to obtain generic adaptable test cases.

D. Evaluation of vertical test reuse approaches

In this section, we present a classification of the evaluation of the proposed vertical test reuse approaches along two dimensions:

- Scale of evaluation (toy example, down-scaled real example, industrial, not mentioned/none) [27].
- Means of evaluation (proof-of-concept, comparative evaluation, none).

We present the results of this classification in Table VII. It is evident that most of the primary studies present a proof-of-concept validation (i.e., a descriptive example implementaition of the proposed approach) using either a down-scaled real example or an industrial scale example. Only two primary studies [13], [18] comparatively evaluate their approach with respect to its objective on an industrial scale.

Paper [17] is purely theoretical, and does not provide any experimental evaluation, while [16], [19], [20] only show a small example (less than 5 components) from the automotive industry that the approach works. Two papers [13], [18] measure test effort reduction while only [13] actually measure the percentage of reused test cases and compare to the number of expected test cases to be reused after performing a reusability analysis of the test cases. The largest and most thorough evaluation was found in the journal paper by P'erez and Kaiser [13], where four different sizes of test objects were evaluated. The largest test object in the evaluation had 164 inputs which we regard as *industrial* sized in this context. The evaluation in two other studies [18], [21] were also *industrial* sized. The first measured reduced effort and the second measured test code redundancy and number of test clones.

IV. Threats to Validity

There can be several threats to the validity of this study. One of the more obvious ones is the possibility of the existence of relevant papers that we failed to identify by our search. However our search covered the most commonly used electronic databases in software engineering and computer science. We complemented the electronic search with forward-backward snowballing to increase the completeness of our search. Other potential sources to lack of completeness of our search concern the paper exclusion process. We excluded papers on other two forms of reuse: in the context of regression testing and software product line testing. There is a threat that papers discuss vertical test reuse in these contexts are applicable in our context too. Moreover we limited our selection of studies to the context of embedded systems. There are studies in other domains (e.g., in web services [28]) where vertical test reuse has been discussed. We also face the threat of bias in study selection as only one researcher applied the inclusion/exclusion criteria. However the study selection criteria were discussed among the authors to promote a common understanding.

V. Conclusion and Future Work

This study presents an overview description of the body of knowledge on vertical test reuse in development of embedded systems. Specifically, we have focused on the following research questions:

RQ1: *What is the distribution of primary studies on vertical test reuse over time, fora of publication and research methods used?*

This study identified 11 primary studies on vertical test reuse for embedded systems, published in the period of 2002 to 2013. Most published work (8 out of 11 included studies) focus on solution proposals, but three studies target evaluation of vertical test reuse.

RQ2: *What are the proposed approaches, items of reuse and motivations given for vertical test reuse?*

The primary motivation for vertical test reuse seems to be reduction of test effort by means of avoiding test rework over test levels. Other motivations include reduction of test case complexity, increased quality of test cases, and less costly test environment creation. The primary thing being reused between test levels is abstract test cases or test models, and reuse is acheived through using adapters, formal methods or clone mining.

TABLE VII
CLASSIFICATION OF VERTICAL TEST REUSE APPROACHES WITH RESPECT TO SCALE AND MEANS OF EVALUATION.

	Toy example	Down-scaled real example	Industrial	Not mentioned/None
Proof-of-concept		[16], [19], [20]	[14], [21], [22], [23]	[15]
Comparative evaluation			[13], [18]	
None				[17]

RQ3: *What is the extent of evaluation of the identified vertical test reuse approaches?*

Majority of the primary studies use proof-of-concept as a means to evaluate their approaches on either down-scaled real or industrial-scaled examples. Few comparative evaluations exist in the field, with the exception of two primary studies that use industrially-scaled examples with sound comparative evaluation.

As for future work, we intend to further investigate the feasibility and practical applicability of applying different approaches to vertical test reuse in embedded systems development in general, and in automotive system development in particular. Further, by means of industrial cases studies and experimentation, we intend to evaluate effects in terms of test efficiency and fault detecting effectiveness of vertical test reuse, as compared to a non-reuse approach.

ACKNOWLEDGMENT

This work was supported by the Swedish Innovation Agency (VINNOVA) through grant 2014-03397 (IMPRINT).

REFERENCES

[1] J. Varnell-Sarjeant, A. A. Andrews, J. Lucente, and A. Stefik, "Comparing development approaches and reuse strategies: An empirical evaluation of developer views from the aerospace industry," *Information and Software Technology*, vol. 61, no. 0, pp. 71 – 92, 2015.

[2] R. Nörenberg, R. Reißing, and J. Weber, "ISO 26262 conformant verification plan," in *8th Workshop on Automotive Software Engineering*, Lecture Notes in Informatics, GI, 2010.

[3] F. Elberzhager, A. Rosbach, J. MüNch, and R. Eschbach, "Reducing test effort: A systematic mapping study on existing approaches," *Information and Software Technology*, vol. 54, no. 10, pp. 1092–1106, 2012.

[4] R. Tiwari and N. Goel, "Reuse: Reducing test effort," *SIGSOFT Software Engineering Notes*, vol. 38, no. 2, pp. 1–11, 2013.

[5] P. Runeson and E. Engström, "Software product line testing – A 3D regression testing problem," in *Proceedings of the 5th International Conference on Software Testing, Verification and Validation (ICST'12)*, IEEE Computer Society, 2012.

[6] E. Engström, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," *Information and Software Technology*, vol. 52, no. 1, pp. 14–30, 2010.

[7] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: A survey," *Journal of Software, Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[8] E. Engström and P. Runeson, "Software product line testing - A systematic mapping study," *Information and Software Technology*, vol. 53, no. 1, pp. 2–13, 2011.

[9] P. A. Da Mota Silveira Neto, I. D. Carmo Machado, J. D. Mcgregor, E. S. De Almeida, and S. R. De Lemos Meira, "A systematic mapping study of software product lines testing," *Information and Software Technology*, vol. 53, no. 5, pp. 407–423, 2011.

[10] I. D. C. Machado, J. D. Mcgregor, Y. a. C. Cavalcanti, and E. S. De Almeida, "On strategies for testing software product lines: A systematic literature review," *Information and Software Technology*, vol. 56, no. 10, pp. 1183–1199, 2014.

[11] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[12] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08)*, British Computer Society, 2008.

[13] A. Marrero Pérez and S. Kaiser, "Bottom-up reuse for multi-level testing," *Journal of Systems and Software*, vol. 83, no. 12, pp. 2392–2415, 2010.

[14] S. Benz, "Combining test case generation for component and integration testing," in *Proceedings of the 3rd International Workshop on Advances in Model-based Testing (A-MOST'07)*, ACM, 2007.

[15] B. Schätz and C. Pfaller, "Integrating component tests to system tests," in *Proceedings of the 5th International Workshop on Formal Aspects of Component Software (FACS'08)*, Elsevier B.V., 2008.

[16] A. Marrero Pérez and S. Kaiser, "Integrating test levels for embedded systems," in *Proceedings of the 2009 Testing: Academic and Industrial Conference - Practice and Research Techniques (TAIC PART'09)*, 2009.

[17] A. M. Pérez and S. Kaiser, "Multi-level test models for embedded systems," in *Software Engineering 2010 - Fachtagung des GI-Fachbereichs Softwaretechnik, 22.-26.2.2010*, 2010.

[18] C. Rodrigues, K. da Silvay, E. Melcherz, J. de Figueiredoz, and D. Guerreroz, "Refactoring VeriSc testbenches to improve the functional verification during the integration phase," in *Proceedings of the 37th Annual Conference on IEEE Industrial Electronics Society (IECON'11)*, 2011.

[19] A. Marrero Pérez and S. Kaiser, "Reusing component test cases for integration testing of retarding embedded system components," in *Proceedings of the 1st International Conference on Advances in System Testing and Validation Lifecycle (VALID'09)*, 2009.

[20] A. Marrero Pérez and S. Kaiser, "Top-down reuse for multi-level testing," in *Proceedings of the 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS'10)*, 2010.

[21] S. Asaithambi and S. Jarzabek, "Towards test case reuse: A study of redundancies in android platform test libraries," in *Safe and Secure Software Reuse* (J. Favaro and M. Morisio, eds.), vol. 7925 of *Lecture Notes in Computer Science*, pp. 49–64, Springer Berlin Heidelberg, 2013.

[22] Y. Choi and C. Bunse, "Design verification in model-based μ-controller development using an abstract component," *Software & Systems Modeling*, vol. 10, no. 1, pp. 91–115, 2011.

[23] S. Schulz, K. Buchenrieder, and J. W. Rozenblit, "Multilevel testing for design verification of embedded systems," *IEEE Design & Test of Computers*, vol. 19, no. 2, pp. 60–69, 2002.

[24] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE'14)*, 2014.

[25] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," *Requirements Engineering*, vol. 11, no. 1, pp. 102–107, 2006.

[26] F. Paterno, *Model-based design and evaluation of interactive applications*. Springer Science & Business Media, 2000.

[27] M. Ivarsson and T. Gorschek, "Technology transfer decision support in requirements engineering research: A systematic review of REj," *Requirements Engineering*, vol. 14, no. 3, pp. 155–175, 2009.

[28] F. Xie and J. Browne, "Verification of component-based software application families," in *Component-based software engineering* (I. Gorton, G. Heineman, I. Crnković, H. Schmidt, J. Stafford, C. Szyperski, and K. Wallnau, eds.), vol. 4063 of *Lecture Notes in Computer Science*, pp. 50–66, Springer Berlin Heidelberg, 2006.