JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
31st December 2018, Volume 3, Issue 3, JSEIS, CADMEI Copyright © 2016-2018
www.jseis.org

SYSTEMATIC MAPPING STUDY OF APPROACHES TO SUPPORT AGILE-INSPIRED EMBEDDED SOFTWARE DEVELOPMENT

[1]DONGCHENG DENG, [1]MICHAEL SMITH, [2]ANDREW F. TAPPENDEN, [3]JAMES MILLER

[1]Department of Electrical and Computer Engineering, Schulich School of Engineering
University of Calgary, Calgary, Canada
[2]Department of Computing Science, The King's University
Edmonton, Canada
[3]Department of Electrical and Computer Engineering
University of Alberta, Edmonton, Canada
Email: andrew.tappenden@kingsu.ca

Academic Editor: Zeljko Stojanov, University of Novi Sad, Serbia.

Published: 31st December, 2018

Funding: The author has no funds.

Article Type: ReviewArticle

ABSTRACT

Since 2002 researchers have investigated the transfer of Agile methodologies into the embedded domain. Specifically, the deployment of an Agile methodology as a defect-reduction strategy to improve the reliability of the software-components of embedded systems. After more than a decade of active research, it is now appropriate to present an overview of the current extent of Agile-inspired embedded software development research (AIESDR). We perform a systematic mapping to categorize and summarize the current status of this area and identify future opportunities in the AIESDR field. We present the results to provide an overview of existing AIESDR and act as a baseline for follow-up AEISDR.

Keywords : embedded software development; agile methodologies; adapted and adopted embedded agile methods; bibliometrics; systematic map;

1. INTRODUCTION

Embedded systems are ubiquitous in today's society with applications ranging from personal and public health to business, public transportation, industry and education. Embedded systems products are characterized as task-specific, highly customized, price and size sensitive. Many embedded systems products are subject to real-time constraints and must meet these stringent requirements on processors with limited computational power and stringent memory requirements. In addition to tight constraints on the computational and power resources, these systems also must functional reliably in harsh conditions for relatively long periods of time with minimum, or no, maintenance and opportunities for updates.

Software defects in business enterprise systems caused an estimated $59.5 billion annual loss of profits in US in 2002, with more than one third of that loss, 22 billion / year, considered as "*reducible with existing best-development practices*" [1]. It is reasonable to anticipate similar losses and possible profit recovery with embedded system software development. In the past decade, the Agile-inspired embedded software development research (AIESDR) field has grown richer and more mature since Grenning [2], Greene [3] and Dahlby [4] first considered applying Beck's Agile methodologies [5, 6] into embedded system development as a pro-active defect reduction process. We believe it is time to systematically identify, categorize, analyze and summarize the state-of-art within the AIESDR field in order to provide both the current *big picture* and future trends.

A systematic literature review study selects and combines research methods, techniques and evidence from available high-quality research papers in order to derive answers to specific research questions. In contrast, a systematic mapping study is a method appropriate to review, analyze and structure papers in a specific research field in order to provide a general overview of the undertaken research [Petersen et al., 2008]. A systematic mapping study is recommended to be used in research areas, such as AIESDR, which lack both primary high-quality studies and cross references between studies [7]. Thus, a systematic mapping study is an effective method for researchers who are interested in gaining a baseline or panoramic view of an emerging research field; potentially prior to undertaking a more comprehensive systematic literature review to obtain a more detailed view based upon specific questions or lines of inquiry [8, 9].

Corresponding Author: Andrew F. Tappenden
Email Address: andrew.tappenden@kingsu.ca

In this paper, we conduct a systematic mapping study to provide a broad overview on research articles employing Agile methods within an embedded software context. This paper is organized as follows: Section 2 provides a brief overview of the similarities and difference between Agile and Embedded development. Section 3 provides a modified software engineering methodology for conducting systematic mapping study. Section 4 provides the two key categories of questions proposed for this AIESDR systematic mapping study, with results presented in Sections 5 and 6. Section 7 gives details of adapted and adopted Embedded-Agile methods and proposed Agile-Inspired embedded lifecycles. Discussions on the findings, trends and implications of the results are presented in Section 8. Finally, Section 9 concludes the paper and proposes future work.

2. TWO RELATED BUT DIFFERENT WORLDS: AGILE AND EMBEDDED

Although embedded systems engineering is far older than Agile [5], both Agile and embedded systems engineering are development processes and can be expected to involve some similar concepts. Embedded system engineering is typically highly-sensitive to the costs of defects entering the field and hence much work is concentrated in this area. Agile, similarly promises *high-quality* software with many processes which either directly or indirectly act as defect reduction strategies. Hence, the question arises: can Agile processes decrease the defect rates found in modern embedded environments?

As will become evident from the results of this survey, Embedded-Agile proponents are particularly enthused with adopting the eXtreme Programming (XP) Agile methodology. Researchers are specifically interested in the concept of using a development process that is "*infected with tests*" [5] as a defect-reduction strategy. XP tests are considered as an executable expression of a user story associated with an item from the customer's product wish-list. Tests are written ahead of the code, not after, when using an XP test driven development (TDD) approach. This write-tests-first strategy has the unique advantage in offering a *"test of Tests"* that is difficult to achieve with a write-tests-last process. An executable is generated with code stubs sufficient to satisfy the linker (or similar compiler architecture) and these stubs are connected to the test code. There is an immediate indication of the lack of proper test coverage. For example, if the "*test of Tests*" is able to execute and passes the tests without the presence of developed code then there is an indication that the testing activities were not adequate. There are other recognized defect-reduction advantages to formalizing the use of a test-first approach. For example, using an XP TDD approach implies that testing is more likely to have occurred before product release if (or when) the project hits scheduling issues; intrinsically leading to lower post-release maintenance costs. Anecdotally we have observed insufficient mentoring within the Embedded-Agile field leading to test development procedures that are often unable to follow the strict TDD test-first protocol. This lack of adherence to the TDD principles is most common with product components with which the developers are unfamiliar. Instead we are willing to accept that an embedded iterative lifecycle with small spirals ending with a required test-last phase and frequent regression testing as a *reasonable* and often *necessary* phase shift from a strict XP TDD approach, and therefore offers most of the same advantages.

To further reduce costs, Agile developers are instructed to *"only write sufficient code, in the simplest manner possible, to satisfy the chosen test"*. In our mind this concept is a re-expression of the old embedded engineering adages relating to "*Keeping it simple (KIS)*" so the developer is more likely to spot obvious mistakes; and *"Give the customers what they really want"* since code that satisfies the tests is also meeting the requirements expressed through those tests. There are other advantages of the Agile re-expression of the *KIS* idea. (1) A developer producing code that "*just satisfies*" a test is not boosting product costs by introducing unnecessary features; (2) It is simpler for the developer to demonstrate *"the next small and quick product increment"* (MMF / MVP rather than prototype) at a *Scrum* meeting with code written simply for its functionality.

It therefore becomes appropriate for the Agile developer to (A) perform *"continuous integration",* merging small, rather than large, pieces of code into the existing project; and (B) "*refactor"* the code for "*maintainability and an appropriate level of quality"* [10] and (C) perform *"frequent regression testing"* by using tests developed during the initial *TDD* stage, to ensure that modified and integrated code still meets the original customer requirements. These Agile aspects are impossible to handle efficiently without automated tool support. The lack of suitable tool support is a frequent expressed concern by embedded developers attempting to become Agile. Currently supported Agile tools may not be transferable because of memory constraints or be inappropriate in the new environment where the embedded software must be moved in a reliable fashion from simulation onto a production target [11-13]. There the migrated / refactored code may closely interact with new hardware conFigureurations requiring that the standard Agile *"refactoring code for maintainability"* may have to be replaced by *"refactoring code for speed*" to meet stricter timing requirements (real-time restrictions) than for enterprise systems, and *"refactoring code for lower power consumption"* may be necessary for mobile device developers [14].

3. SYSTEMATIC MAPPING METHODOLOGY

In this section, we first review the differences between systematic mapping studies and systematic literature review studies and then present the systematic mapping process used in this study.

Differences between systematic mapping and systematic literature review studies

Literature reviews, as a type of secondary research, have two major underpinnings, systematic literature reviews and systematic mapping studies, which differ in many traits [8, 9].

Goal: Systematic literature review involves in-depth inspection of high-quality comparative papers to answer specific research questions. Systematic mapping studies are designed to provide an investigation of all research publications, regardless of quality, in a certain research area to obtain a summarized state-of-art of that field. Thus, results of systematic mapping studies are mostly statistics on various dimensions of relevant facets of the research field, whereas systematic literature review results are aggregated evidence on a certain method / technique.

Research Questions: Systematic mapping studies look at broad generic questions, e.g. how many papers proposed a new method A or tool B. This contrasts to systematic literature reviews which ask specific questions based on empirical evidence and / or a comparative study of experimental results; e.g. is method C better than D?

Depth versus breadth: A larger body of papers is covered via a systematic mapping study as the process is deliberately designed to cover everything related to a field. Despite more papers being discovered, a systematic mapping study is designed to be completed quickly with only abstracts and keywords being examined.

Searching, inclusion / exclusion strategies and threats to validity: Systematic literature reviews exclude papers that are low quality or not highly related to a specific method/technique of a certain research field to make sure that the review is evaluating the best quality results. In a newly emerging field, such as AIESDR, there will not yet be expert consensus on what method or tools have high- or low-quality. Even what constitutes a relevant topic within the area will be unclear at a certain level. Therefore, it is appropriate for systematic mapping studies to use general keywords during a search to provide a broader range of results. However, the need to use general search terms during systematic mapping studies to identify a larger quantity of papers that might (possibly) have relevant content must be anticipated to result in some incorrect inclusions or wrong classification of papers. The newness of the AIESDR field, and associated lack of general acceptance, must be considered a possible threat to the validity of a systematic mapping study. In Section 5, we will provide validity statistics to evaluate the possible impact of issues such as (i) the inconsistent use of Agile terms, and (ii) the deliberate use of non-AIESDR keywords in publication titles and abstracts as authors attempt to gain acceptance in established journals and conferences that do not normally cover the combined area of Agile and embedded system development.

Modified systematic mapping process

We have found it necessary to add a step to the systematic mapping methodology for software engineering topics proposed by Petersen et al. [9] in order to undertake this Embedded-Agile study.

1. The research questions are used to define the study's research scope.
2. Search for all possible studies before applying screening. The proposed research questions and field of study should guide the search strings developed.
3. Develop and apply an established set of inclusion and exclusion criteria to better identify those papers properly within scope of this new research area.
4. Perform key-wording. This is a systematic approach to develop a classification scheme which takes all existing studies into account [9]. In this step, the reviewers read the abstract to identify keywords and concepts that characterize the paper's contribution and identify the context of the research. The final set of keywords is then clustered to form categories (facets), and groups of categories.
5. *New Step:* Extended key-wording or quasi-systematic literature review. We attempted to follow the suggestions of [9] to develop the classification scheme by investigating the abstract of the studies to reduce the time and effort required. However, when the abstract of a study was not well-formed for Agile-related keywords or functionality to be recognized, we examined the introduction, conclusion and sometimes the body of the study to gain a better understanding of the proposed concepts.
6. With the classifications established, the included papers are sorted into a scheme and a table of frequencies for each keyword discussed. This permits the identification of which categories and keywords have been emphasized by past researchers, with the gaps indicating possibilities for future research.

4. PROPOSED PROCESS FOR AIESD SYSTEMATIC MAPPING

4.1 Research questions

We have generated two main categories of AIESDR research questions:

1. Bibliometric analysis and demographic trends (RQ1) following [15] to understand the depth of study in this new field, and
2. systematic mapping research questions (RQ2) as suggested by [9].

To have superior fine-grained views on these two research questions, we further divide them into sub-questions:

Bibliometric and demographic trends (RQ1)

By asking the following research questions, a bibliometric and demographic analysis of the research field is presented. Assessing the bibliometric and demographic trends can assist in identifying the best practice, papers, researchers and venues for follow-up research [15, 16]. Answers to the RQ1 group of questions results are given in Section 5.

RQ1.1: Year of publication: What is the volume of publication across each year in last decade? We map and crosslink, this research question with various RQ2 sub-questions in order to see possible AIESDR trends.

RQ1.2: Article type classification: Is the article a conference paper, journal paper, technical report, thesis, book or magazine article? Since the research community of AIESDR is a relatively new research area, we want to know what article types has been favored by the researchers and developers.

RQ1.3: Venue counts: Which venues (conference, journals and etc.) are popular for papers from AIESDR? This provides information for searching for existing AIESD papers or identifying publishing areas receptive to this new mixture of Agile and embedded development ideas.

RQ1.4: Most cited AIESDR papers: Absolute citation numbers and normalized citation numbers (average citation count since the paper was published) are presented and ranked.

RQ1.5: Observed challenges to validity: As discussed in Section 3, the procedures used in systematic mapping studies must be anticipated to lead to possible threats to the validity of the results generated [8, 9]. Accordingly, we discuss the validity threats identified as an aid in designing future systematic mapping studies and possible follow-up, more in-depth literature reviews.

Key AIESD research areas (RQ2)

We identified the key questions on AIESDR field as:

1. Which of the Agile methods have been transferred from the enterprise world to embedded systems software development?
2. Which Agile methods have been tailored for AIESD use?
3. What tools and methods are available to developers who want to adopt an Agile approach to embedded software development?
4. Are case studies and other approaches available to provide a level of mentoring for new Embedded-Agile developers in tool use and method application?
5. Which facets of Agile are being ignored, or under-utilized by developers and researchers seeking improved development approaches in a safety critical field?

These questions are translated into research questions as follows with answers to this category of research questions given in Section 6.

RQ2.1: Research type: Is the paper a validation, evaluation, solution, philosophical, opinion or experience paper [17]? This research question is general and independent of the specific AIESDR research area. The rational for this question is to classify papers into different research methods and identify which research methods have been focused upon or neglected. The research types were treated as mutually exclusive, meaning a paper can only belong to one research type. No AIESDR validation research types were found. The solution research type covers all studies proposing a new solution.

RQ2.2: Research contribution: What type of contribution does the paper offer to the research? As suggested by Petersen et al. [9], it is important to know whether the study proposes a new tool, a process to implement a

method, a method to solve a problem, a model (example) of the method, or provides a metric to measure results / techniques or evaluation results of a practice. A given paper can provide multiple contributions.

RQ2.3: Adapted Agile methods / TDD used: Identification of papers that specifically discuss TDD for use in an embedded software development context. This question arose from a review of discovered studies in which we found that papers in AIESDR are heavily TDD focussed. The reason that TDD has gained much attention in AIESDR is because TDD is an actual development practice; it is easier to be tailored and adopted onto the embedded software development and the benefits of performing TDD is more verifiable [5, 18-22]. Therefore, we propose that papers that discuss TDD be discussed separately through this research question. These papers are mapped to reflect the trend, research type and contribution to provide a clear indication on how TDD has been modified to suit the embedded software development and the tools available to foster the adoption of TDD.

RQ2.4: Domain of study: Onto what embedded system domains are the Agile methods applied? This research question provides a general idea of which domain of study the paper focuses on, allowing readers to identify papers in a specific field of interest, e.g. medical embedded products, embedded system education, control system or general embedded software.

RQ2.5: Case study included: Does the paper include a case study section? Case studies provide important empirical evidence for researchers and developers to evaluate the results of a certain method or technique. Equally important, case studies are a route for mentoring developers who want to move in an Embedded-Agile direction.

RQ2.6: Functionality associated with proposed AIESDR tools: Tools are suggested to enable or provide better features on: unit test, regression test, acceptance test, mock generation (using software to mimic the behavior of existing or planned hardware and/or software; e.g. simulate a return value of a proposed sensor), build utility (e.g. automatically link the required objects), model-based testing, refactoring, integration testing, version control and image processing for medical devices software development. We classify the tool functionality purely based on what the paper said; evaluation of the tool functionality was considered out of the scope of this systematic mapping study.

4.2 Search approach

As a relatively new area of study, we recognize that AIESDR embedded aspects are not being generally accepted by main stream Agile developers, while AIESDR's software engineering aspects are not generally accepted by main stream Embedded Developers. We therefore considered it appropriate to investigate possible matches from small industrial conferences, personal blogs, Agile-related websites in addition to those from the main stream scientific literature. To meet this requirement, the search strings associated with Embedded-Agile concepts and application areas were presented to the following data bases and search engines:

a) IEEE Xplore, ACM Digital Library, Springer Link, Elsevier;

b) Google Scholar and Microsoft Academic Search; and

c) Google.

The search strings used for this initial overview of the two worlds of Agile and embedded systems were related to key Agile concepts (Agile, XP, scrum, TDD and Lean) and general embedded systems related terms (embedded, embedded software, biomedical, military). Biomedical and military devices are embedded products and developers of them might be interested in investigating alternative development techniques to reduce the number of introduced defects and/or prevent introducing new defects into these products which require utmost reliability.

We conducted an initial systematic mapping study during the period September 2011 to December 2011 and updated the results in October 2013. Given the lack of direction available from any previous AIESDR secondary studies, the newness of the field and the perceived lack of cohesiveness across the field, we chose the following search strings for this initial AIESDR systematic mapping. The use of these strings with the Google database, intended to identify possible 'non-academic' articles such as blogs, generated a large number of possible matches. As this was an overview using a systematic mapping study approach, rather than a detailed literature search, only the first 500 Google hits generated by the general search strings were reviewed in detail.

> (Agile OR "extreme programming" OR XP OR scrum OR "test driven development" OR TDD OR Lean) AND (embedded OR "embedded software" OR biomedical OR military OR army)

An analysis of possible validity challenges and advantages incurred by using such general strings are discussed in Section 5. By investigating the paper's title and abstract, we narrowed the search results down to 171

studies. These studies, starting in 2002, were in form of books, papers, blog articles, theses, technical reports and commercial magazine articles.

4.3 Screening of papers and key-wording of abstracts

As a standard approach to include and exclude the studies from the systematic mapping study pool only the abstract of the paper is evaluated, which is a major difference from systematic literature review studies [9]. The criteria of inclusion and exclusion applied onto the studies' abstract are summarized in Table 1. As a result, 78 studies are included in this systematic mapping study. We used the term included, selected or discovered studies to refer to these studies interchangeably. For space and readability issues in Figureures and tables, we have chosen to display the discovered publications using a S-number reference notation, for example [S1], [S2], … [Sxx]. All discovered papers are detailed in Electronic Appendix A.

Table. 1 Inclusion and exclusion criteria for screening of publications

Inclusion:	Studies in any form (books, conference / journal papers, blog articles and etc.) concerning the application of Agile methods onto embedded system software development. Specially, studies about biomedical and military engineering using Agile methods are also included.
Exclusion:	Studies that are out of the scientific and engineering domain or did not focus on the discussion of adapting or adopting Agile methodologies to embedded system software development. Studies that were not written in English were excluded. Only the latest study was included if two or more studies were considered to contain too many similarities.

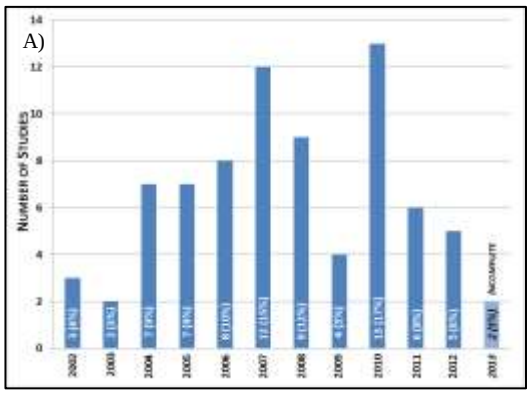
Key-wording is a systematic approach to extract the primary traits from all existing studies in order to classify and structure them. We developed the classification scheme by mainly investigating the abstract of the studies as suggested in [9]. When the abstract of a study was not well-formed for Agile-related keywords or functionality to be recognized, we reviewed the introduction, conclusion and sometimes the body of the study to gain a better understanding of the study, a quasi-systematic literature review. The classification scheme developed was presented in the research questions (Section 4.1).

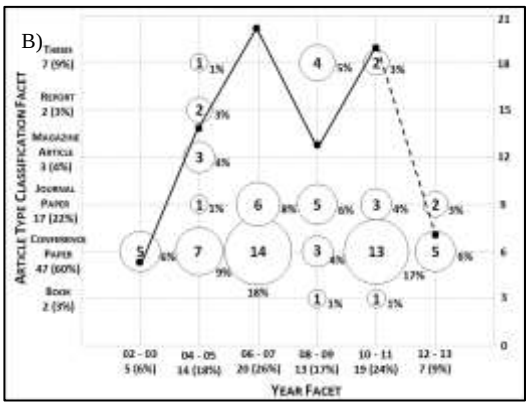
5. BIBLIOMETRIC AND DEMOGRAPHICS TRENDS (RQ1)

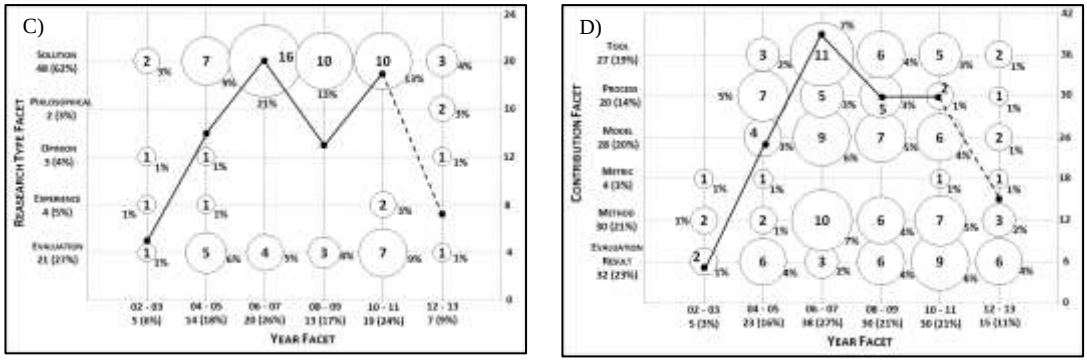
RQ1.1: Year of study publication facet

In this research question, we mapped the year of publication facet to RQ1.2, RQ2.1 and RQ2.2 to provide an overview of the trend of the AIESDR research community with two consecutive year's group as one category in Figures. 1B to 1D. The shaded column in Figure. 1A and dotted lines in Figures. 1B to 1D reflect the fact that the review will not have included all the papers accepted for publication in 2013.

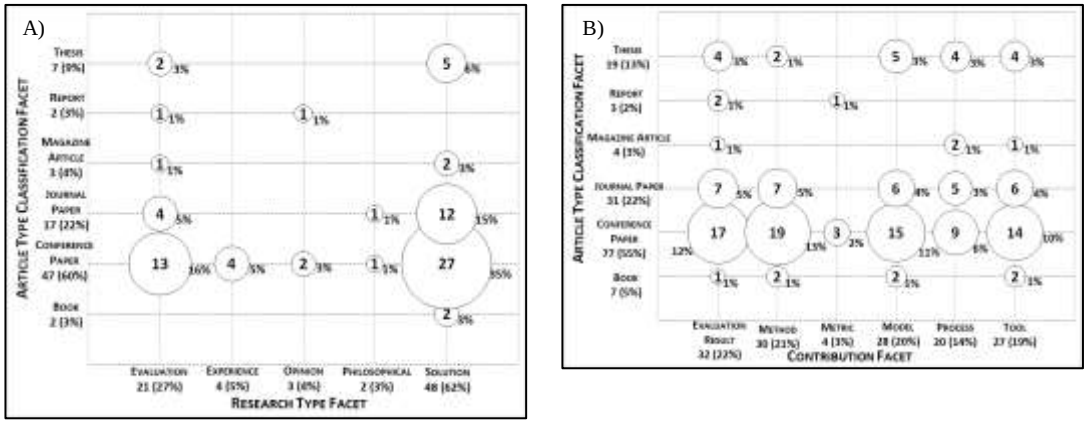
Two publication peaks are shown, 2007 and 2010 in Figure. 1A. Figure. 1B maps the year of publication and article type facets to present the trend of different publication types. Figure. 1C shows that evaluation studies have a relatively flat publication count across the years whereas the solution studies have a peak in 2006 / 2007 and had high publication levels from 2004 to 2011. We anticipate that the number of philosophical studies will continue to grow as authors are recognize that it is time to identify, structure and summarize the current existing outputs from research studies. Figure. 1D maps the year of publication and contribution facets to show the changes of contribution made by AIESDR papers across the years.







Figureure. 1 A) histogram of year of study publication. relationships of the year of study publication facet with the B) article type, C) research type and D) contribution facets are shown. details of the selected studies used to generate the Figureures in this survey are given in electronic appendix b.



Figureure. 2 Relationship between A) research type and B) contribution facets with the article type classification facet.

RQ1.2: Article type classification facet

Since AIESDR is a relatively new research area, we categorize different article types in order to see the structure of current available studies. Figure. 2A and 2B show the mapping of research type and contribution facets against the article type classification facet respectively. Conference papers and journal papers contribute the greatest number of papers (60% and 22% respectively in Figure. 2A) and most contributions (55% and 22% respectively in Figure. 2B).

Table. 2 Venues with at least 2 publications (ranked by number of publications)

Venue Title	Type	Publication Count	Citation Count
Thesis	Thesis	7	3
XP Int. Conf.	Conference	6	63
Embedded Systems Conf.	Conference	6	40
Agile Conf.	Conference	4	80
Int. Conf. on Computer Software and Applications	Conference	3	14
Circuit Cellar Magazine	Magazine	3	3
IEEE Software Magazine	Journal	2	19
Journal of Signal Processing Systems	Journal	2	4
Proceedings of European Conf. on the Use of Modern Information and Communication Technologies	Conference	2	2

RQ1.3: Venue counts

This RQ revealed that 50 different venues / publication sources were used for the 78 studies in our pool. This diversity reflects the perceived general usefulness of the Embedded-Agile idea in different domains; e.g. biomedical, control systems, education etc. However, we do not think that such a scattering is necessarily a good

phenomenon. Table 2 shows that only 8 venues have at least 2 publications which we feel reduces the ease with which collaboration and knowledge distribution and mentoring can occur amongst of the AIESDR community. As will be discussed further in Section 5.5, the wide range of venues that AIESDR researchers are attempting to target for their publication presents a challenge to the validity of this systematic mapping study.

RQ1.4: Most cited papers

We used two metrics to measure and identify the most cited (popular) papers in the pool: (1) the absolute (total) number of citations of the paper, and (2) the normalized number of citations of the paper i.e. the average citation count of a paper per year since its publication year; this metric is defined is [15] as:

$$\boldsymbol{NormalizedCitations} = \frac{\boldsymbol{AbsoluteCitation}}{\boldsymbol{Review\ year\ -PublicationYear+1}}$$

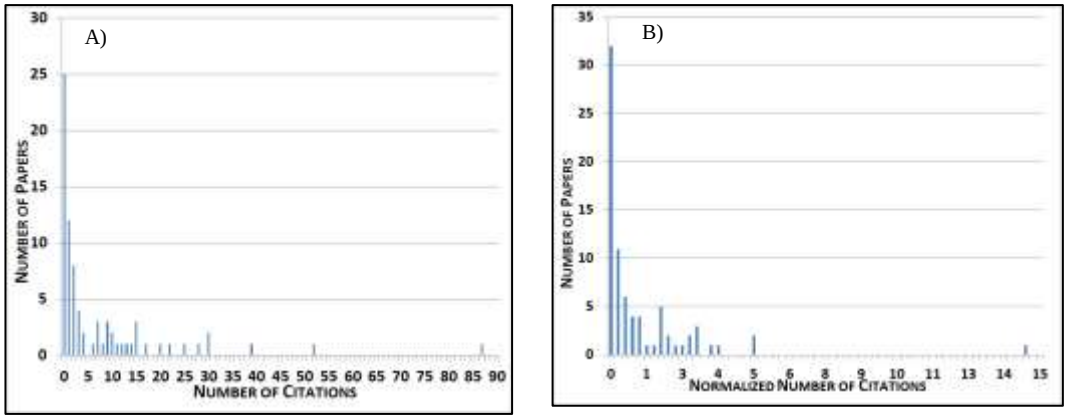
For example, [S57] with 87 absolute citations, published in 2008 and reviewed in year 2013, has a normalized citation count of $\frac{87}{2013-2008+1}$ = 14.50

According to Table 3, the most cited papers are [S57], [S43] and [S19] based on both total citation and normalized citation numbers. Salo et al. [S57] conducted a survey based on 13 industrial companies in 8 European countries and 35 individual embedded software development projects which used Extreme Programming and Scrum. Using Google Scholar to generate citation metrics on this industrial study without specific filtering for Embedded-Agile concepts showed a large number of citations in a relatively short period of time from both inside and outside the AIESDR field.

Instead of simply moving the full Agile techniques onto the embedded software development, Manhart et al. [S43] specified that integrating TDD into some classical software engineering processes, fostered the acceptance of Agile techniques. This was among the first papers (published in 2004) which promoting the adoption and adaption of TDD. Greene [S19] worked for the *Intel Corporation* when they were developing the firmware for the Intel® Itanium® processor family in 2004. Figure. 3A and 3B illustrate the Histogram of number of citations and normalized number of citations for all selected studies.

Table. 3 The top 20 studies (ranked by absolute citation number)

Study No.	First Author	Year	Citation #	Normalized Citation #
[S57]	O. Salo	2008	87	14.50
[S43]	P. Manhart	2004	52	5.20
[S19]	B. Greene	2004	39	3.90
[S38]	P. Kettunen	2008	30	5.00
[S56]	J. Ronkainen	2003	30	2.73
[S74]	N. Schooenderwoert	2004	28	2.80
[S34]	D. Kane	2006	25	3.13
[S20]	J. Grenning	2002	22	1.83
[S37]	P. Kettunen	2005	20	2.22
[S64]	M. Smith	2005	17	1.89
[S15]	B. Douglas	2009	15	3.00
[S38]	M. Karlesky	2007	15	2.14
[S35]	M. Karlesky	2006	15	1.88
[S58]	J. Savolain	2010	14	3.50
[S21]	J. Grenning	2007	13	1.84
[S71]	J. Srinivasan	2009	12	2.40
[S6]	O. Cawley	2010	11	2.75
[S14	T. Dohmke	2008	10	1.67
[S48]	J. Miller	2007	10	1.43
[S24]	J. Grenning	2011	9	3.00



Figureure. 3 Histogram of number of A) Citations and B) Normalized number of citations for all discovered studies

RQ1.5: Observed challenges to validity

Systematic mappings are, by their nature, a contradiction in terms. They are intended for a broad overview of a newly emerging field with no consensus yet available on what to consider as relevant expert knowledge for generating appropriate inclusion and exclusion criteria for search strings or even for identification of the extent of the domain. We attempted to mitigate these issues by

1. Reviewing and adapting systematic mapping paper approaches from other software engineering fields [8, 9, 15, 23];
2. Presenting our findings to an experienced industrial practitioner with 15 years of experience in the area of practical application of Agile development processes in a number of fields [10];
3. Applying our own research and industrial experiences.

In this study, we deliberately used very general Agile terms (Agile, extreme programming, scrum and test-driven development) to develop the systematic mapping search strings for this initial overview of AESDR concepts. According to guidelines of conducting systematic mapping in software engineering domain [9], only the abstract of the paper needs to be reviewed to decide whether to include or exclude the paper and extract the keywords for use in the classification scheme. While this approach is commonly used to reduce the time and effort to generate a systematic mapping, it is obvious that such studies could be biased by including / excluding a paper incorrectly or biased by not correctly identifying all possible variations of a set of keywords. As discussed earlier, we found it necessary to undertake a quasi-systematic literature review approach and use the study's introduction, body and conclusion if the abstract was too vague.

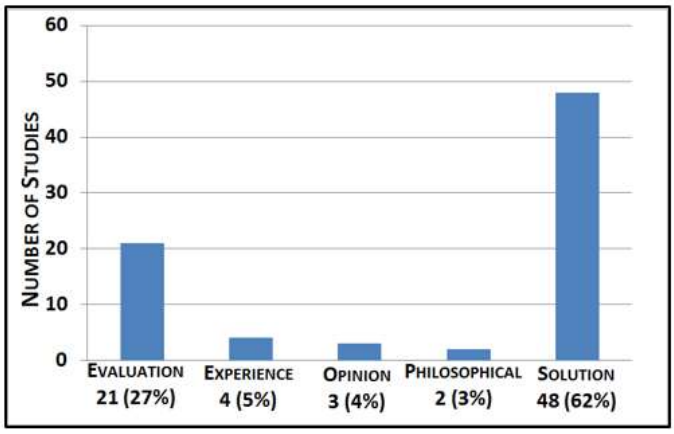
To gain some idea of the extent of this inherent systematic mapping problem (i) we identified how many papers that we personally knew had been published but this systematic study had failed to discover and (ii) we attempted to identify why these papers remained undiscovered despite the application of very general search terms. The undiscovered paper [24], and related blogs, focuses more on Agile-inspired integrated circuit development of hardware or AIICD related to system on a chip design. The undiscovered papers [13, 25] cover concepts in TDD and code coverage tools whose implementation was made possible by custom use of hardware features built into existing embedded system processors. These hardware units designed into the processor core are not regularly available to Enterprise system developers but are a fundamental tool in the embedded developer's arsenal. However, these papers had abstracts intentionally targeted towards acceptance by a main stream software engineering conference not normally identified as covering embedded system issues. The title and the abstract of the undiscovered [26] paper was also tailored to emphasize that paper's general software engineering aspects rather than stressing Agile embedded concepts. We consider that a key reason for these studies remaining undiscovered in this systematic mapping as a direct consequence of the current difficulty in publishing AIESDR ideas in the main stream literature as discussed in the early section on venues (*RQ1.3*).

Given, that our "extended search" identifies only another 4 articles, we remain confident that "most" contributions are present in the survey. We suggest that attempting to uncover further missing studies by performing a further systematic mapping using new search terms is inappropriate. Instead we propose an extension to the modified systematic mapping process discussed in Section 3.2. This new step involves performing a quasi-systematic literature review of papers identified in the references for a particular AIESDR facet present in this paper's Figures. We provide a list of our discovered studies relating to each Figureure in electronic appendix B to facilitate this process.

6. SYSTEMATIC MAPPING RESULTS (RQ2)

RQ2.1: Research type facet

Figure. 4 shows the distribution of the research method used. One study can only belong to one research type. This biases the solution type towards having a higher occurrence as we classify a paper to be of this research facet if it proposes a solution, regardless of whether it also evaluates, gives opinions and provides experience to the solution.



Figureure. 4 Histogram of research type facet

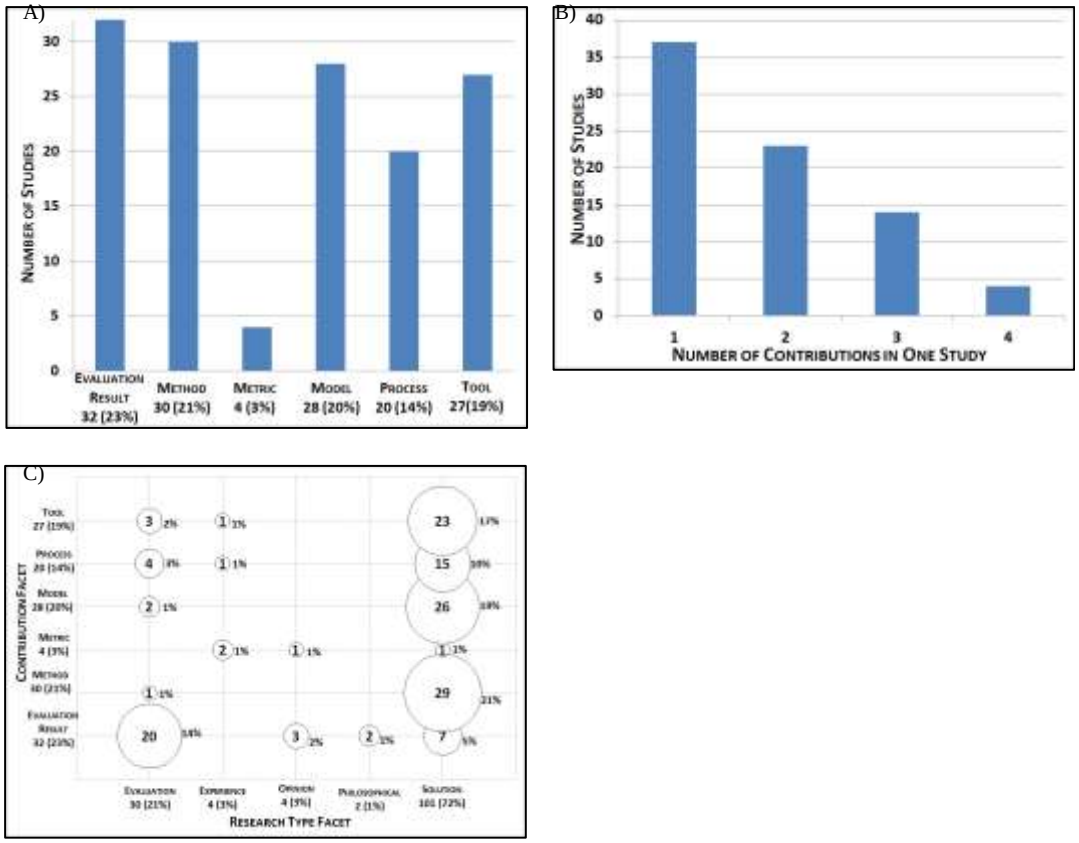
There are 21 studies (27%) that evaluate the applicability of well-established Agile methods in the context of embedded software and 48 studies (62%) are proposing new methods / solutions to tailor the Agile methods for use within embedded software development. We believe that the dominance of these two research types (69 papers, 89% in total) is an indication that Agile methods are appealing to researchers and industry people in embedded software development domain as they have shown their desire to assess or tailor these methods. However, this also indicates that the desktop / web application oriented Agile methods may still needs special consideration when they are applied in an embedded context. A few studies, Cordemans et al., Shen et al. and Albuquerque et al. (S2, S9, S59) discuss or summarize the existing proposed new solutions and provide evaluation results of the AIESDR field. Antonio et al. [27] presented a systematic mapping study on embedded software architectures unrelated to Agile methodologies. Xie et al. [28] generated a systematic mapping study of embedded software development. None of these studies generated a panoramic view of AIESDR to determine to what extent embedded system developers and researchers have responded to an observation by Smith et al. [14] which we paraphrase as:

"It is one thing to adapt an Agile process, such as method or tool, for embedded system development; but an entirely different thing for the actual philosophy of Agile concepts to be adopted by embedded system developers".

We suggest that a growth in flagship studies and high-quality secondary studies would increase the acceptance of AIESDR ideas amongst both software engineers and embedded developers.

RQ2.2: Contribution facet

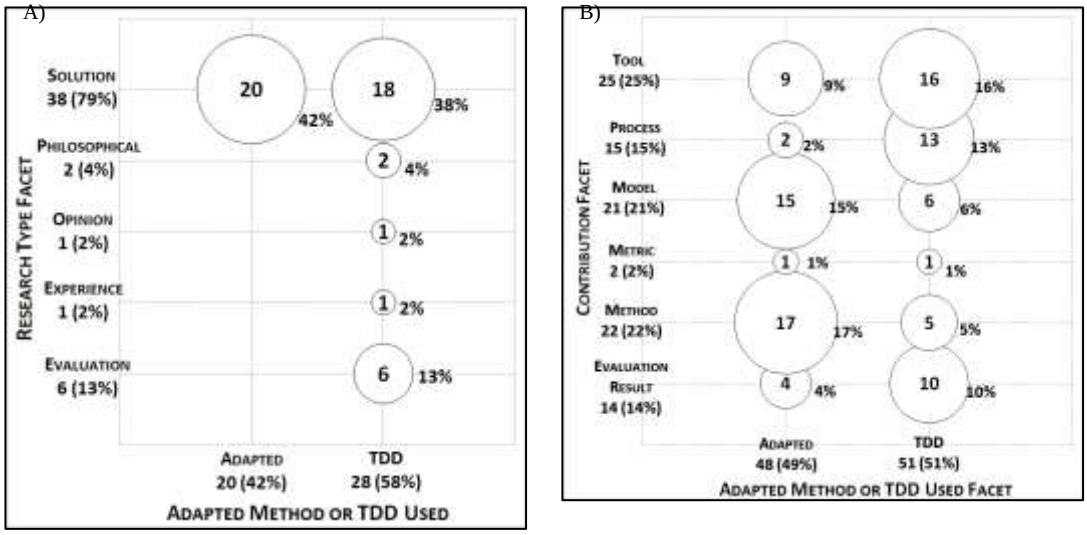
Figure. 5A shows the distributions of contributions that the pool of studies has offered. One study can offer one or more contributions and therefore the total amount of the contributions is larger than the number of studies. For example, [S67] made 3 contributions: (1) proposed tools for unit test and acceptance test; (2) a new solution (method) on how to tailor the Agile methods into the embedded software development cycle; and (3) detailed steps (models) on instructing people how to apply the new method. The number of contributions per study is shown in Figure. 5B so that the papers with most contributions can be recognized.



Figureure. 5 Histograms of A) Contribution facet and B) number of contributions in one study. C) Research Type facet mapped against the Contribution facet

New tools or new use of old tools were proposed in 27 out of 78 studies (35%). This implies: (1) developers can actually take advantage of Agile more easily when tools are available to support the use of Agile methods within Embedded Software Development; while (2) inadequate tool support remains an obstacle that limits the applicability of Agile methods in this new environment [S12] [S70]. We encourage researchers and industrial developers to continue developing / proposing new tools or new use of tools to enable us to become more Agile.

Figure. 5C shows 78 (55%) contributions associated with new processes, methods and models. Evaluation results are presented in 32 out of the 78 contributions (41%) which indicates that there are few evaluations of the 101 solution studies (72%). More model studies (examples on conducting a process or method) would be useful as they provide mentoring opportunities to provide the guideline, opinion and experience of utilizing Agile methods in an AIESDR context. Studies with new metrics to evaluate methods seem to have experienced less attention. The systematic study indicated that there are no controlled experiments e.g. studies that compared the cost, development speed, defect rate or developers / customers' satisfaction, etc. between Embedded-Agile and the traditional embedded development lifecycles (water-fall and V-shaped). The philosophical studies are systematic literature reviews published in 2012.



Figureure. 6 Systematic mapping of A) research type and B) Contribution facets against the adapted methods/TDD used facet

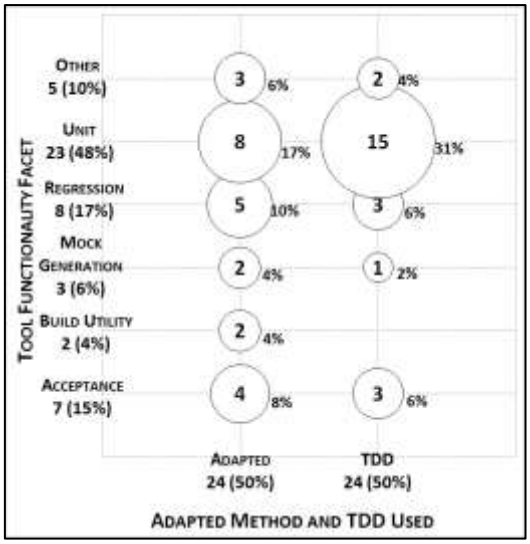
RQ2.3: TDD and other non-TDD adapted agile methods

In this research question, we investigated the TDD specific and non-TDD specific Agile methods adapted for the Embedded-Agile context identified within our pool of studies as proposed in Section 4.1. Figure. 6A maps the adapted method and TDD used facet against the research type facet. According to Figure. 6A, there are 48 papers (62% out of 78 discovered papers) that discussed adapted methods and TDD. To tailor Agile methods to become more suitable to embedded software development, 20 out of 78 studies (26%) propose adapted methods. Papers that propose new adapted methods are classified as solution papers, while papers that discussed TDD have varieties of research types. Solution research type (79%) has the greatest number of adapted methods / TDD papers. Figure. 6B maps the contribution facet against the adapted methods / TDD used facet. According to Figure. 6B and Figureure 5C, there are 48 papers that discussed adapted methods and TDD contribute 99 contributions (70%) out of 141 contributions in 78 papers. Metrics on how to evaluate the effectiveness and efficiency of TDD and adapted Agile methods are needed.

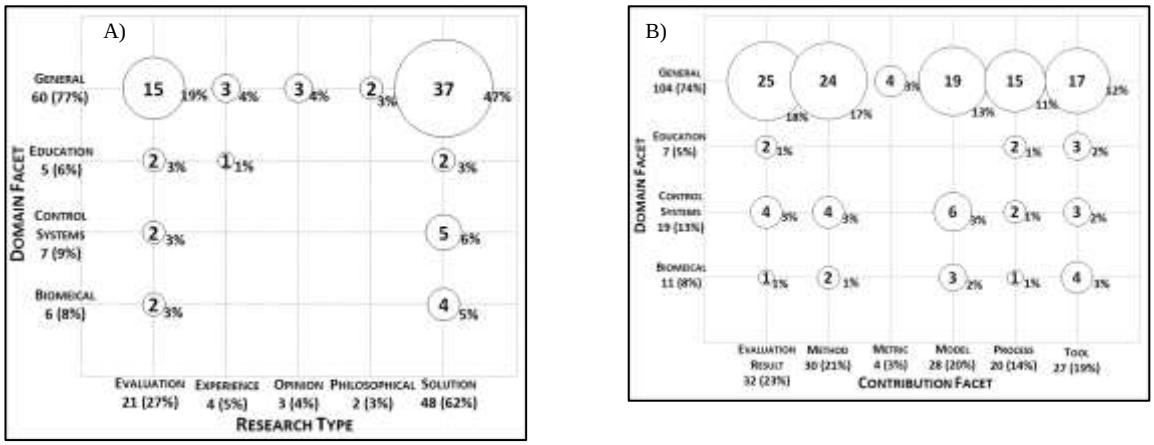
As can be seen in the Figure. 6B, studies discussing TDD proposed more processes (concrete steps that can be followed) on how to conduct TDD on the embedded content while the studies discussing adapted Agile methodologies proposed more methods on how to apply Agile methods in an innovative way. Only 4 evaluation results are provided for the adapted Agile methods, indicating the lack of investigation on new adapted methods. Adapted Agile methods, on the other hand, provided more method contributions (how to solve a particular problem) than process contributions (concrete steps on how to implement a method). More than 50% (25 out of 48 papers) of the adapted Agile methods and TDD papers propose new tools or old tools used in a new way.

RQ2.4: Tool functionality facet

Because of space limitations we group the tool functionalities of model-based testing, refactoring, integration testing, static code analysis, test insertion, image processing and version control as other in Figure. 7. We found no study proposing a tool to help Embedded-Agile refactoring. There are 25 contributions offering 48 tool functionalities according to Figure. 6B and Figure. 7.



Figureure. 7 Systematic mapping of adapted methods / TDD used facet and tool functionality facet.

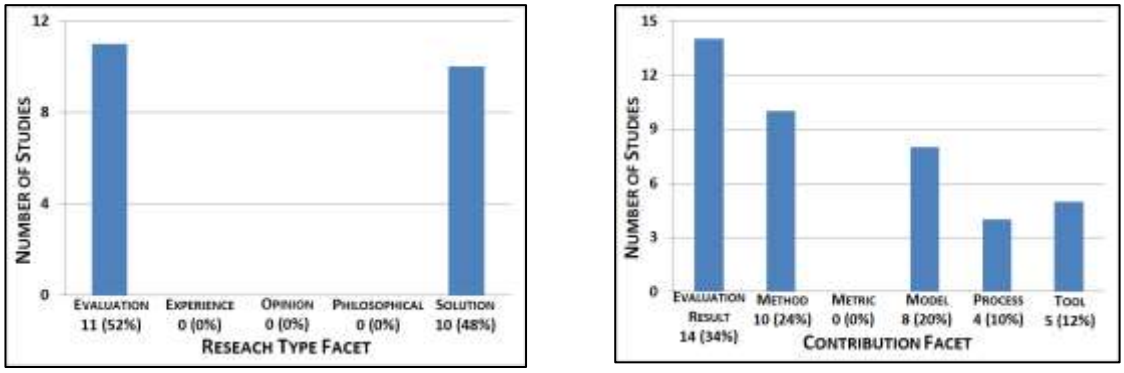


Figureure. 8 Systematic mapping of A) research type and B) contribution facets against the domain facet.

RQ2.5: Domain of study facet

Figure. 8A maps the domain of study against the research type. Studies are categorized to a specific domain only if they mention the domain they investigated in the keywords or in the abstract. Otherwise we categorize the study into the general domain. Figure. 8B presents the systematic mapping of contribution facet against the domain facet.

RQ2.6: Case study included (CSI) facet



Figureure. 9 Histograms of A) Research type and B) Contribution facets that included case studies.

We considered the inclusion of case studies as the first step towards providing the mentoring needed so that a newly adapted processes becomes an adopted process. 21 out of 78 contributions (27%) offer case studies in evaluation and solution studies as shown in Figure. 9A, a histogram of the CSI facet against the research-type facet. It is to be expected that there be no case studies for opinion and philosophical contributions since these either express personal opinion which do not rely on any work of others or are secondary studies. Figure. 9B presents the contributions made by the papers that included case studies.

7. ADAPTED AND ADOPTED AGILE PRACTICES AND TOOLSETS

According to RQ1.3 and RQ1.5 results, AIESDR studies have been scattered into large numbers of venues (only 1.56 studies / venue), which increases the difficulty of researchers in taking advantage of available AIESDR. Therefore, we believe it is appropriate to go beyond the concepts of a systematic mapping study to provide limited details of proposed testing frameworks, tools and AIESDR life-cycles.

Testing frameworks

EMBEDDED-UNIT: Unit Test Tool for Embedded Software Development. Embedded-Unit is a unit testing tool based on CppUnitLite [29] and updated to use the more extensive UnitTest++ [Llopis and Nicholson, 2013] testing frameworks. Modifications have been applied in order to meet the memory and speed constraints for embedded systems; i.e. this tool can run directly on evaluation boards and product targets in addition to simulators. Processor on-chip timers are used to aid with the non-functional testing needed for real-time code. A recovery mechanism allows testing to continue when prototype software / hardware interfaces fail [S10, S48, S55, S63, S64, S67, S69].

SYSTIR: System Test Framework. Systir helps introduce input to a system and compare the output to which is expected in system tests. Systir allows system tests to include code that binds to libraries across a variety of languages of user defined domain specific languages [S36].

UNITY: Unit Test Framework for C. Unity is a lightweight testing framework for C. It can output test results through standard I/O (*STDIO*), a serial port, or via simulator output [S36].

Embedded-agile support tools

C-MOCK: Mock Function Library for C. C-MOCK is a Ruby-based tool to automate the creation and maintenance of mock functions for unit testing in C-based systems [S36].

E-COVER: Code Coverage Analysis Tool. TRACE Unit is a full code coverage analysis tool. Speed is improved over software code-coverage tools by following the program flow between the tests and code by utilizing the hardware present in current modern processors [Tran, 2008]. This approach has recently been modified for automated requirements traceability [Wiederseiner et al, 2011].

E-FIT / E-FITNESSE: Acceptance Test Cases Generation Tool. This tool is intended to assist customers express their requirements in terms of acceptance tests and follow successful product development through all five stages of the eXtreme Programming Inspired Embedded-Agile lifecycle. [S7, S67].

E-RACE: Data race identification tool in multi-threaded systems using shared memory. To reduce the chances of false positive and false negatives, E-RACE avoids the overhead of software data race tools by employing the processor's data watch and instruction watch units to monitor memory activity [S68] [Huang et al., 2008]. Recently an FPGA-based Agile test support (ATS) co-processor has been suggested to provide equivalent support across all families of processors [S12, S70].

FUXI: is described as an Agile Development environment for embedded systems [S76].

XEST: is described as an automated framework for regression testing of embedded software [S51].

Proposed embedded-agile lifecycles

Grenning's Embedded TDD lifecycle [2] provides a detailed overview of moving embedded code from simulation, through prototyping on an evaluation board onto testing the production model, see Table 5. The eXtreme Programming Inspired (XPI) embedded cycle [S67] covers an Embedded-Agile approach to (1) customer acceptance tests [S7], (2) research into, and development of new algorithms using MATLAB [S55] that are moved into (3) simulation before (4) optimization for real-time operation on the target [S10] [S64] [S67]. Recent extensions have been suggested to allow an Agile approach (5) to the development of FPGA co-processors to handle critical real-time issues or to identify data-races in multi-threaded systems [S12, S70].

Table. 5 Comparison of proposed embedded-agile lifecycles

Embedded TDD (ETDD) lifecycle [S21]				**eXtreme Programming Inspired (XPI) Embedded-Agile lifecycle [S7, S10, S12, S55, S64, S66, S67, S70]**		
Stage	USES	PRODUCES		Stage	USES	PRODUCES
				XPI Stage 1 Initial interaction with customer	User stories and wish lists	Acceptance tests expressed via FIT and FitNesse tools; Excel spreadsheet
				XPI Stage 2 Discussion with research team	XP1–tests; new algorithm concepts	Unit tests expressed in MATLAB language
ETDD-Stage 1 C simulation	Customer wish list	Unit tests running in PC development environment.	- MORE FREQUENT TESTING - LESS FREQUENT -	XPI Stage 3 C Simulation	XP1-tests, XP2-tests	Unit tests running using target's compiler using a target simulator or on evaluation board
Stage 2 Compile tests for target	ETDD – 1 tests	Unit tests running using target's compiler on target simulator				
Stage 3 Run test on evaluation board	ETDD 1-2 tests, hardware interface	Additional unit tests associated with hardware interaction		XPI Stage 4 Optimization to meet real-time requirements	XP1-tests, XP2-tests, XP3-tests, real time requirements, hardware interface	Unit tests running on evaluation board or target. Optimized C / C++ code or custom assembly code
Stage 4 Run tests on target hardware	ETDD1-ETDD3-tests	Tests that run on target hardware				
Stage 5 Manual testing	ETDD1- 4 tests	Tests that run on target hardware with manual intervention				
				XPI Stage 5 Movement to co-processor to meet real-time requirements	XPI 1 - tests, XPI 2 - tests, XPI 3 - tests, XPI 4 - tests	Unit tests mocking the co-processor in software. Unit tests with actual co-processor implemented with mocked and actual busses [S12, S70]
				Possible XPI Stage 6 IC development Proposed in [Johnson, 2011]		

Johnson discussed attempts to apply Agile concepts to IC development [Johnston, 2005] at a Calgary Agile Methodologies User Group (CAMUG) informal monthly meeting. Modifications to XP processes to handle chip development and simulation formed the majority of that talk.

A key element in using Agile at all stages during embedded system development is the ability to write tests that validate the hardware, even when the hardware does not yet exist. A testable design can be achieved by using a Model, Conductor and Hardware (MCH) design pattern [S35, S36] which isolates the functional logic from hardware.

8. DISCUSSIONS ON FINDINGS, TRENDS AND IMPLICATIONS

Based on the searching string (mostly the combination of Agile-relevant words and embedded-relevant words) applied, 171 studies were found and 78 studies of them are included to perform this systematic mapping study. Although we expressed the difficulty in finding the papers (50 venues for 78 papers), this number is comparable to other recent systematic mapping studies in the software engineering domain: e.g. [30] with 45 included papers; [23] with 64 included papers; [31] with 33 included papers and [15] with 79 included papers.

RQ1.1 presents the trend from the beginning the AIESDR research field, 2002, to the most recent discovered paper in 2013. Figure. 1A reveals that a general increase of number of papers from 2002 to 2007 and there are two publication peaks in 2007 and 2010. From Figure. 1B, conference papers are the most common contribution to the AIESDR community with two publication peaks of 2006 to 2007 and 2010 to 2011. Evaluation result show two steady increases: from 2002 to 2005 and from 2006 to 201, Figure, 1D. We believe the first increase is because the flagship people want to popularize the use of Agile and the second as more and more people began to evaluate the adoption and adaptation of Agile onto the embedded software development domain. RQ1.2 maps the article type classification facet to research type facet and contribution facets. Combining the mapping result of Figures. 2A and 2B, book and thesis type of studies increased (3% and 9% of publications contributing 5% and 13% of contributions) with other types of studies contributions remaining approximately in the same proportion.

We presented the top venues in Table 2 in RQ1.3 and the citation count and normalized citation count for each paper are shown in Table 3 in RQ1.4. The most popular papers can be extracted from Table 3 easily and the distribution of number of citations and normalized citations are revealed in Figures. 3A and 3B, respectively. The absolute citation and the normalized citation numbers of our study were less than those presented in [15]. We believe that this is a further indication of the research work of AIESDR is more isolated than within other research fields leading to less cross references. We highly encourage researchers and industrial developers to synchronize the existing research fruit into their current research and work. RQ1.5 investigated and determined the possible impact of observed threats to the validity of our study.

There were a large number / percentage of solution papers proposed, according to Figure. 4, RQ2.1. This Figureure suggested two things:

1. researchers are very enthusiastic to propose new method / solutions to better apply Agile methods onto embedded software development.
2. some Enterprise-Agile methods did not appear to easily transfer without special care, modifications or new methods and tools.

We expect that solution papers will keep growing in the future. However, the generation of evaluation, opinion or experience researches on the new solutions still requires more attention in order to synchronize the current research results. Philosophical studies, as a type of secondary study, are highly encouraged since it reviews and summarizes a number of papers in the field to provide the big picture.

Solution-related attributes (e.g. tools, processes, methods and models) are frequent contribution, Figure. 5A, RQ2.2. However, additional secondary studies are still required to assess which are the better toolsets and practices for the AIESDR community. New metric methods would encourage evaluations on new solutions. We believe the reason that TDD, RQ2.3 is favored for embedded software development is because: (1) TDD is supposed or proven to be a defect-reduction practice in a number of AIESDR and non-AEISDR communities; (2) TDD is a practical programming practice that can be actually followed and performed. Evaluation papers are needed on the adapted methods, Figure. 6B.

RQ2.4 investigates the tool functionalities presented in the included studies. Figure. 7 indicates that discussions on unit testing tools are frequent, but there are no embedded tool specifically supporting the refactoring often associated with TDD. We believe the lack of such discussions is because major embedded development environments already possess some form of feature that would support the standard aspects of Agile refactoring [CrossCore® Embedded Studio, accessed, 2013; Code Composer Studio, accessed 2013; Atmel Studio accessed 2013 and VisualDSP++, accessed 2014). However, according to Smith et al. [Smith et al. 2009, Smith et al., 2013], refactoring in AIESDR life-cycles has many different meanings, and basic refactoring tools are not suitable to assist in the transfer of verified, high-level (platform independent) and sometimes slow code to a faster or cheaper version processor.

RQ2.5 categorizes the pool of studies into various domains. The gaps in Figure. 8A and 8B, represent missing bubbles for opinion papers for control system, biomedical and education domains. This research question groups studies in the same domain for the use of people who are interested in one specific AIESDR domain. RQ2.6 presents the statistical results of papers that included case studies by research type and contribution facets respectively but there are no proposed metrics to validate the case studies, Figure. 9B.

By going beyond the scope of a standard systematic mapping study into a more quasi-systematic literature study, testing frameworks and supporting tools from the pool of studies are investigated in Section 7.1 and 7.2. Section 7.3 presents two adapted Embedded-Agile lifecycles. Smith et al.'s XPI embedded life cycle [S7, S10, S12, S55, S64, S66, S67, S70] captures the whole embedded software development from developing requirements with customers in the form of acceptance tests to transferring memory-heavy or time-consuming C/C++ code from embedded processors to task-specific co-processors for speed or lower cost. Grenning's embedded TDD lifecycle [S21] provides more detail in handling the equivalent of XPI stage 3 and 4, which are product development stages with emulator / actual hardware.

9. CONCLUSION AND FUTURE WORK

Researchers have proposed a number of reasons and methods on why and how Agile approaches can offer benefits on reducing defect rates in embedded software development. However, the lack of synchronization and summarization of existing research prevents the embedded software development group utilizing the full advantages available with the adaption and adoption of Agile methodologies. In this paper, we presented a systematic mapping study on papers that investigated adopting or adapting Agile methods into embedded software domain in order to offer an overview in this research field. The study found 171 studies of potential interest; and after exclusion criteria were applied, 78 studies are identified which were published between 2002 and 2013. To identify, categorize and summarize the current status of AIESDR research area, a set of classification schemes has been generated. Papers were then sorted into the schemes and data are extracted from the pool of studies. Data is presented in the form of mappings to answer each research question. Questions covering bibliometric and demographic trends, year of publication, article type classification, top venues, most cited papers and active researchers are presented. In addition, questions on the research type, contribution, adapted Agile methods including TDD, tool functionalities, domain of study, whether case studies are included, and article type of the paper are shown via systematic mapping. Furthermore, by going beyond the basic scope of systematic mapping, the adapted and adopted Agile methods are presented and discussed covering three aspects: testing framework, support tools proposed and embedded Agile lifecycle.

Our study indicates that AIESDR research area is a hot topic but with little synchronization and summarization. By classifying the pool of studies into different categories, the characteristics of the AIESDR field can be identified. Solution-proposing paper are favoured and making up the majority of contributions. However, evaluation, opinion and experience papers on these new solutions are needed. Such areas are examples of a guideline to assist researchers in conducting future research in this area, which is one of the main objectives of this study.

ELECTRONIC APPENDICES

The electronic appendixes for this article can be accessed in the University of Calgary, PRISM, Digital Library.

Appendix A: Discovered Studies [S*xx*] for Systematic Mapping Study.

Appendix B: Discovered Studies [S*xx*] Associated with Each Figureure.

ACKNOWLEDGEMENTS

The authors would like to thank S. Choudhury (University of Calgary, Canada) and A. Geras (Calgary) for assistance in preparing and reviewing the manuscript.

REFERENCES

1. Tassey, G., *The economic impacts of inadequate infrastructure for software testing. National Institute of Standards and Technology, 2002.* Forschungsbericht (Zitiert auf Seite 2), 1996.
2. Grenning, J. *Extreme programming and embedded software development*. in *Embedded Systems Conference*. 2002.
3. Greene, B. *Agile methods applied to embedded firmware development*. in *null*. 2004: IEEE.
4. Dahlby, D., *Applying agile methods to embedded systems development.* Embedded Software Design Resources, 2004. 41: p. 1014123.
5. Beck, K., *Test-driven development: by example*. 2003: Addison-Wesley Professional.
6. Beck, K., *Extreme programming explained: embrace change*. 2000: addison-wesley professional.
7. Barbara, K. and S. Charters, *Guidelines for performing systematic literature reviews in software engineering.* Keele University, UK, 2007. 9.
8. Kitchenham, B.A., D. Budgen, and O.P. Brereton, *Using mapping studies as the basis for further research–a participant-observer case study.* Information and Software Technology, 2011. 53(6): p. 638-651.

9. Petersen, K., et al. *Systematic Mapping Studies in Software Engineering*. in *EASE*. 2008.
10. Geras, A., *Agile Software Engineering Practioner*. 2012, Personal Communication: Calgary, Canada.
11. Grenning, J., *Applying test driven development to embedded software.* IEEE Instrumentation & Measurement Magazine, 2007. 10(6).
12. Miller, J., et al. *An XP inspired test-oriented life-cycle production strategy for building embedded biomedical applications*. in *Testing: Academic and Industrial Conference-Practice And Research Techniques, 2006. TAIC PART 2006. Proceedings*. 2006: IEEE.
13. Tran, A., M.R. Smith, and J. Miller, *A Hardware-Assisted Tool for Fast, Full Code Coverage Analysis, 19th International Symposium on Software Reliability Engineering*, in *19th International Symposium on Software Reliability Engineering, ISSRE 2008*. 2008: Seattle, USA.
14. Smith, M., et al., *A more agile approach to embedded system development.* IEEE software, 2009(3): p. 50-57.
15. Garousi, V., et al., *A systematic mapping study of web application testing.* Information and Software Technology, 2013. 55(8): p. 1374-1396.
16. Garousi, V. and T. Varma, *A bibliometric assessment of canadian software engineering scholars and institutions (1996-2006).* Computer and Information Science, 2010. 3(2): p. 19.
17. Wieringa, R., et al., *Requirements engineering paper classification and evaluation criteria: a proposal and a discussion.* Requirements engineering, 2006. 11(1): p. 102-107.
18. Williams, E.M.M.a.L. *Assessing test-driven development at IBM*. in *25th International Conference on Software Engineering*. 2003.
19. Beck, C.A.a.K., *Extreme programming explained: embrace change*. 2004: Addison-Wesley Professional.
20. Nagappan, T.B.a.N. *Evaluating the efficacy of test-driven development: industrial case studies*. in *In Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. 2006.
21. Szała, L.M.a.L., *2007*. Software Process Improvement, The impact of test-driven development on software development productivity—an empirical study: p. 200-211.
22. Nschiappan Nagappan, E.M.M., Thirumalesh Bhat and Laura Williams, *Realizing quality improvement through test driven development: results and experiences of four industrial teams.* Empirical Software Engineering, 2008. 13(3): p. 289-302.
23. Runeson, E.E.a.P., *Software product line testing–a systematic mapping study.* Information and Software Technology, 2011. 53(1): p. 2-13.
24. Johnson, N., *Applying Agile to IC Development... We're Not That Different After All*. 2011, CAMUG - Calgary Agile Methods User Group.
25. Fuang (Lily) Huang, M.S., Albert Tran and James Miller. *E-RACE, A Hardware-Assisted Approach to Lockset-Based Data Race Detection for Embedded Products*. in *19th International Symposium on Software Reliability Engineering, ISSRE*. 2008. Seattle, USA.
26. Christopher Wiederseiner, V.G., Michael Smith. *Tool Support for Automated Traceability of Test/Code Artifacts in Embedded Software Systems*. in *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2011.
27. E. A. Antonio, F.C.F.a.S.C.P.F.F. *A Systematic Mapping of Architectures for Embedded Software*. in *IEEE Second Brazilian Conference on Critical Embedded Systems (CBSEC)*. 2012.
28. Mingjuan Xie, M.S., Guoping Rong and Dong Shao. *Empirical studies of embedded software development using agile methods: a systematic review*. in *In Proceedings of the 2nd international workshop on Evidential assessment of software technologies*. 2012.
29. Feathers, M. *Cpp Unit Lite 21*. 2011 [cited 2013 September 2013]; Available from: http://c2.com/cgi/wiki?CppUnitLite.
30. Neto, P.A.d.M.S., et al., *A systematic mapping study of software product lines testing.* Information and Software Technology, 2011. 53(5): p. 407-423.
31. Palacios, M., J. García-Fanjul, and J. Tuya, *Testing in Service Oriented Architectures with dynamic binding: A mapping study.* Information and Software Technology, 2011. 53(3): p. 171-189.

AUTHORS PROFILE