Changelog		
24/01/2013	Updated RQs, changed document Structure (last part), Updated example of Process Management analysis	npaternoster
12/01/2013	Example of Analysis of Work Practices (Process Management)	cgiardino
07/01/2013	Updated Introduction	cgiardino
24/12/2012	Updated Introduction, Abstract, Document Structure, Tables with work practices extraction	npaternoster, cgiardino
07/12/2012	Creation of first draft	npaternoster, cgiardino

Table 1: Changelog

Contents

Software development in startup companies: A systematic mapping study

Nicolò Paternoster[a], Carmine Giardino[a], Michael Unterkalmsteiner[a], Tony Gorschek[a]

[a]*Software Engineering Research Lab, Blekinge Institute of Technology, SE-371 79 Karlskrona, Sweden*

Abstract

Context: Software startups are newly created companies with no operating history and extremely fast in producing cutting-edge technologies. These companies develop software under highly-uncertain conditions, tackling fast-growing markets under severe lack of resources. Therefore, especially in the early stage, software startups present an unique combination of characteristics which pose several challenges to software development activities.
Objective: This study aims to structure and analyze the literature of software development in startup companies in order to identify the quality and understand the main work practices reported from practitioners and researchers.
Method: A systematic mapping study has been performed applying a wide-range of related search terms to key electronic databases and selecting relevant papers. We created a systematic map by applying a classification schema, ranked the selected papers according their rigor and relevance, and analyzed the reported main work practices.
Results: A total of 37 studies were selected, classified and evaluated from an initial set of 943 articles. Only 14 papers are entirely dedicated to software development in startups, and 9 of those produced a weak contribution (advice and implications (5); lesson learned (3); tool (1)). Sixteen studies are focused on managerial and organizational factors. Moreover the studies are generally not rigorously designed and cover only small samples of startups.
Conclusion: The mapping study provides the first systematic summary of the literature. Software development in startup companies offers researchers an overview of the current state-of-the-art. Although the results attest an increased interest in the multifaceted challenges posed by startups in recent years, they reveal several gaps that need to be addressed.

Keywords: Software Development, Startups, Systematic Mapping Study

1. Introduction

A wide body of knowledge has been created in recent years through several empirical studies, investigating how companies leverage software engineering (SE) [1]. Nevertheless, all the software development activities, critical for the survival of a newly created company, are not thoroughly investigated by the existing publication fora. In fact, very few publications have identified, characterized and mapped work practices in software startups [2].

Hence, understanding how startups take advantage from work practices is essential to support the number of new businesses launched everyday[1]. New software ventures such as *Facebook*, *Linkedin*, *Spotify*, *Pinterest*, *Instagram*, and *Dropbox*, to name a few, are good examples of startups that evolved into successful businesses. Startups typically aim to create high-tech and innovative products[2], and grow by aggressively expanding their business in highly scalable markets.

Despite many successful stories, the great majority of startups fail within two years from their creation, primarily due to self-destruction rather than competition [4]. Operating in a chaotic, rapidly evolving and uncertain domain, startups face intense time-pressure from the market and are exposed to tough competition [5, 6]. In order to succeed in this environment, it is crucial to choose the right features to build and be able to quickly adapt the product to new requests constrained by very limited amount of resources [2].

From a software engineering perspective startups are unique, since they develop software in a context where processes can hardly follow a prescriptive methodology [7]. Startups share some characteristics with other domains such as small companies and web engineering, present a combination of different factors that makes the development context different from established companies [8]. Indeed, more research is needed to support their engineering activities guiding practitioners in taking decisions and avoiding choices that could easily lead business failure [9, 10].

To understand the quality and main contributions of the state of the art regarding software engineering in startups, in this study we perform a systematic mapping aiming to:

- Identify the publication fora and quality.
- Understand how authors report characteristics of startups.
- Extract and discuss work practices identified by authors.

The systematic mapping study has allowed us to outline the nature and extent of the research conducted in software development in startups. We analyzed 943 articles, where 37 articles have been selected used for analysing the study rigor and relevance and classifying existing work practices reported by researchers and practitioners. We identified 14 papers are entirely dedicated to software development in startups. Nine of those produced a weak contribution (advice and implications (5); lesson learned (3); tool (1)). Sixteen studies are focused on managerial and organizational factors. Furthermore, results are generally not rigorously designed, unspecific, or even contradictory. ¡INSERT HERE ONE OR TWO KEY ANALYSIS POINTS ON WORK PRACTICES¿

Companies and practitioners may take advantage of the analyzed work practices and researchers may focus on exploiting weak and un-

[☆]Blekinge Institute of Technology

[1]According to a recent study, solely in the US *"startups create an average of 3 million new jobs annually"* [3].

[2]In this study we use the term *"product"* for both software products and software services.

charted research topics revealed by the mapping review of the existing studies.

The remains of this paper is structured as follows: in Section we do this in Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do thatin Section we do that.

2. Background

Here we should define

- established companies
- software-intensive
- software development and
- differences between sw.dev in startups and established.
- Mention web and small.
- engineering activities ,
- process
- work practices

Since studying entrepreneurship requires multidisciplinary competences, software development in startups cannot be seen as a single unit. In order to apply a proper analysis it is required to master several concepts.In this chapter we provide a basic knowledge of the concepts listed above, grounding our research work in the surrounding context.

2.1. Startup and Entrepreneurship

Modern entrepreneurship, which was born more than thirty years ago [11], has been boosted by the advent of the consumer internet markets in the middle of the nineties and culminated with the notorious *dot-com bubble burst* of 2000 [12]. Several years later, with the massification of the internet and mobile devices, we are now assisting to an impressive proliferation of software ventures - metaphorically referred as the *startup bubble*. The easy access to vast potential markets and the low cost of services distribution are appealing condition for modern entrepreneurs [13]. Inspired by stories of overwhelming successes, a large number of software businesses are created every day. However, the great majority of these companies fail within two years from their creation [4]. Just by looking at the number of new business incubators which appeared in the last three years one can evaluate the importance of startups [14]. The wave of disruption in new technologies has led companies to be more and more competitive, forcing themselves to radical organizational and innovational renewals, which bring many companies to the attempt of behaving like startups [15].

In view of the lack of agreement on an unique definition of the word *startup* (see Section **??**).

2.2. Software Development in Startups

The implementation of methodologies to structure and control the development activities in startups is a major challenge for engineers [7]. Generally, the management of software development is achieved through the introduction of software processes, which defines what steps the development organizations should take at each stage of production and provide assistance in making estimates, developing plans and measuring the quality [16]. In the last decades, several models have been introduced to control software development activities. However, their application in startup companies doesnt report significant benefits [17, 7, 2].

In such context, software engineering (SE) faces complex and multifaceted obstacles in understanding how to manage development processes. Sutton defines startups as creative and flexible in nature and reluctant to introduce process or bureaucratic measures which may hinder their natural attributes [2]. Also Bach refers to startups as *"a bunch of energetic and committed people without defined development processes"* [18]. In fact, startups have very limited resources and typically wish to use them to support product development instead of establishing processes [7, 19]. Some attempts to taylor lightweight processes to startups, reported basic failure of their application : *"Everyone is busy, and software engineering practices are often one of the first places developers cut corners"* [20]. Rejecting the notion of repeatable and controlled processes, startups prominently take advantage of unpredictable, reactive and low-precision[3] engineering practices [2, 22, 23, 24].

Moreover, as a matter of fact, most startups develop packaged applications rather than software for a specific client [25]. Issues related to this domain are addressed in literature by the area known as *market-driven software development* (sometimes called *packaged software development* or *COTS software development* [26]). Among other results, researchers emphasize the importance of time-to-market as a key strategic objective [27, 27, 28] for companies operating in this domain. Other peculiar aspects which influence the software development are related to requirements, which are reported to be often "invented by the software company" [29], "rarely documented" [30], and can be validated only after the product is released to market [31, 32]. Under these circumstances, failure of product launches are largely due to "products not meeting customer needs" [26].

Accordingly, product-oriented practices help startups in having a flexible team, with workflows that leave them the ability to quickly change the direction according to the targeted market [19, 2]. At this regard, many startups focus on team productivity, asserting more control to the employees instead of providing them rigid guidelines [22, 23, 24]. Despite some studies tried to address the above mentioned issues, from the systematic review of the literature we found only a few SE works in this specific area, as confirmed by other studies [7, 17, 33, 2]. Moreover the studies, identified in our systematic review, appear to be highly fragmented and spread across different areas rather than constituting a consistent *body of knowledge* (see Secion **??**). Notwithstanding, a new interesting area of the SE research, trying to tackle the problem of the *technical debt*, seems to bring interesting implications in studying development in software startups. The metaphoric neologism was originally introduced by Cunningham in 1992 [34] and has recently attracted the attention of SE researchers[4]. The concept of *technical debt* is well illustrated in [38] : *"The idea is that developers sometimes accept compromises a system in one dimension (e.g., modularity) to meet an urgent demand in some other dimension (e.g., a deadline), and that such compromises incur a "debt" on which "interest" has to be paid and which the "principal" should be repaid at some point for the long-term health of the project"*. The compromise between high-speed and high-quality engineering is faced daily by software startups, not only in terms of architecture design but in multifaceted aspects (weak project management, testing, process

[3] The term *"low-precision"* has been derived from [21].

[4] To attest the fact that *technical debt* is gaining traction among researchers, we mention two important contributions which characterize the *"debt landscape"* : [35, 36] and a dedicated workshop [37] organized by the Software Engineering Institute and ICSE.

control, . . .).

3. Research methodology

For the review of the existing literature we chose the SMS, which is one of the most appropriate methodologies capable in dealing with wide and poorly-defined areas [39, 40]. A more traditional *Systematic Literature Review* (SLR) [40] would have been a less viable option due to the wide breadth of the research problem and an apparent lack of a strong academic support material (differences between SLR and SMS are thoroughly discussed by Petersen et. al. in [39]).

A first preliminary literature survey in the SE databases revealed a quite wide gap in works addressing our research problem but at the same time it showed us how broad the domain was. In fact, understanding the general approach undertaken by software startups to develop software requires us to explore different research areas.

One of the most appropriate methodologies, which helps researchers to investigate wide and poorly defined fields, is the *Systematic Mapping Study* (SMS), also known as *Scoping Study* [40]. Despite SMS has its roots in medical research, the growing body of knowledge of SE and the tendency towards evidence-based approaches [41] made SMS a technique increasingly adopted by software engineers [42]. SMS provides systematic guidelines to classify existing works obtaining an overview of the area, which can be easily transferred to other researchers and startups practitioners.

A definition which confirms the suitability of this methodology is provided by Kitchenham et al. in [40]: *"SMSs are designed to provide a wide overview of a research area, to establish if research evidence exists on a topic and provide an indication of the quantity of the evidence.[. . .]"* - and the authors continue - *"[. . .] with the aim of influencing the future direction of primary research."*

3.1. Research Questions

- RQ-1: What is the state-of-the-art in literature pertaining to software engineering in startups?
  - RQ-1.1: What are the common publication fora?
  - RQ-1.2: What is the quality of the selected studies?
    - RQ-1.2.1: What is the scientific rigor of the studies reported in literature?
    - RQ-1.2.2: What is the industrial relevance of the studies reported in literarature?
  - RQ-1.3: How is software engineering characterized in software startups?
  - RQ-1.4: What are the main work practices reported in association with software engineering within startups?

//define terms

3.2. Research process overview

For executing the *Systematic Mapping Study* we scanned studies throughout the scientific databases following the process suggested by Petersen et al. in [39] as shown in Figure 1, where the upper blocks represent an *action* and the underlying block represents its *output*.

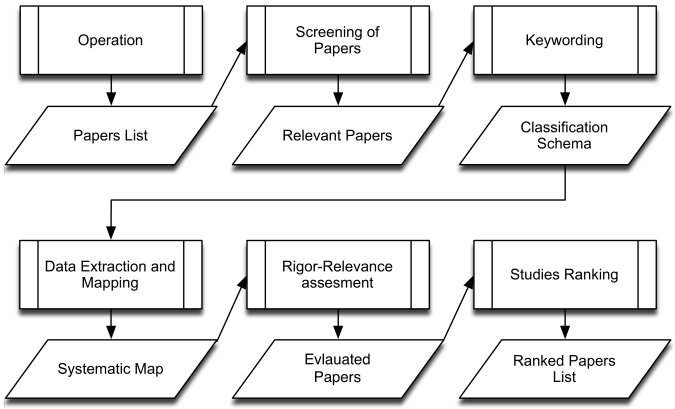

Figure 1: Process overview adapted from [39]

Starting from the research problem defined in **??**, the first step of our systematic study was *Operation*. The whole SMS procedure has been executed simultaneously in pair on the same screen, handling conflicts at the end of the *data extraction and mapping* process by reviewing the rationale of similar decisions taken during the *screening of papers*. When necessary we performed an in-depth review of the article[5].

3.3. Retrieving and screening articles

TThe first step for conducting a systematic search was iteratively building a *search string* composed by different *keywords* that emerge by reviewing the scope of the research questions. To reflect our wide research problem we needed a very broad *search string*, which was generated from three *core concepts* identified as fundamental to our problem domain, i.e.:

- Software startup.
- Development.
- Process.

Following Rumseys guidelines [43], for each core concept we identified synonymous, related concepts, broader concepts, wider concepts, alternative spelling and part of speech that contributed to elaborate the list of terms included in our search string.

The second step was the identification of relevant and significant databases to include in our search process. To increase the internal validity of our research and the chances of finding relevant material we included in our target 5 of the most important peer-reviewed scientific online databases, namely:

- Compendex/Inspec [44].
- ACM Digital Library [45].
- IEEE Xplore [46].
- ISI Web of Science [47].
- Scopus [48].

To be more confident of the results, we included in the set of databases *Google Scholar* [49], which indexes an extremely large set of data, both peer and non-peer reviewed. All the mentioned databases were selected in view of their ability to handle advanced queries.

[5] If the conflicts persisted after an in-depth review of the article, the third and fourth author of this paper... -/ take the final decision.

We then proceeded in customizing the *search string* and iteratively perform the search while refining search *keywords*. The overall process we used is shown in Figure 2.

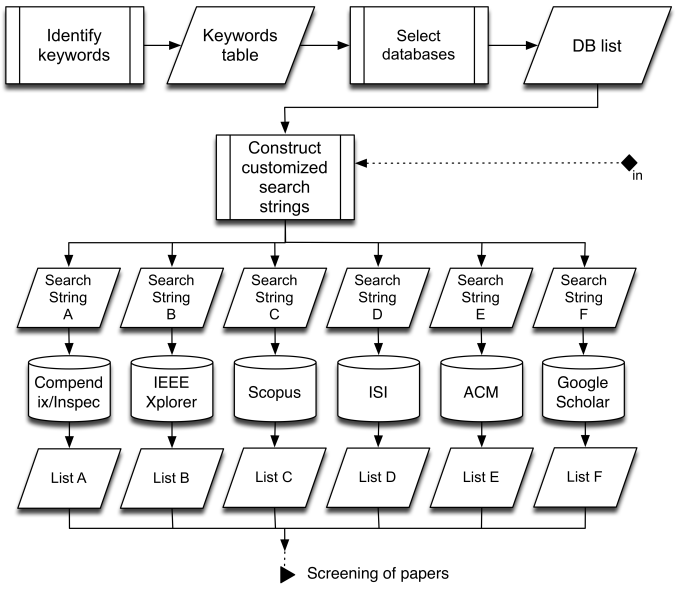

Figure 2: Systematic Mapping Study - Operation

The procedure established in the figure above produces six different outputs - one per database - which are further analyzed as shown in Figure 3. The terms which formed our final search string are shown in Table 2, next to the core concept they represent[6].

Table 2: Mapping Study - Search String Keywords

Core Concept	Terms
Software Startups	software startup*; software start-up*; early-stage firm*; early-stage compan*; high-tech venture*; high-tech start-up*; start-up compan*; startup compan*; lean startup*; lean start-up*; software package start-up*; software package startup*; IT start-up*; IT startup*; software product startup*; software start up*; internet start-up*; internet startup*; web startup*; web start-up*; mobile startup*; mobile start-up*;
Development	develop*; engineer*; model*; construct*; implement*; cod*; creat*; build*;
Strategy	product*; service*; process*; methodolog*; tool*; method*; practice*; artifact*; artefact*; qualit*; ilit*; strateg*; software;

In forming our *search string*, detailed in Appendix **??** (Table .14), terms belonging to different core concepts were linked with the logical *and*, while terms on the same row have been linked with the logical *or*.

After executing the search strings on each database we saved the bibliographic references in 6 separated *BibTeX* files, containing a total number of 1417 items, coming from different databases as shown in Table 3.

[6]The symbol * indicates the use a wildcard, mainly for including plural forms and alternative spellings.

Table 3: Retrieved papers source overview

List	Database	Items
List A	Inspec/Compendex	508
List B	IEEE Xplore	104
List C	Scopus	357
List D	ISI Web of Knowledge	219
List E	ACM Digital Library	71
List F	Google Scholar	158
	Total:	**1417**

To execute the screening of papers we defined and followed a rigid step-by-step workflow, shown in Figure 3.

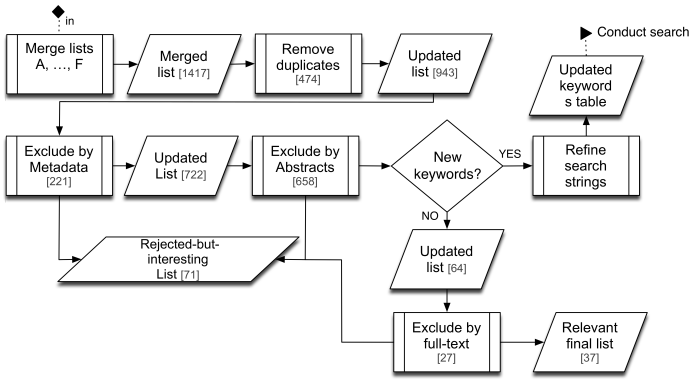

Figure 3: Screening of papers

With the support of a tool for bibliographic management [50] we merged the six lists together and removed the duplicate items in two distinct steps: first we used an automated feature to detect and remove items based on the author, year and title of the publication. Then, we manually deleted other duplicates that were not detected by the tool, for a total number of 474 duplicates.

Before proceeding with further exclusions we defined our *inclusion/exclusion criteria*, aiming to include only peer-reviewed papers which brought some contribution to the body of knowledge of software engineering, by explicitly mentioning results related to software development in startups. Additionally we decided to reject articles with the following characteristics:

- Non-peer reviewed (grey literature, books, . . .).
- Not written in English.
- Clearly obsolete results (more than 20 years old).
- Related to non-software companies (biotech, manufacturing, electronics, . . .).
- Related only to established companies (VSE, SME, research spin-offs).
- Related only to technicalities of startups (algorithms, programming languages, . . .).

We started analyzing the metadata of each article in pair (title, venue, keywords and the publication year) to exclude items that clearly didnt satisfied the inclusion criteria (221), leaving important decision for the subsequent steps. Then, going through a more in-depth review, we analyzed abstracts of each paper to determine whether the article satisfied our inclusion criteria, eliminating 658 items. In case of internal conflicts, hard decisions, or incomplete abstracts we read through an entire article, excluding 27 additional items. As shown in Figure 3,

while screening abstracts we improved the search process by enriching the initial search strings with new keywords identified in retrieved articles and iteratively conducting a new search (Figure 2).

During the screening of papers, we have found that some articles, not included, had the potential of contributing to our analysis. For this reasons we created an additional list (*Rejected-but-interesting List*) that we filled during the exclusion process with 71 managerial articles filtered and analyzed in a dedicated section in Appendix **??**.

During this review process we kept track of the rationale for each exclusion, visually presented in the right column of the Figure 4.

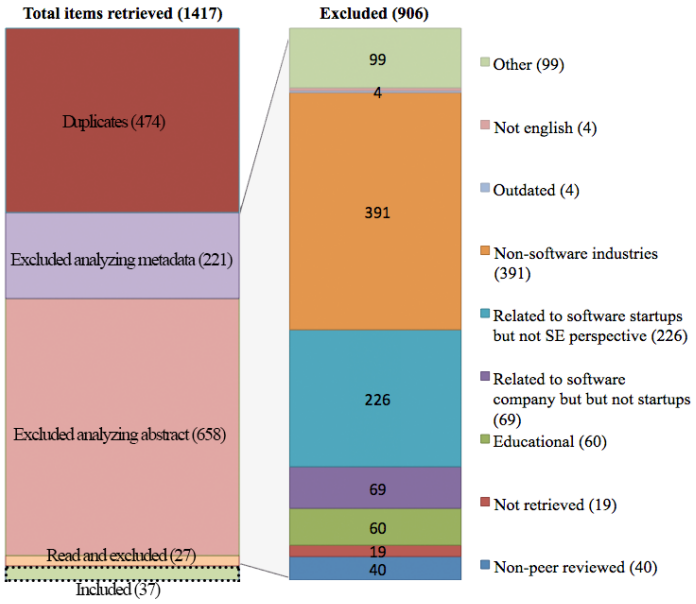

Figure 4: Studies selection process overview

The majority of paper we excluded were not related to the software industry (391), not interesting under a SE perspective (226) or related to established companies (69). Other excluded work were off-topic (99), educational (60), non peer-reviewed (40), non-English(4) or outdated (4).

3.4. Mapping

//Distinguish between this keywords and terms of the search string!! //Here specify how keywords have been extracted

To generate the *classification schema* we used the technique of *keywording*, following the systematic process illustrated in Figure 5.

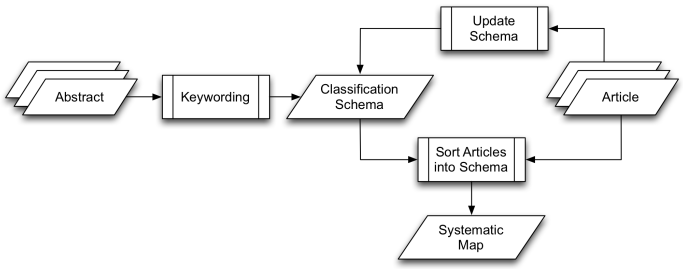

Figure 5: Classification schema creation adapted from [39]

The first step consisted in reading abstracts of the relevant studies, assigning them a set of keywords, to spot the main contribution area of a paper. To get a rough understanding of the research area represented by the sample of collected papers, we combined together the defined keywords forming a high-level set of categories, which initially generated a first representation of the selected articles. Then, by progressively trying to fit the articles into categories, the schema underwent a refinement process by being updated to receive new data.

When executing the actual extraction, we motivated with comments the non-obvious decisions we took in a spreadsheet, writing the rationale behind the belonging of an article to a certain category rather than another. The output of this process is shown in *Results and Analysis* (see Subsection **??**) and represents the final *systematic map*.

3.5. Data extraction

After we defined the *classification schema* (presented in Subsection **??**)we proceeded to systematically extract data from the selected relevant studies. We filled a spreadsheet where, for each article, we assigned a specific category of the *schema*, and additionally we collected other information inspired by other similar studies [51, 52]:

//Distinguish metadata and non-metadata //Talk about themes and how we extracted them

- Article title.
- First author.
- Year of publication.
- Synthesis of results (one-line).
- Keywords.

We took advantage of the data extraction process to identify an additional relevant aspect which emerged while reading abstracts and documents: the recurrent patterns of common attributes among startup companies (resulting *themes* are reported in Subsection **??**). Moreover, the bibliography of each extracted article has been screened to identify other possible relevant articles to our research, adopting the *snowballing* technique[7] [53].

//Data extraction we should add the content review we made by reading it

3.6. Rigor and relevance assessment

The primary challenge of SE is to transfer research results and knowledge to practitioners showing valid and concrete advantages [54]. To assess how results are presented in our studies selection, we integrated the traditional framework of SMS with an additional step, that is the evaluation of papers *rigor* and *relevance*.

We used a systematic and validated model developed by Ivarsson and Gorschek [54] to evaluate the scientific rigor and the industrial relevance of each paper. The model provides a complete set of rubrics to measure *rigor* and *relevance* dividing these two factors in different aspects, and quantifying the way in which each aspect is considered in the study (also see [55] for a real application).

As described by the authors, rigor refers to the precision or exactness of the research method used and how the study is presented. Aspects that are considered are:

- Context - description of development mode, speed, company maturity and any other important aspect where the evaluation is performed.

[7]Note that in our case with snowball sampling we didnt identify any additional relevant article, but in the case that relevant articles were retrieved, a refinement of keywords and reprocess of the *systematic mapping study* would have been required as described in Figure 2.

- Study design - description of the variable measured, treatments, control used and any other design aspect considered.
- Validity - description of different types of threats to validity evaluating conclusion, internal, external and construct validity.

On the other hand the relevance of an article consists of the realism of the environment where the study is performed and in the degree to which the research method undertaken has been investigated in several literature reviews. Aspects that are considered are:

- Subjects - use of subjects who are representative of the intended users of the technology.
- Context - use of settings representative of the intended usage setting.
- Scale - use of a realistic size of the applications.
- Research method - use of a research method that facilitates investigating real situations and relevant for practitioners.

Aspects related to the rigor of the study are scored with three score levels: *weak*, *medium* and *strong* description. Whilst, aspects related to relevance are scored 1 if contributing, 0 otherwise. For the sake of brevity we reported only the high-level quantification table (see Table 4). The detailed rubrics, used to evaluate the articles, can be found in [54].

Table 4: Rigor and relevance quantification

Rigor Ri = Ri1 + Ri2 + Ri3	
Context described (Ri1)	
Study design described (Ri2)	
Validity discussed (Ri3)	
Each aspect scored according to following scheme	
Weak presentation	0
Medium presentation	.5
Strong presentation	1
Relevance: Re = Re1 + Re2 + Re3 + Re4	
Context (Re1)	
Research method (Re2)	
User/Subject (Re3)	
Scale(Re4)	
Each aspect scored 1 if contributing to relevance, 0 otherwise	

Finally to obtain the final score of rigor and relevance the sum is performed according to the number of aspects that are classified as contributing to the industrial rigor and relevance respectively.

With this approach we intended to extend the scope of the traditional *Systematic Mapping Study*, which is limited to paper classification without assessing the quality of underlying studies. In our research, by combining a classic SMS with the study of rigor, relevance and the creation of a simple ranking function (see next subsection), we obtained a quick tool to partially overcome the above-mentioned limitations.

3.7. Ranking of articles

The final step, which concludes the mapping study, is the creation of a ranking function. We assigned a score to each paper, evaluating it in face of its *classification schema*s categories, *rigor* and *relevance* score, and two additional factors which characterize the venue of the study and the publication year.

We provided a rough estimation of the actual *value* that each paper brought to the research domain, by giving more importance to recent rigorous journal articles entirely devoted to the topic and presenting empirical results relevant to practitioners. To reflect those criteria, we used tables for converting each factor into an arbitrary numerical value in the range between 0 and 10. The conversion tables which were used to quantify the internal score of each dimension are discussed in Appendix **??**. The final ranking of the 37 relevant articles is presented in *Results and analysis* (see Subsection **??**).

//Maybe add tables with scores motivation?

3.8. Threats to validity

We identified potential threats to the validity of the systematic mapping and its results, which are presented in this section together with the mitigation strategies, structured following the example of a recent study [56].

3.8.1. Publication bias

Systematic reviews suffer from a common bias due to the general problem that *positive outcomes are more likely to be published than negative ones* [56]. In our study this threat is moderate, since the conclusions we draw and the *theoretical framework* we present are based on empirical data. The literature review is only used to make a comparison and validation of our results.

3.8.2. Threat to identification of primary studies

The approach we used to construct the *search string* (see Section **??**) aimed to collect the larger number of articles related to software development in startups as possible.

However, a limitation of the current search string lies in the non-inclusion of the stand-alone terms *"startup"* or *"start-up"*, to avoid the screening of more than 20 thousands papers, mostly irrelevant because related to the english phrasal verb "to start up", largely used in many disciplines to indicate the commencing moments of an engine. Then, to mitigate the risk of losing some relevant articles, we included in the search string many redundant terms in logical *or*, to increase the chances of catching any paper somehow related with software development in startup companies limiting the risk of obtaining an unreasonable number of articles with an extremely low *precision* (defined as the ratio of retrieved relevant and all retrieved [57]) .

Considering the precision rate of the current *search string* (3.92%), the value we obtained is very low anyway, with 37 studies selected from an initial sample of 943. However we were not interested in obtaining high precision as much as we wanted to obtain an high *recall*, that is the expressed by the ratio of retrieved relevant articles and the existing relevant items [57]. Despite the recall is based upon an unknown quantity hard to estimate (especially in an area where no systematic reviews have ever been conducted to the best of our knowledge), to have our evaluated and validated relevant list we submitted our systematic review results to prominent researchers in the area (identified with the mapping study itself) immediately after executing the methodology. The risk of leaving out relevant results is further mitigated by the use of multiple databases, which cover the majority of scientific publication in the field.

We were not able to retrieve 19 items since they were not available neither in online catalogs nor in the three libraries we consulted. However, this is a minor risk as we had access to their titles, keywords and venues, which gave us a good degree of confidence they were not relevant. Additionally, considering our 3.92% precision rate, the number of relevant articles in this small sample would have been statistically around *one*.

Another threat was given by the fact that relevant studies could have been published after the actual execution of the systematic review. To mitigate this risk we used the alerts featured on *Engineering Village* and *Google Scholar* database which allowed us to receive a weekly

newsletter with new items targeted by our *search string*. At the current date (29th October 2012) no new relevant studies emerged.

Furthermore we noticed a high inconsistency in the use of the word *startup* by different researchers, even in the same area. We couldnt identify a single and widely accepted definition of *startup*, but we identified multiple and somewhat conflicting definitions. With the support of thesaurus and librarians specialized in software engineering we considered their suggestions for additional improvements. Under these conditions, the attempt to identify a body of knowledge and restrict the scoping of our research was highly challenging. However, with the systematic study we were able to get a complete overview of how the term is used and sometimes misused, so we could focus the remains of our research on the early stage of startup companies, that is the most neglected stage by empirical primary studies so far.

Since startups and entrepreneurship in general are appealing for many sectors of the economy, an additional threat lies in the fact that some relevant information can be found in other academic sectors not considered in this systematic review.

3.8.3. Threats to study selection and data extraction

An important bias to consider - as discussed in [51] - is related to the selection of publications and the data extraction process, that in our case has been mitigated with an up-front definition of the inclusion/exclusion criteria [40]. The extent to which a SMS is able to provide an overview of software engineering topics has been sufficiently reported by the comparison analysis with the systematic literature review, discussed in [39].

The selection of relevant articles can be further biased by the personal opinions of researchers executing the process. To mitigate this threat and partially supply to our junior experience, we defined and documented a rigid protocol for the selection of each paper and we collaborated to mutually adjust each other biases by conducting the selection together and dedicating a reasonable amount of time to review conflicts, as suggested by [40]. Furthermore we documented the rationale behind the exclusions, to support us in the process of consensus creation by coherently consulting the history of previously taken decisions.

Another threat is related to researchers personal judgments, which can interfere with the evaluation of rigor and relevance of selected studies. However, the model we used to perform the assessment [54] provides rigorous guidelines and detailed rubric tables which have been observed to express our judgment more objectively.

An additional threat can undermine validity of the *ranking* of selected studies (see Section **??**) that we computed, as the scores for each category and the weights for each dimensions have been arbitrarily chosen to reflect our criteria. However, we consider this threat marginal as we didnt draw any important conclusions based on the ranking table and we specified that the score assigned to each paper has been used only to sort lists of relevant studies in the tables along the document. Furthermore we used an automatic spreadsheet to compute the final scores, which allowed us to adjust scores and weights, observing the effect of the final ranking in real time. For validating our ranking, we tried to modify scores/weights values several times, and we observed that the final ranking was not significantly altered by reasonable numerical adjustments, as long as we kept the order of importance between concepts.

Additionally, we used multi-methodological triangulation [58, 59] to confirm the validity of our results, by intersecting the data extracted with *systematic mapping study* to the empirical data which we obtained by *semi-structured interviews* and *follow-up questionnaires*.

4. Classification schema

The above identified keywords, together with existing taxonomies, contributed to the formation of a formal *classification schema*. The final schema, which emerged from the process specified in Subsection **??**, consists of four dimensions or *facets*:

- *Research type*: to represent the type of study undertaken.
- *Focus*: to describe the main focus of the research.
- *Contribution type*: to map the different types of outcome of the study.
- *Pertinence*: to distinguish between articles entirely devoted to engineering activities in startups and the ones which are only mentioning something related to it.

Each facet is formed by different categories described in the following four tables - one per dimension - which were used to map the selected studies and obtain a systematic map of the area.

The first dimension, *Research type* facet (see Table 5), can be used to distinguish between different types of studies abstracting from the specific underlying research methodology, unlike other types of classification [54]. The research types have been adapted from [60]. The second dimension describes the categories of the *focus facet*, which were obtained by clustering together sets of keywords previously identified (see Table 6).

We separated studies concerning software development practices from articles focused on the high-level process management. Furthermore we provide two additional categories to classify articles focused on specific *tools and technologies* and more *managerial/organizational* studies. The third dimension, *Contribution facet*, is presented in Table 8.

Similarly to the taxonomy used in [61], this facet expresses which kind of contribution a study brings to the fields. Contribution types can be divided in *weak* (*advices and implications*, *lesson learned*, *tools* and *guidelines*) and *strong* (*theory*, *framework/methods* and *model*). Finally the *pertinence facet* (see Table 7) helps in distinguish between studies fully pertaining to engineering activities in software startups (*full* pertinence), studies partially focused on some engineering activities in startups (*partial* pertinence), and finally studies only mentioning some relevant information pertained to software development in startups (*marginal* pertinence).

Research type Facet	
Evaluation Research	Methodology is implemented in practice and an evaluation of it is conducted. That means, it is shown how the research is implemented in practice (solution implementation) and what are the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation). This also includes to identify problems in industry.
Solution Proposal	A solution for a problem is proposed, the solution can be either novel or significant extension of an existing methodology. The potential benefits and the applicability of the solution is shown by a small example or a good line of argumentation.
Philosophical Papers	These papers sketch a new way of looking at existing things by structuring the field in form of a taxonomy or conceptual framework.
Opinion Papers	These papers express the personal opinion of somebody whether a certain technique is good or bad, or how things should been done. They do not rely on related work and research methodology.
Experience Papers	Experience papers explain on what and how something has been done in practice. It has to be the personal experience of the author.

Table 5: Classification schema - Research type facet

Focus facet	
Type	Description
Software development	Engineering activities used to develop software.
Process management	Engineering activities used to manage the development activities.
Tools and technology	Instruments used to create, debug, maintain and support development activities.
Managerial/organizational	Aspects that are related to software development, by means of: resource management and organizational structure.

Table 6: Classification schema - Focus facet

Contribution facet	
Type	Description
Model	Representation of an observed reality by concepts or related concepts after a conceptualisation process.
Theory	Construct of cause-effect relationships of determined results.
Framework/Methods	Models related to constructing software or managing development processes.
Guidelines	List of advices, synthesis of the obtained research results.
Lesson learned	Set of outcomes, directly analysed from the obtained research results.
Advice/Implications	Discursive and generic recommendation, deemed from personal opinions.
Tool	Technology, program or application used to create, debug, maintain or support development processes.

Table 7: Classification schema - Contribution Facet

Pertinence Facet	
Type	Description
Full	Entirely related (main focus) to engineering activities in software startups.
Partial	Partially related to engineering activities in software startups. Main research focus related to engineering activities.
Marginal	Marginally related to engineering activities in software startups. Main research focus different from engineering activities.

Table 8: Classification schema - Pertinence Facet

5. Results

This section presents the results of the *systematic mapping study*. From an initial sample of 943 articles, we selected 37 relevant studies pertaining engineering activities in startups.

The section is structured according to the RQ-1 and its sub-questions:

- Publication distribution (see Subsection **??**).
- Rigor and relevance (see Subsection **??**).
- Contextual features of startups (see Subsection **??**).
- State-of-the-art: summary (see Subsection **??**).

5.1. Publication distribution

Figure 6 shows the frequency distribution of the publication year, from 1994 to 2011.

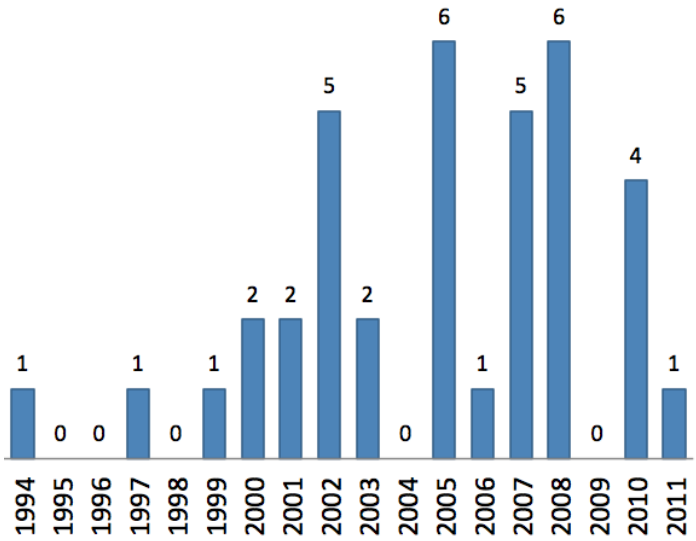

Figure 6: Publication distribution-year

To characterize what are the main topics covered within the area of engineering activities in software startups, we assigned to each article a set of keywords based on the content and the articles own keywords (when available in the metadata), during the process of *keywording* (see Subsection **??**). From the 37 relevant studies we extracted a total number of 327 keywords (117 uniques) averaging about 9 keywords per article. To provide a visual representation of the content of the studies, we show the frequency of occurrence of each keyword in a *tag-cloud* fashion[8] in Figure 7. The table containing the raw data, which was used to generate it, is shown in Appendix **??** (see Table .15).

[8]The tag cloud was generated with the online service (http://tagcrowd.com/).

Figure 7: Keywords cloud overview

The *tagcloud* (see Figure 7) was obtained by assigning keywords to each study, reflecting the most important topics covered by the authors. Then, counting the occurrence of each keyword, the reader can grasp an overview of the main concern of the studies in the sample.

Furthermore, by reading the articles and analysing their references, we observed that the selected studies are poorly interconnected to each other. They represents low intra-referencing among authors which makes it harder to ground findings to existing works.

Finally, by fitting the selected studies into the classification schema, we built the *systematic map*, shown in Table 9.

Author (year)	Pertinence	Focus	Research Type	Contribution Type	Ref.
Coleman (2008)	Full	Process Management	Evaluation Research	Model	[7]
Coleman (2007)	Full	Process Management	Evaluation Research	Theory	[33]
Coleman (2008)	Full	Process Management	Evaluation Research	Theory	[17]
Kajko (2008)	Full	Process Management	Evaluation Research	Model	[9]
Häsel (2010)	Marginal	Managerial & organizational	Evaluation Research	Model	[62]
Hanna (2010)	Marginal	Managerial & organizational	Evaluation Research	Model	[63]
Deakins (2005)	Partial	Managerial & organizational	Experience Paper	Model	[64]
Camel (1994)	Full	Software Development	Evaluation Research	Lesson Learned	[65]
Silva (2005)	Full	Software Development	Evaluation Research	Lesson Learned	[66]
Midler (2008)	Partial	Managerial & organizational	Evaluation Research	Framework & Methods	[67]
Taipale (2010)	Full	Software Development	Experience Paper	Advice & Implications	[68]
Chorev (2006)	Marginal	Managerial & organizational	Evaluation Research	Model	[23]
Zettel (2001)	Full	Software Development	Solution Proposal	Framework & Methods	[69]
Jansen (2008)	Partial	Software Development	Evaluation Research	Lesson Learned	[70]
Sutton (2000)	Full	Process Management	Opinion Paper	Advice & Implications	[2]
Heitlager (2007)	Full	Process Management	Solution Proposal	Tool	[19]
Tingling (2007)	Full	Software Development	Evaluation Research	Advice & Implications	[71]
Deias (2002)	Full	Software Development	Experience Paper	Advice & Implications	[72]
Stanfill (2007)	Marginal	Managerial & organizational	Solution Proposal	Advice & Implications	[73]
Wood (2005)	Partial	Software Development	Experience Paper	Advice & Implications	[74]
Steenhuis (2008)	Marginal	Managerial & organizational	Evaluation Research	Lesson Learned	[75]
Yogendra (2002)	Partial	Managerial & organizational	Evaluation Research	Guidelines	[76]
Ambler (2002)	Full	Software Development	Experience Paper	Lesson Learned	[77]
Crowne (2002)	Full	Software Development	Solution Proposal	Advice & Implications	[4]
Mater (2000)	Partial	Managerial & organizational	Evaluation Research	Model	[78]
Kakati (2003)	Marginal	Managerial & organizational	Evaluation Research	Model	[24]
Kuvinka (2011)	Partial	Software Development	Experience Paper	Advice & Implications	[79]
Su-Chuang (2007)	Marginal	Software Development	Evaluation Research	Advice & Implications	[80]
Sau-ling Lai (2010)	Marginal	Managerial & organizational	Evaluation Research	Lesson Learned	[81]
Mirel (2000)	Partial	Managerial & organizational	Solution Proposal	Advice & Implications	[82]
Himola (2003)	Marginal	Managerial & organizational	Solution Proposal	Advice & Implications	[83]
Kim (2005)	Marginal	Managerial & organizational	Evaluation Research	Model	[84]
Wall (2001)	Partial	Software Development	Experience Paper	Advice & Implications	[85]
Yoffie (1999)	Marginal	Managerial & organizational	Evaluation Research	Guidelines	[86]
Bean (2005)	Marginal	Tools and technology	Philosophical Paper	Advice & Implications	[87]
Tanabian (2005)	Marginal	Managerial & organizational	Opinion Paper	Advice & Implications	[22]
Fayad (1997)	Marginal	Process Management	Philosophical Paper	Advice & Implications	[88]

Table 9: Systematic map overview

The above table quickly reveals the nature of the study conducted, the extent of pertinence with the research problem, the type of contribution produced and the major research area investigated. Although the table is complete and contains the entire classification schema, it is still hard to extract relevant information from it. For this reason we computed the frequency of publications in each category using an electronic spreadsheet. In this way it is easier to emphasize what has been achieved in the past by researchers in the area, identifying gaps to drive the direction of our *grounded theory* case study, and suggesting possible future researches.

As suggested by Peterson et al. [39], to provide a good overview on how the topic is structured, we present our *systematic map* using multi-dimensional bubble charts (*"x-y scatter plots with bubbles in categories intersections"*) where the size of the bubble is determined by the number of publications corresponding to the *x*-*y* coordinates. Differently from other similar studies [89, 41], in our *classification schema* each data point is represented by four features. Thus, we created three plots to visualize all the six possible combinations of *facets*, giving a complete overview of the systematic map.

For example, the big bubble in the top-left part of Figure 8 indicates that 11 studies (29.73% of the total) are focused on *managerial and organizational factors* and conducted through an *evaluation research*. Simultaneously in the same figure is possible to observe, for instance, how 7 items with *managerial and organizational* focus contributed to the body of knowledge with a *model*. However, by looking at Figure 9, one can quickly notice that 5 out of the total 9 *models* have only a *marginal* pertinence with engineering activities in software startups.

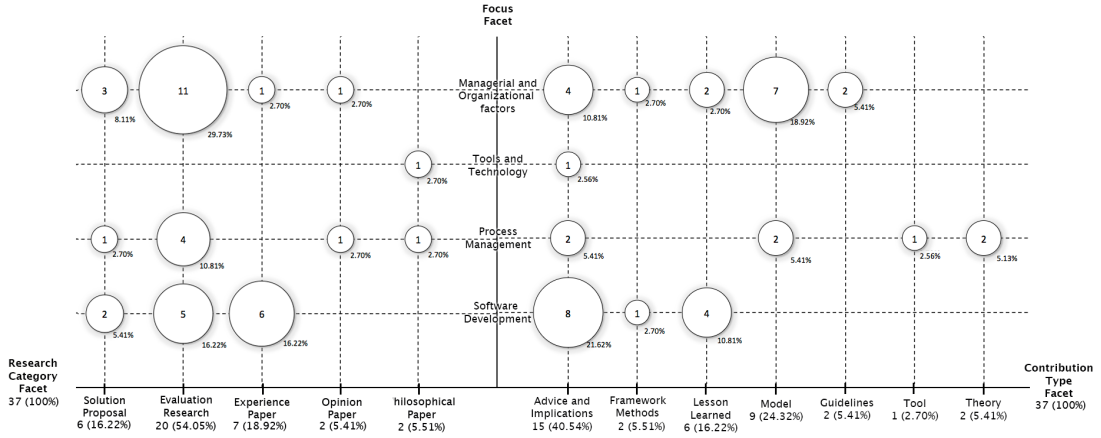

Figure 8: Systematic map - *Focus*, *contribution* and *research type*

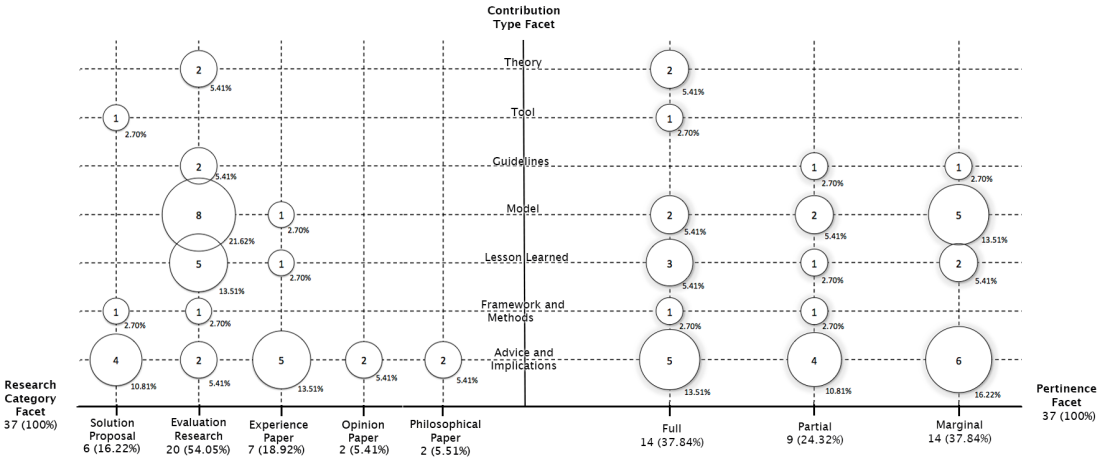

Figure 9: Systematic map - *Contribution*, *pertinence* and *research type*

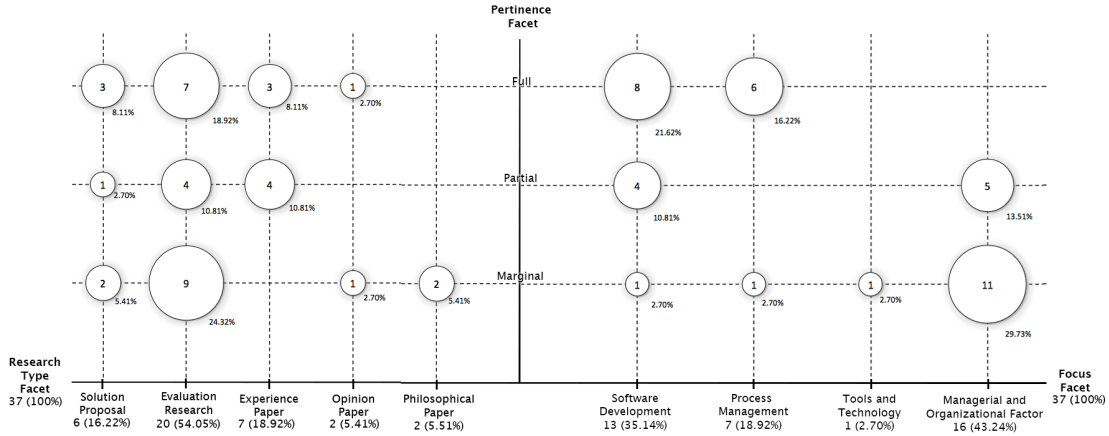

Figure 10: Systematic map - *Pertinence*, *focus* and *research type*

By analyzing in depth the three bubble charts representing the *systematic map* of the selected studies, we report a list of most significant observations:

- First of all, by looking at the *pertinence* facet we can observe that only 14 papers (37.84%) are entirely dedicated to software development in startups, and 9 of those produced a *weak contribution* (*advice and implications (5)*; *lesson learned (3)*; *tool* (1)).
- Observing the *focus* facet, it is easy to see how 16 articles (43.24%) are focused on *Managerial and organizational factors*, which are only relatively interesting under a SE perspective. In fact, none of those 16 articles have a *full pertinence*.
- The overall *contribution types* produced by the studies are for the greater part *weak* (24 items, 64.86%). Of the 13 remaining studies (35.13%), which produces a *strong contribution type*, only 4 have the *focus* on what is considered fundamental for our research problem (*software development* and *process management*).
- A good portion of the selected studies were carried out using an *evaluation research* (20 items, 54.05%) which is the only *research type* which involves a field study. However, by simultaneously looking at other facets, we can observe how 11 of these evaluation researches are related to *managerial and organizational factors* , and only 7 out of 20 have a *full pertinence* with engineering activities in software startups. On the other hand the *contribution types*, produced by *evaluation researches*, are widely distributed on the map, whilst other - less empirical - *research types* generally brought only to *weak contribution types*.
- 14 out of 20 studies, with *focus* on *process management* (7) and *software development* (13), have a *full pertinence* with our research area.

So, if we don't consider items which produced a *poor contribution type*, items which do not have *full pertinence* with software development in startups and items with a non-empirical research approach, only four studies remain ([7, 17, 33, 9]), and represent the most prominent contribution to the field.

5.2. Contextual features of startups

In this part we present the results of the data extraction process described in Subsection **??**, where we identified peculiar characteristics of startup companies as described in the literature. The results confirms that there is no agreement on a standard definition which specify what exactly a startup is. Different authors provide different definitions and they use the term *startup* referring to context which are often quite different. This makes any attempt to identify a solid and coherent body of knowledge very challenging.

Defining what makes a software startup unique is an interesting problem: apparently is not strictly related to the size of the company. For instance, some authors call *startups*, companies with 6 employees [82], whilst others refer to *startups* with more than 300 employees [86, 7]. It is not about the age of the company either : some authors studied *startups* which have been operating for many years [81], while others are more strict about companies recently founded [9]. For other authors *start-up* is a phase, and it is not clear where to draw the line anyway. Others claim that startups works on *innovative* products, but they dont give the exact definition of innovation, which makes this factor a little confusing (a recent systematic study identified *"41 definitions of innovation in 204 selected primary SE studies"* [90]).

To illustrate how authors use the term *software startup*, we systematically extracted *themes* which characterize companies inquired in the selected relevant studies. We were able to identify 15 main *themes* which are reported in Table 10. The last two columns on the right side indicate respectively the frequency of occurrence of a specific theme and its references.

ID	Theme	Description	Frequency	Ref.
T.1	Lack of resources	Economical, human, and physical resources are extremely limited.	13	[76], [86], [4], [9], [65], [17], [33], [7], [63], [2], [77], [73], [22]
T.2	New company	The company has been recently created.	4	[4], [86], [65], [23]
T.3	Small Team	Startups starts with a small numbers of individuals.	7	[69], [86], [4], [9], [2], [23], [22]
T.4	Uncertainty	Startups deal with a highly uncertain ecosystem under different perspectives: market,product features, competition, people and finance	8	[19], [84], [17], [33], [7], [88], [67], [22], [83]
T.5	Little working history	The basis of an organizational culture are not present initially.	2	[86], [77]
T.6	Higly Risky	Failure rate of startups is extremely high.	4	[19], [9], [65], [22]
T.7	Not self-sustained	Especially in the early stage startups need external funding to sustain their activities (Venture Capitalisst, Angel Investments, Personal Funds, ...)	2	[69], [63]
T.8	Low-experienced team	A good part of the development team is formed by people with less than 5 years of experience and often recently graduated students.	5	[65], [17], [33], [77], [24]
T.9	Flat organization	Startups are usually founders-centric and everyone in the company has big responsabilities. No high-management.	4	[86], [9], [66], [22]
T.10	One product	Company's activities gravitate around one product/service only.	7	[77], [79], [87], [72], [66], [7], [68]
T.11	Innovation	Given the highly competitive ecosystem, startups need to focus on highly innovative segments of the market.	11	[19], [76], [82], [75], [70], [81], [2], [62], [62], [67], [24]
T.12	Time-pressure	The environment often forces startups to release fast and to work under constant pressure (terms sheets, demo days, investors' requests)	8	[69], [65], [7], [71], [2], [64], [83], [78]
T.13	Highly Reactive	Startups are able to quickly react to changes of the underlying market, technologies, and product. (compared to more established companies)	11	[69], [9], [65], [17], [7], [71], [2], [88], [77], [79], [72], [66]
T.14	Third party dependency	Due to lack of resources,to build their product startups heavily rely on external solutions: External APIs,Open Source Software, outsourcing, COTS, ...	7	[86], [74], [85], [70], [81], [63], [2], [77]
T.15	Rapidly Evolving	Successful startups aim to grow and scale rapidly.	8	[76], [86], [65], [7], [80], [2], [77], [79], [64]

Table 10: Mapping Study - Recurrent themes

When discussing software startups, thirteen authors reported a general lack of human, physical and economical resources (T.1). For this reason, startups deeply depend upon external software solutions such as third party APIs, COTS and OSS (T.14). Other studies refer to companies which are able to quickly react to changes in the market and technologies (T.13), under remarkably uncertain conditions (T.4). Some authors indicate that these companies are focused on highly innovative segments of the market (T.11), generally working on a single core-product (T.10) under extremely high time-pressure (T.12). Furthermore, eight authors write about startups as fast growing companies (T.15) designed to rapidly scale-up. Other researches mention a very small founding team (T.3), which is often composed by low-experienced people (T.8) with a very flat organization structure (T.9) where the CEO is sometimes a core developer itself. Finally, other studies agree on the highly risky nature of these businesses (T.13) newly created (T.2) and therefore with no or little working history (T.5).

It is important to understand that the above mentioned contextual factors have a relevant impact on software development activities[9], making them different from established companies [2]. And since there is no common agreement on the use of the term *startup* and its implications, it should be an important responsibility of the authors to specify which kind of companies the study actually refers to, in order to avoid misleading and ambiguous results. In the selected studies, most of the authors used the term *startup* without explicitly mentioning what exactly they meant.

For this reason in Chapter 2 we specified that our case study and consequently the results of the grounded theory regard to *newly created and product-centered companies, in the time-frame that goes from the idea conception to the release of the first product in highly scalable markets* (this formulation was initially shaped around the definitions used in [91, 92, 13]). We finally dedicated a subsection to discuss the generalizability of results to similar domains (see Section **??**).

5.3. Rigor and relevance

Figure 11 shows a *pie chart* representing the distribution of the relevant sample in the three *venue* categories.

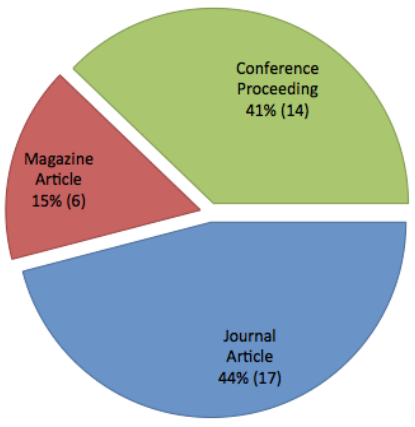

Figure 11: Publication distribution - Venue

Despite the scientific validity of a study cannot be a mere consequence of the venue where it has been published, the peer-review process, required for publishing a journal article, is generally much more rigorous and formal than the procedure to get an article published on a scientific magazine or accepted to a conference [93]. As can be seen in Figure 11, only 17 publications (44%) in the sample are journal articles, while the remaining 56% consist of conference proceedings and magazine articles. Although this feature alone is not enough to represent a direct implication on the quality[10], it can be interpreted as a first indicator of the scientific quality of the sample, formally assessed with the rigor and relevance process here discussed.

Following the process described in Subsection **??**, we assessed the *rigor* and *relevance* of each paper by summing up the contribution of individual *aspects* defined in the evaluation model [54]. The detailed results of this process are shown in Table 11. Observe that the maximum possible value for rigor ($Ri = Ri1 + Ri2 + Ri3$) is 3, and for relevance ($Re = Re1 + Re2 + Re3$) is 4 (see the detailed quantification Table 4).

[9] To further attest the validity our theoretical model, in Section **??** we use the above mentioned *themes* to execute a comparative analysis with the theoretical framework we developed with the *grounded theory* case study.

[10] The publication criteria are determined by the specific editor of the journal/magazine or the committee of a conference. There is a vast multitude of excellent quality studies presented in conference proceedings and magazines, and many example of poor-quality journal articles.

First author (year)	Ri1	Ri2	Ri3	**Ri**	Re1	Re2	Re3	Re4	**Re**	Ref.
Coleman (2007)	1.0	1.0	1.0	**3.0**	1.0	1.0	1.0	1.0	**4.0**	[33]
Coleman (2008)	1.0	1.0	1.0	**3.0**	1.0	1.0	1.0	1.0	**4.0**	[17]
Coleman (2008)	1.0	1.0	1.0	**3.0**	1.0	1.0	1.0	1.0	**4.0**	[7]
Camel (1994)	0.5	1.0	1.0	**2.5**	1.0	1.0	1.0	1.0	**4.0**	[65]
Häsel (2010)	0.0	1.0	1.0	**2.0**	1.0	1.0	1.0	1.0	**4.0**	[62]
Hanna (2010)	0.0	1.0	1.0	**2.0**	1.0	1.0	1.0	1.0	**4.0**	[63]
Deakins(2005)	1.0	0.5	1.0	**2.5**	1.0	1.0	0.0	1.0	**3.0**	[64]
Chorev (2006)	1.0	1.0	0.0	**2.0**	1.0	1.0	0.0	1.0	**3.0**	[23]
Kajko (2008)	1.0	0.5	0.0	**1.5**	1.0	1.0	0.0	1.0	**3.0**	[9]
Jansen (2008)	0.5	0.0	0.5	**1.0**	1.0	1.0	0.0	1.0	**3.0**	[70]
Midler (2008)	0.5	0.5	0.0	**1.0**	1.0	1.0	0.0	1.0	**3.0**	[67]
Steenhuis (2008)	0.0	1.0	0.0	**1.0**	1.0	1.0	0.0	1.0	**3.0**	[75]
Yogendra (2002)	1.0	0.0	0.0	**1.0**	1.0	1.0	0.0	1.0	**3.0**	[76]
Wood (2005)	0.0	0.5	0.0	**0.5**	1.0	1.0	0.0	1.0	**3.0**	[74]
Tingling (2007)	0.0	0.5	0.0	**0.5**	1.0	1.0	0.0	1.0	**3.0**	[71]
Su-Chuang (2007)	0.0	0.0	0.0	**0.0**	1.0	1.0	0.0	1.0	**3.0**	[80]
Sutton (2000)	1.0	0.0	0.0	**1.0**	1.0	1.0	0.0	0.0	**2.0**	[2]
Kakati (2003)	0.0	1.0	0.0	**1.0**	0.0	1.0	1.0	0.0	**2.0**	[24]
Mater (2000)	0.5	0.0	0.0	**0.5**	1.0	1.0	0.0	0.0	**2.0**	[78]
Yoffie (1999)	0.5	0.0	0.0	**0.5**	1.0	1.0	0.0	0.0	**2.0**	[86]
Deias (2002)	0.5	0.0	0.0	**0.5**	1.0	1.0	0.0	0.0	**2.0**	[72]
Silva (2005)	0.0	0.0	0.0	**0.0**	1.0	1.0	0.0	0.0	**2.0**	[66]
Wall (2001)	0.0	0.0	0.0	**0.0**	1.0	1.0	0.0	0.0	**2.0**	[85]
Kuvinka (2011)	0.0	0.0	0.0	**0.0**	1.0	1.0	0.0	0.0	**2.0**	[79]
Mirel (2000)	0.0	0.0	0.0	**0.0**	1.0	1.0	0.0	0.0	**2.0**	[82]
Zettel (2001)	0.5	0.0	0.5	**1.0**	1.0	0.0	0.0	0.0	**1.0**	[69]
Ambler (2002)	0.0	0.0	0.0	**0.0**	1.0	1.0	0.0	0.0	**2.0**	[77]
Stanfill (2007)	0.5	0.5	0.0	**1.0**	0.0	1.0	0.0	0.0	**1.0**	[73]
Taipale (2010)	0.0	0.0	0.0	**0.0**	1.0	1.0	0.0	0.0	**2.0**	[68]
Sau-ling Lai (2010)	0.0	0.0	0.0	**0.0**	1.0	1.0	0.0	0.0	**2.0**	[81]
Kim (2005)	0.5	0.0	1.0	**1.5**	0.0	0.0	0.0	0.0	**0.0**	[84]
Crowne (2002)	1.0	0.0	0.0	**1.0**	0.0	0.0	0.0	0.0	**0.0**	[4]
Heitlager (2007)	1.0	0.0	0.0	**1.0**	0.0	0.0	0.0	0.0	**0.0**	[19]
Himola (2003)	0.5	0.0	0.0	**0.5**	0.0	0.0	0.0	0.0	**0.0**	[83]
Bean (2005)	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0	0.0	**0.0**	[87]
Fayad (1997)	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0	0.0	**0.0**	[88]
Tanabian (2005)	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0	0.0	**0.0**	[22]

Table 11: Mapping Study - Rigor-relevance results

The table is sorted by placing at the top the studies, which received the highest scores in the two dimensions summed together. It is straightforward to observe that Colemans studies received the maximum score for both scientific rigor and industrial relevance. By contrast, there are many papers which report a 0 score (11 zeros for rigor, 7 zeros for relevance), providing a first confirmation of what was only a suspect about the generally low reliability of a good part of these studies.

To further extract useful insight, it is more convenient to adopt a graphical representation of the rigor and relevance. We use the same approach previously utilized to depict the *systematic map* : a bubble chart where the bubble size is proportional to the number of publications, corresponding to the coordinates *x*-*y*, along with some statistical data shown for each dimension (Figure 12).

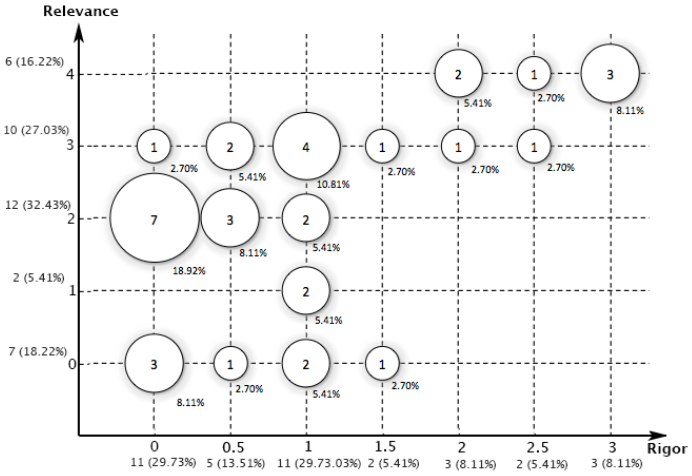

Figure 12: Rigor-relevance overview

If we dont consider studies in the highest region of scores (see the top-right part of Figure 12) for industrial *rigor* ($\geq$ 2) and *relevance* ($\geq$ 3) from the rest, it is easy to observe how the greater part of the selected studies (78.38%) received mediocre scientific rigor and industrial relevance scores. Only 8 items belong to the *high-score region*.

6. Analysis of the state-of-the-art

This section discus xyz and is organized as following

6.1. Topics treated in the literature

Beside the obvious high-frequency of the keyword *software-startup*, a pronounced interest for topics such as *management* (19 times), *software-process* (17), *agile-methodologies* (10) and *web-development* (7) is visible. For software-process, it is worth noticing how studies are concentrated around *process-improvement* (4) and *process-formation* (2). Among the different agile methodologies, a particular consideration is given to *extreme programming* (7), while other methodologies such as *RUP* (2) and *Scrum* (1) received little attention. This represents a first evidence of our initial assumptions that, in the general meaning of the word, startups are *agile* companies.

Among others, we can observe some topics that are quite peculiar for this area, such as *time-to-market* (6), *rapid-development* (3) and *founder* background (14).

The particular attention given by the literature to those topics, especially aspects related to the *team*, *agile-methodologies*, *time-to-market*, and *web-development*, helped us in refining the specific research domain, contributing to adjust the trajectory of the *case study* conducted with startups' practitioners (see Section **??**) and driving the formation of a *theoretical model*, which is presented and explained in Section **??**. However, the wide variety of topics, combined with the very-low frequency in which each topic appears, represents a symptom of a general scarcity of primary studies investigating specific issues.

6.2. Systematic map and distribution

More than 80% of the studies in the selected sample have been published in the last ten years, which compared to the long-standing history of the SE discipline and more in general software companies (half of a century), is a quite short time-frame. Only 7 relevant articles dated prior to 2002 discuss software startup related issues[11]. To make a comparison, when looking at one of the closest domain such as software engineering in small companies, we can observe how the specific literature started many years before 2002. For example, in 1994 Brodman was already discussing how to adapt the CMM methodology to small organizations [94]. Only 10 years later we find an empirical study inquiring startups and development methodologies [66].

The yearly distribution of publications attest the novelty of the startup phenomenon, which has been basically enabled and amplified by the potentially huge markets and distribution channels offered by the internet and mobile devices [15, 24]. This context posed a set of new problems and challenges which can hardly be faced using traditional approaches [2].

Summarizing the implications regarding the RQ-1.1 (*How is the body of knowledge distributed in literature?*) we have found that:

- The literature lacks of relevant primary studies which investigate engineering activities in software startups. The 37 selected studies are researches spread across different scientific areas and journals. In addition, they are weakly interrelated by mutual references. This confirms what was partially revealed by the first non-systematic literature survey and, at the same time, discloses a complex underlying field.
- About 80% of the studies have been published in the last decade, opening a set of new issues brought by the advent of internet and mobile devices which enabled fast and wide proliferation of software startups. The state-of-the-art under a SE perspective is quite recent, especially when compared to siblings areas such as SE in small companies, which is much more advanced in terms of number of primary studies and provided evidences.
- The selected studies focus on a wide variety of topics of which we provided a complete overview. Despite more attention is given to some sort of lightweight methodologies for startups and some discussion about process formation and improvement, the evidence produced are definitely undersized to support the dimension of the actual startup phenomenon.
- To asses the detailed distribution of the body of knowledge we mapped the identified studies into a classification schema which considers four dimensions, namely: research type, contribution type, main focus and pertinence. Of a total of 37 selected studies, 16 of them (43.24%) are focused on *managerial and organizational factors*, which are only partially interesting under a SE perspective. Additionally, only 14 studies (37.84%) are entirely dedicated to software development in startups and 9 of those produced weak contribution types.

[11] Other 4 articles were discarded in the selection process because they didnt comply with the selection criteria and then considered as obsolete (see screening process in Subsection **??**).

- Overall, we identified only 4 important contributions to the field that are published in scientific journals, are entirely dedicated to engineering activities in startups, provide strong contribution type and are conducted through an evidence-based research approach [7, 17, 33, 9]. Three of these studies are from the same author, Gerry Coleman.

6.3. Rigor and Relevance

Moreover, the biggest bubble in the chart shows 7 studies (almost 19% of the total) which received an average score for industrial relevance (2) but only a 0 for the scientific rigor. This kind of studies, according to the authors of the rigor-relevance model [54], presents a major issue: although their findings appear to be somewhat appealing for practitioners (average relevance), the extremely poor scientific rigor of the study will make a possible knowledge transfer to the industry highly unlikely or highly dangerous. Indeed Kitchenham states that one of the most important factors for having academic results applied in the industry is by providing proper scientific evidences of the implications [95, 96].

Finally, by looking at how bubbles are distributed on the chart (see Figure 12) with no need of sophisticated statistical methods, it is clear the relation between items which received the highest scores for *rigor* and the ones which received the highest score for *relevance*. Since the two dimensions represent two independent variables the result is not straightforward. In this particular sample, it indicates that results of studies, which have been rigorously reported, are generally more relevant for the industry (and vice-versa). By thoroughly analyzing the studies in the high rigor-relevance region, we observed that the authors have strong academic backgrounds but at the same time they have worked for many years in direct contact with startups. This is even more visible when comparing them to the other authors in the sample with lower value of rigor and relevance[12]. This could somehow explain the reason why, in our sample, the rigorous studies are also highly relevant.

Summarizing the implications regarding the RQ-1.2 (*What is the industrial relevance and scientific rigor of the published studies?*) we have found that:

- Only 17 publications (44%) in the sample are journal articles. The remaining 56% consists of conference proceedings and magazine articles, which usually require a less rigorous peer-review process than a scientific journal publication.
- By assessing the rigor and relevance of the selected studies with a systematic procedure we attested that the majority of works are poorly relevant for startups and are presented with low scientific rigor. By looking at the detailed rigor-relevance chart, it can be observed how most of the studies are located in the lower region of the rigor-relevance map. Additionally, eleven articles received 0 for scientific rigor, and seven articles received 0 for rigor.
- Colemans studies [7, 17, 33], which appears to be a prominent researcher in this area, received also the highest score both for scientific rigor and industrial relevance. This provided to our research a solid starting point. In fact, these references are used a considerable amount of times throughout the thesis document.
- Seven studies, which received an average score for industrial relevance, were marked with 0 for the scientific rigor. Thus, although their results could be potentially relevant (2) for the industry, the low rigor makes a possible adoption in startups highly-risky or highly-unlikely. These studies represent almost 19% of the total.
- The most rigorous studies in our sample are highly relevant for practitioners, and vice-versa. We observed that in those cases, the authors are academics with a deep connections with the industry or direct experiences in startups.

6.4. Defining startups

Summarizing the implications regarding the RQ-1.3 (*What are the features which characterize the context of software development in startups, reported in literature?*) we have found that:

- Different authors use the word *startup* actually referring to different kind of companies. It is very hard to identify an explicit definition of the context in which the companies operate, often given for granted. Under these conditions, trying to identify a coherent body of knowledge is even more challenging and is even harder for practitioners to somewhat adopt the results without having a proper context. For instance, some results about *"startups"* have been obtained by studying companies with more than 200 employees and others from companies with 6 employees. To make an example, from a SE engineering point of view, the size of the company actually have several implications on the software development activities [97].
- To capture the implicit contextual features that authors used when referring to software startup companies we identified a set of 15 main *themes* and counted their frequency in the selected studies. The most frequent reported themes concern the general lack of resources, highly reactiveness and flexibility, intense time-pressure, uncertain conditions and fast growth.
- Since the contextual boundaries of startups resulted to be highly blurred, it is responsibility of the researchers who refer to *"startups"* to explicitly mention which are the features of the company that the study is actually concerned with (in most of selected studies an explicit contextualization has been neglected). For this reason we explicitly declared the area of interest of our case study in Introduction (see Chapter 1).

6.5. Summary of findings

In this subsection we provide the overall answer to the RQ-1 : *What is the current state-of-the-art in the SE literature pertaining to engineering activities in startups?*.

To emphasize the importance of articles, which brought prominent contributions to the area under investigation, we followed the procedure described in Subsection **??**, computing a score (in the range $[0 - 10]$) for each selected study. Table 12 lists the dimensions which were used to assign the final score, next to the arbitrary weight to balance the ranking criteria.

Table 12: Ranking weights

W	Dimension (id)	Weight
w_1	Pertinence (P)	.25
w_2	Rigor (Ri)	.175
w_3	Relevance (Re)	.175
w_4	Age (A)	.15
w_5	Venue (V)	.1
w_6	Contribution (C)	.05
w_7	Research type (R)	.05
w_8	Focus (F)	.05
	TOTAL:	**1**

[12] Since it is not possible to obtain a complete background study of each author, we based our assumption on a small biographic review.

The most important dimension, counting for the 25% alone, is *Pertinence*, followed by *rigor* and *relevance*, respectively counting 17.5% each. The remaining dimensions are increasingly less important to the final score, which is computed for each paper i by adding up the contributions as shown in the formula below.

$$Score_i = (w_1 \cdot P_i) + (w_2 \cdot Ri_i) + (w_3 \cdot Re_i) + (w_4 \cdot A_i) + (w_5 \cdot V_i) + (w_6 \cdot C_i) + (w_7 \cdot R_i) + (w_8 \cdot F_i)$$

The final score is presented in Table 13. The dimensions, which contributed to the final score, are the columns representing: (A)ge, (Ri)gor, (Re)levance, (V)enue, (P)ertinence, (C)ontribution type, (R)esearch type and (F)ocus.

First author (year)	**Score**	A	Ri	Re	V	P	C	R	F	Ref.
Coleman (2008)	**9.70**	1.20	1.75	1.75	1.00	2.50	0.50	0.50	0.50	[7]
Coleman (2007)	**9.70**	1.20	1.75	1.75	1.00	2.50	0.50	0.50	0.50	[33]
Coleman (2008)	**9.70**	1.20	1.75	1.75	1.00	2.50	0.50	0.50	0.50	[17]
Kajko (2008)	**8.09**	1.20	0.88	1.31	0.70	2.50	0.50	0.50	0.50	[9]
Häsel (2010)	**7.47**	1.50	1.17	1.75	1.00	0.75	0.50	0.50	0.30	[62]
Hanna (2010)	**7.47**	1.50	1.17	1.75	1.00	0.75	0.50	0.50	0.30	[63]
Deakins(2005)	**6.87**	0.90	1.46	1.31	1.00	1.25	0.50	0.15	0.30	[64]
Camel (1994)	**6.61**	0.15	1.46	1.75	0.70	1.25	0.30	0.50	0.50	[65]
Silva (2005)	**6.58**	0.90	0.00	0.88	1.00	2.50	0.30	0.50	0.50	[66]
Midler (2008)	**6.55**	1.20	0.58	1.31	1.00	1.25	0.40	0.50	0.30	[67]
Taipale (2010)	**6.53**	1.50	0.00	0.88	0.70	2.50	0.30	0.15	0.50	[68]
Chorev (2006)	**6.43**	0.90	1.17	1.31	1.00	0.75	0.50	0.50	0.30	[23]
Zettel (2001)	**6.32**	0.60	0.58	0.44	1.00	2.50	0.40	0.30	0.50	[69]
Jansen (2008)	**6.25**	1.20	0.58	1.31	0.60	1.25	0.30	0.50	0.50	[70]
Sutton (2000)	**6.11**	0.60	0.58	0.88	0.60	2.50	0.30	0.15	0.50	[2]
Heitlager (2007)	**6.08**	1.20	0.58	0.00	0.70	2.50	0.30	0.30	0.50	[19]
Tingling (2007)	**5.99**	1.20	0.29	0.00	0.70	2.50	0.30	0.50	0.50	[71]
Deias (2002)	**5.92**	0.60	0.29	0.88	0.70	2.50	0.30	0.15	0.50	[72]
Stanfill (2007)	**5.74**	1.20	0.88	1.31	0.70	0.75	0.30	0.30	0.30	[73]
Wood (2005)	**5.70**	0.90	0.29	1.31	1.00	1.25	0.30	0.15	0.50	[74]
Steenhuis (2008)	**5.65**	1.20	0.58	1.31	0.70	0.75	0.30	0.50	0.30	[75]
Yogendra (2002)	**5.55**	0.60	0.58	1.31	0.70	1.25	0.30	0.50	0.30	[76]
Ambler (2002)	**5.53**	0.60	0.00	0.88	0.60	2.50	0.30	0.15	0.50	[77]
Crowne (2002)	**5.48**	0.60	0.58	0.00	0.70	2.50	0.30	0.30	0.50	[4]
Mater (2000)	**5.45**	0.60	0.29	1.31	0.70	1.25	0.50	0.50	0.30	[78]
Kakati (2003)	**5.41**	0.90	0.58	0.88	1.00	0.75	0.50	0.50	0.30	[24]
Kuvinka (2011)	**5.28**	1.50	0.00	0.88	0.70	1.25	0.30	0.15	0.50	[79]
Su-Chuang (2007)	**5.26**	1.20	0.00	1.31	0.70	0.75	0.30	0.50	0.50	[80]
Sau-ling Lai (2010)	**5.23**	1.50	0.00	0.88	1.00	0.75	0.30	0.50	0.30	[81]
Mirel (2000)	**4.98**	0.60	0.35	0.88	1.00	1.25	0.30	0.30	0.30	[82]
Himola (2003)	**4.57**	0.90	0.58	0.44	1.00	0.75	0.30	0.30	0.30	[83]
Kim (2005)	**4.53**	0.90	0.88	0.00	0.70	0.75	0.50	0.50	0.30	[84]
Wall (2001)	**4.28**	0.60	0.00	0.88	0.60	1.25	0.30	0.15	0.50	[85]
Yoffie (1999)	**4.22**	0.60	0.29	0.88	0.60	0.75	0.30	0.50	0.30	[86]
Bean (2005)	**3.50**	0.90	0.00	0.00	1.00	0.75	0.30	0.15	0.40	[87]
Tanabian (2005)	**3.10**	0.90	0.00	0.00	0.70	0.75	0.30	0.15	0.30	[22]
Fayad (1997)	**2.45**	0.15	0.00	0.00	0.60	0.75	0.30	0.15	0.50	[88]

Table 13: Mapping Study - Ranking of selected studies

It can be observed that the same 4 papers which occupy the first 4 positions of the above presented *ranking* ([7, 17, 33, 9]), are in the most interesting regions of the *systematic maps* described before, and at the same time are rated as highly rigorous and relevant (Table 13). These studies are, in fact, the ones which mostly guided and supported us in the research process. In particular Coleman's studies received the highest scores mainly because his works are actually focused on software process in startups from a SE perspective, undertaking rigorous and well documented approaches, and producing meaningful outcomes published on relevant journals. However, we observed that the three publications are, in fact, derived from the same study with 21 companies.

Although a little part of the companies inquired in his publications had an extremely limited number of employees involved in software development, the greatest part of them had more than 20 developers/engineers, making its results hard to generalize to actual early-stage startups with 3 or 4 founding members.

The results of the systematic mapping of the literature, fostered our motivation in filling the gap by executing a case study on early-stage startups. Thus, our case study is clearly distinct from what has already been investigated by Coleman. In fact, the companies, which we inquired in the interviews at the time of the first release, employed an average number of about 8 employees, contrary to Colman's companies which employed up to 190 employees.

Moreover, as Coleman himself admits, his results suffer from a limitation which is not present in our study due to the fact that his respondents were covering high-managerial roles (Managers/CTOs/Head of development) which are in practice *"one or more steps removed"* from actual carrying out of the software development [33]. Therefore their opinions are strongly biased towards methodologies they are themselves proposing to their teams. By contrast, in the early stage of the studied startups, the totality of our respondents were directly involved within all the activities of software development, giving to us the possibility to actually attest more reliable data[13].

Finally, we summarize what are the results of the overall systematic mapping, conducted through: creation of a classification schema which was used to map the existing literature; a systematic assessment of the rigor and relevance of the selected studies; and finally identifying contextual features which characterize software startups. The overall results of the analysis are summarized as follows (answering RQ-1):

- The evidences provided by the 37 selected studies are, for the most part, inadequate to understand the underlying phenomenon of software development in startups. To the current date, twelve years after Sutton assessed that startups have been neglected from process studies [2], the gap has been only partially filled.
- From the provided *systematic map*, it can be observed that only 13 studies out of 37 selected, are entirely dedicated to the study of software development in startups. The remaining articles are only partially or marginally mentioning relevant contributions to the area. Additionally, by visualizing the four dimensions of the systematic map with the technique proposed by Petersen et al. [39], we were able to provide detailed insights by simultaneously analyzing multiple facets of the literature distribution.
- The creation of a coherent body of knowledge about software startups is restrained by the fact that different authors use the word *startup* in different contexts (above summarized by showing the more cited contextual *themes*). Some authors use the word *startup* for small new companies while others considered startups companies with hundreds of employees. Others refer to innovative companies or operating in uncertain markets, and for others, *startup* is the name of a phase in a software project. Given this lack of consensus and consistency, when investigating software startups, it is responsibility of the researcher to make explicit mention to the particular context affected by the study. In the selected studies an explicit mention was often neglected, affecting the generalizability of results.
- The results of selected studies are mediocrely relevant for the industry. Furthermore, considering the general lack of high scientific rigor, it will be very unlikely that the findings can have a real impact on startup companies. Since transferring the knowledge to the companies should be one of the most important concern of SE research [54], the next generation of studies should provide more relevant and scientifically rigorous evidences. However, in our sample, we identified a small portion of studies with both very high relevance and very high rigor. We observed that the authors of those papers have generally academic background with strong connections or past experiences with startup companies. This might infer that the combination of excellent academics skills (providing rigor) with practical experience in the field (providing relevance) fosters the quality of a startup-related study.
- We identified only four prominent contributions to the field by ranking the studies according to a defined score, considering different features of the study. Furthermore we realized that three of these studies, conducted from the same author, are based on the same empirical data collected in 21 companies with different profiles.
- Some of the features, which characterize startups, are common to other SE domains (innovation, market-driven development, small companies, short time-to-market, ...). However, the unique combination of elements, which coexists in modern startup companies, poses a new series of issues which need to be addressed with primary studies in the specific domain. A new and consistent body of knowledge should support activities and decisions of the fast growing number of startup companies, through evidence-based research [95, 98]. The need of more studies is attested by the impressing proliferation of (non-peer reviewed) books dedicated to startups, which quickly became best-sellers (see Appendix **??** for detailed books review).
- Finally, all of the above mentioned results fostered our motivation in pursuing research in this field. In particular, we observed that most studies focused on *mature* startups and we couldn't identify any relevant empirical evidences discussingengineering activities in the very early-stage of the startup creation. This factor contributed in driving the direction of our case study towards a cross-sectional analysis in the time frame that goes from the idea conception to the first release of the product.

[13] As remarked by Coleman, interviewing engineers in mature companies can be misleading to understand the higher-level dynamics, as they might not be aware of all process issues. By contrast, in our research, respondents have been wearing multiple hats during the first stages, acting as managers and engineers at the same time.

7. Work Practices in startups

//this section is about xyw and is organized as follows, according to our classification schema we will analyze how the literature discuss work practices in startups

Work practices	
Software Development	90
Managerial/organizational	59
Process management	44
Tools and technology	6
Sum	199

7.1. Overview of Software development in Startups

In this chapter we present the most relevant studies contributing to the formation of a *body of knowledge* focused on engineering activities in software startups. The materials reviewed here were mostly collected during the execution of a systematic review of the literature (detailed in Section **??**).

The word *startup* appeared in the SE literature for the first time in 1994 in an article written by Carmel [65] where he studied the *time-to-completion in young package firm*[14]. He noticed how these companies were particularly innovative and successful, advocating for the need of more research investigating software development practices so as to replicate success and try to transfer it to other technology sectors.

Only a few engineering studies in this specific area have been published in the years that followed (see Chapter **??**). Moreover the studies we identified in the systematic review appear to be highly fragmented and spread across different areas rather than constituting a consistent *body of knowledge*. In fact, we were able to identify only four empirical SE studies published prior to 2012 which are entirely dedicated to the topic of software development approaches in startups, designed executed and presented in a rigorous way[15]. In the remaining part of this section we provide an overview of the related works that we have identified during our research.

First of all, a research published in 2000 by Sutton [2] confirmed a general lack of studies in this area, claiming that *"software startups represent a segment that has been mostly neglected in process studies"* and it has been further confirmed with the empirical studies of Coleman et al. [7, 17, 33] eight years later.

One of the most prolific SE researcher in the area of startups is Gerry Coleman. He started working with irish startup companies and developing a *"lightweight software process for startups based on agile practices"* [99], which has been presented at a conference in 2004 [100] but apparently the research evolved into a different direction[16]. Indeed, his attention moved from startups to small enterprises [10]. In fact, he published an article titled *"Using grounded theory to understand software process improvement"* [33], which includes detailed explanation of the research methodology undertaken for his analysis. His results show different factors that influence and hinder the formation of processes in startups and small companies.

First insights reveal how software startups are product-oriented in the first period of their development phase [19]. Despite good achievements at the beginning, software development and organizational management increase in complexity [101, 102] causing deterioration of performance over time. Briefly, the necessity of establishing initial repeatable and scalable processes cannot be postponed forever[17].

A study of Kajko-Mattsson [9], which investigated a Swedish software startup, reported a heavy lack of requirements gathering process, minimal project management, lack of control over the change requests, absence of documentation to track the status and progress of the process and defective releases. Accordingly, Ambler et al. report how two startups approaching to an upcoming IPO started to require processes to focus on scalable solutions, in view of the growing companys size in terms of users and employees [77]. In this regard, Crowne, in [4], specifies different stages through which software startups evolve. Starting without any established workflows, startups grow over time, creating and stabilizing processes to eventually improving them only when sufficiently mature.

As studied in [71], the maturity of a company affects the extent to which processes are adopted. The author reports how introducing Extreme Programming (XP) principles [104] in the development process was challenging because of the need of trained team-members for fully implementing the methodology[18]. In fact, [66] was able to start with all the XP practices in place only after six months of coaching the team, trying to enhance maturity from day-one. Nevertheless, even then, customization of practices were inevitably implemented to adapt the processes to the undertaken startups context [72].

But when startups have no time for training and orienting activities, as discussed in [2], their main focus remains on team capabilities instead of prescriptive processes hiring people who can *"hit the ground running"* [86]. Empowering the team and focus on methodological attributes of the processes oriented in prototyping, proof-of-concepts, mock-ups and demos, to test basic functionalities, have been the primary priority in startups as described in [65]. Only when they grow, formal methodologies arise, followed by a conduction of quality assurance and long-term planning processes [86].

As partially discussed above, contributions to flexibility and reactiveness of the development process has been conducted prominently by means of *Lean* [105] and *Agile* [106] methodologies (also reported in [68, 79]), where the extreme uncertain conditions lead startups to learn fast from trials and errors with a strong customer relationship in order to avoid wasting time in building wrong functionalities and prevent rapidly exhaustion of resources [67, 83, 2]. Customer involvement in software development has also been discussed in [76] as important factor to encourage an early alignment of business concerns to the technology strategies, because both are salient considerations to be successful [71].

When startups take a development approach that is mainly product-oriented rather than process-oriented, it is essential to have a flexible team, with a workflow that helps them quickly to change direction according to the target market [2]. In this regard, many startups have been focused on team productivity, providing more control to the employees instead of providing them rigid guidelines [22, 23, 24].

Then, when startups are mature enough to support software process improvement (SPI), the solutions considered according to the state-of-the-art are oriented towards light-weight processes such as a design of development process based on XP, proposed by Zettel in [69]. The process consists of a set of activities and artifacts (in addition to some important roles) defined in order to identify responsibilities and tools

[14] The software development challenges of a startup have changed dramatically in the last 20 years.

[15] Moreover the studies under consideration [7, 17, 33, 9] have investigated mainly mature startups, whilst the empirical research we performed is focused on early-stages startups. This issue is further discussed in the analysis of the systematic literature review (see Section **??**)

[16] Coleman has been personally contacted and he confirmed that the lightweight process in question has not been further adopted neither *"developed into the later research"*

[17] This has been confirmed by Peter Thiel, co-founder of *Paypal*, *Asana* and other successful businesses, who declared that *"there is no real chance of setting things up correctly such that the rest unfold easily. But you should still get the early stuff as right as possible"[103].*

[18] According to XP creator Kent Beck, to be effective XP requires to be carefully applied : *"If you follow 80 percent of the process, you get just 20 percent of the benefits"* [104].

to utilize. But, despite the promising benefits reported by Zettel, we were not able to identify any future evaluation in real-world settings.

Another attempt of SPI in startups has been conducted by Deakins et al. introducing a *Helical Model for managing e-commerce development environment* [64]. Also in this case, the author prescribed broad guidelines for a rapid high-quality development process, which underwent limited testing only in academic settings.

The first publication mentioning the problem of *one-size-fits-all*, related to the SPI representations for startups, is described in [88]. The author reveals the problem in actuating the same *best-practices* criteria for established companies in 10-person software startups. Thoroughly remarked in [2], Sutton states that problems of SPI in software startups arise because of: the dynamism of the development process, which precludes repeatability; organizational maturity, that cannot be maintained from startups in view of lack of corporate direction; severe lack of resources, both human and technological for process definition, implementation, management, training . . . ; in conclusion, the primary benefits of SPI do not address startups, which instead of promoting product quality, aim to minimize time-to-market.

Additionally, the role of SPI has always been neglected because seen as an obstacle to the development teams creativity and flexibility as described in [17] and to the need of a quick delivering product process environment. In fact, product quality is left aside in favour of minimal and suitable functionalities to shorten the time-to-market. As reported in two studies of Mater and Mirel [78, 82], quality aspects, mostly taken in consideration in internet startups, are oriented to usability and scalability, even though the market and application type heavily influences the quality-demand [7, 84].

Finally, to maintain the development activities, oriented to limited but suitable functionality, many studies suggest to externalize the complexity of parts of the project to third party solutions by means of outsourcing activities, software reuse and open-source strategies [63, 70, 85, 87].

In conclusion, since *"all decisions related to product development are trade-off situations"* [83], generally startups optimize workflows to the dynamic context they are involved into. In fact they typically adopt any development style that might work to support their first needs in what is called the *"Just do it"* school of software startups [92]. Additionally, as remarked by Coleman, *"many managers just decide to apply what they know, as their experience tells them it is merely common sense"* [7].

To summarize - although a number of studies have been discussed in this section - the existing material appears to be inadequate to deal with the increasing importance of startups demands. In fact, as confirmed by the analysis of results of the systematic review (see Section **??**) this area appears to be, to some extent, immature. Nonetheless, we are recently assisting an impressive proliferation of books (such as [91, 92, 15, 107]) and pseudo researches [108], which are gaining traction among practitioners. For this reason we have dedicated a subsection of Appendix **??** to review the most relevant part of the *grey literature* which affects our research.

7.2. Process management practices

The process management represents all the engineering activities used to manage product development in startups. A study of Sutton [2] recognized the need of flexibility to accommodate frequent changes in the development environment, and reactiveness to obtain rapid and timely response in applying methodologies. Moreover, in a startup all the activities are conducted towards the development of the product and not in improving processes mainly because of scarce resources [19]. Coleman in [17] reports that the activities related to process improvement and process assessments are neglected since they represent a waste of time and money for startups [88, 7].

Agile methodologies have been considered the most acceptable processes given their certain degree of freedom in quickly changing direction following the business strategy [68]. In this context, fast releases with an iterative and incremental approach shorten the lead time from idea conception to production with fast deployment [68]. The benefits of having weekly releases and frequent build cycles to help addressing the uncertainty of the market has been further reported by Tingling [71], Ambler [77] and Silva [66].

In order to shorten time to market, prototyping is essential according to [64, 65]. To allow better prototyping activities, evolutionary workflows are needed to implement "soft-coded" solutions in the first phases until the optimal solution is found [64, 2].

Coleman in [33] reports how XP is the most used development methodology across startup companies because of the recuded process cost and low level of documentation requirements compared to serial and rigorous methodologies. Nevertheless, the same author confirms that to implement practices effectively, they need to be tailored to the specific features that characterize each development context [33, 7]. This concurs with the practices of allocating varying effort for formalizing specifications, design, documentation and testing in tailored development methodologies [77, 69, 65] that emphasize the importance of minimal process management.

Finally, to explain this orientation towards of flexible and reactive development approach, startups can be characterized by applying the Cynefin framework [109]. Within this framework, startups cross the complex and chaotic domains. Those two domains represent the areas where applying rigorous process management to control development activities is not effective since solutions are not predictive. Instead, flexible and reactive methods, designed to stimulate customer feedback, increase the number of perspectives and solutions available to a decision maker. Nevertheless, it is not by chance that software startups are more disruptive than established and bureaucratic companies. Moving from complex to chaotic spaces, software startups open up new possibilities of creation, generating the condition for innovation. This dynamic is named divergence-convergence, and it is one of the many movements described by the Cynefin framework.

//Summarizing paragraph MISSING: - You need here a short paragraph summarizing your analysis. Mind you: the summary is not about what you have done, but what can be learned ("take away" value) from your analysis.

7.3. Software development practices

Software Development	
RE	19
Analysis	14
Implementation	5
QA	21
Deployment	2
RE and Analysis	1
Sum	62

7.3.1. Requirements Engineering

7.3.2. Design and Architecture

7.3.3. Implementation and maintenance

7.3.4. Quality Assurance

7.3.5. Other software development practices

7.4. Managerial and organizational practices

Documentation	
SD Documentation	3
M Documentation	4
PM Documentation	1
Sum	8

7.5. Other practices

//should be about tools and technologies

8. Conclusions and future work

This text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusions.

This text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusions.

This text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusions.

This text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusionsThis text should be replaced with the conclusions.

Appendix .1. Search Strings

The search strings, utilized to retrieve relevant studies (see Subsection **??**), have been adapted to the underlying search technology, as shown in Table .14.

String A - Compendex/Inspec [44]
('early-stage firm' OR 'early-stage company' OR 'high-tech venture' OR 'high-tech ventures' OR 'high-tech start-up' OR 'high-tech start-ups' OR 'high-tech startups' OR 'high-tech startup' OR 'start-up company' OR 'start-up companies' OR 'startup company' OR 'startup companies' OR 'software startup' OR 'lean startup' OR 'lean start-up' OR 'lean startups' OR 'software startups' OR 'software package startups' OR 'software package start-up' OR 'software package start-ups' OR 'software package startup' OR 'IT start-ups' OR 'IT start-up' OR 'IT startup' OR 'IT startups' OR 'software start-up' OR 'software start-ups' OR 'software product startup' OR 'software product startups' OR 'software start up' OR 'web startup' OR 'web start-up' OR 'web startups' OR 'web start-ups' OR'internet startup' OR 'internet start-up' OR 'internet startups' OR 'internet start-ups' OR 'mobile startup' OR 'mobile start-up' OR 'mobile startups' OR 'mobile start-ups')AND (develop* OR engineer* OR model* OR construct* OR implement* OR cod* OR creat* OR build*) AND (software OR product* OR service* OR process* OR methodolog* OR tool* OR method* OR practice* OR artifact* OR artefact* OR qualit* OR 'non-functional requirement' OR 'non-functional requirements' OR ilit* OR strateg*)
String B - IEEE xplore [46]
('early-stage firm' OR 'early-stage company' OR 'high-tech venture' OR 'high-tech ventures' OR 'start-up company' OR 'start-up companies' OR 'startup company' OR 'high-tech start-up' OR 'high-tech start-ups' OR 'high-tech startups' OR 'high-tech startup' OR 'startup companies' OR 'software startup' OR 'lean startup' OR 'lean start-up' OR 'lean startups' OR 'software startups' OR 'software package startups' OR 'software package start-up' OR 'software package start-ups' OR 'software package startup' OR 'IT start-ups' OR 'IT start-up' OR 'IT startup' OR 'IT startups' OR 'software start-up' OR 'software start-ups' OR 'software product startup' OR 'software product startups' OR 'software start up' OR 'web startup' OR 'web start-up' OR 'web startups' OR 'web start-ups' OR'internet startup' OR 'internet start-up' OR 'internet startups' OR 'internet start-ups' OR 'mobile startup' OR 'mobile start-up' OR 'mobile startups' OR 'mobile start-ups') AND (development OR developing OR engineer OR engineering OR model OR construct* OR implement* OR cod* OR creat* OR build*) AND (software OR product OR products OR service OR services OR process OR processes OR artifact* OR artefact* OR quality OR qualities OR 'non-functional requirement' OR 'non-functional requirements' OR ilities OR methodology OR methodologies OR tool OR tools OR method OR methods OR practice OR practices OR strategy OR strategies)
String C - Scopus [48]
ABS(('early-stage firm' OR 'early-stage company' OR 'high-tech venture' OR 'high-tech ventures' OR 'start-up company' OR 'start-up companies' OR 'high-tech start-up' OR 'high-tech start-ups' OR 'high-tech startups' OR 'high-tech startup' OR 'startup company' OR 'startup companies' OR 'software startup' OR 'lean startup' OR 'lean start-up' OR 'lean startups' OR 'software startups' OR 'software package startups' OR 'software package start-up' OR 'software package start-ups' OR 'software package startup' OR 'IT start-ups' OR 'IT start-up' OR 'IT startup' OR 'IT startups' OR 'software start-up' OR 'software start-ups' OR 'software product startup' OR 'software product startups' OR 'software start up' OR 'web startup' OR 'web start-up' OR 'web startups' OR 'web start-ups' OR'internet startup' OR 'internet start-up' OR 'internet startups' OR 'internet start-ups' OR 'mobile startup' OR 'mobile start-up' OR 'mobile startups' OR 'mobile start-ups')AND (develop* OR engineer* OR model* OR construct* OR implement* OR cod* OR creat* OR build*) AND (software OR product* OR service* OR process* OR artifact* OR artefact* OR qualit* OR 'non-functional requirement' OR 'non-functional requirements' OR ilit* OR methodolog* OR tool* OR method* OR practice* OR strateg*))

Table .14 – *Continued on next page*

Table .14 – *Continued from previous page*

String D - ISI Web of Knowledge [47]
TS=((’early-stage firm’ OR ’early-stage company’ OR ’high-tech venture’ OR ’high-tech ventures’ OR ’start-up company’ OR ’start-up companies’ OR ’high-tech start-up’ OR ’high-tech start-ups’ OR ’high-tech startups’ OR ’high-tech startup’ OR ’startup company’ OR ’startup companies’ OR ’software startup’ OR ’lean startup’ OR ’lean start-up’ OR ’lean startups’ OR ’software startups’ OR ’software package startups’ OR ’software package start-up’ OR ’software package start-ups’ OR ’software package startup’ OR ’IT start-ups’ OR ’IT start-up’ OR ’IT startup’ OR ’IT startups’ OR ’software start-up’ OR ’software start-ups’ OR ’software product startup’ OR ’software product startups’ OR ’software start up’ OR ’web startup’ OR ’web start-up’ OR ’web startups’ OR ’web start-ups’ OR’internet startup’ OR ’internet start-up’ OR ’internet startups’ OR ’internet start-ups’ OR ’mobile startup’ OR ’mobile start-up’ OR ’mobile startups’ OR ’mobile start-ups’)AND (develop* OR engineer* OR model* OR construct* OR implement* OR cod* OR creat* OR build*) AND (software OR product* OR service* OR process* OR artifact* OR artefact* OR qualit* OR ’non-functional requirement’ OR ’non-functional requirements’ OR ilit* OR methodolog* OR tool* OR method* OR practice* OR strateg*))
String E - ACM [45]
((Abstract:(’early-stage firm’ OR ’early-stage company’ OR ’high-tech venture’ OR ’high-tech ventures’ OR ’start-up company’ OR ’start-up companies’ OR ’startup company’ OR ’startup companies’ OR ’software startup’ OR ’lean startup’ OR ’lean start-up’ OR ’high-tech start-up’ OR ’high-tech start-ups’ OR ’high-tech startups’ OR ’high-tech startup’ OR ’lean startups’ OR ’software startups’ OR ’software package startups’ OR ’software package start-up’ OR ’software package start-ups’ OR ’software package startup’ OR ’IT start-ups’ OR ’IT start-up’ OR ’IT startup’ OR ’IT startups’ OR ’software start-up’ OR ’software start-ups’ OR ’software product startup’ OR ’software product startups’ OR ’software start up’ OR ’web startup’ OR ’web start-up’ OR ’web startups’ OR ’web start-ups’ OR’internet startup’ OR ’internet start-up’ OR ’internet startups’ OR ’internet start-ups’ OR ’mobile startup’ OR ’mobile start-up’ OR ’mobile startups’ OR ’mobile start-ups’)) AND (Abstract:(develop* OR engineer* OR model* OR construct* OR implement* OR cod* OR creat* OR build*)) AND (Abstract:(’software’ OR product* OR service* OR process* OR methodolog* OR tool* OR method* OR practice* OR artifact* OR artefact* OR qualit* OR ’non-functional requirement’ OR ’non-functional requirements’ OR ilit* OR strateg*)))
String F - Google Scholar [49]
(’software startup’ OR ’software startups’)AND (develop* OR engineer* OR cod* OR creat*) AND (software OR product* OR process* OR methodolog* OR tool* OR method* OR practice* OR artifact* OR qualit* OR ’non-functional requirements’ OR strateg*)

Table .14: Search strings

Appendix .2. Selected studies overview

We extracted a brief *one-line* sentence which summarize the content of each study and can be used by the reader to grasp the idea behind the articles without reading the full text. A set of keywords have been assigned to each article during the initial stages of creation of the classification schema (see analysis of results in Section **??**). The result of this process is presented in Table .15.

Ref.	Author (year)	One-line contribution	Keywords
[7]	Coleman (2008)	' [...] the previous experience of the person tasked with managing the development work is the prime influencer on the process a company initially uses. Other influencers include the market sector in which the company is operating, the style of management used and the size and scale of the company operations.'	Software Process, Process formation, Resources, Founder, XP, RUP, Agile Methodologies.
[33]	Coleman (2007)	'Background of Software Development Manager was central to the initial process that a software company used.'	Software Process, Software Process Improvement, Grounded Theory, Factors Influencing Process.
[17]	Coleman (2008)	'Our research found that SPI programmes are implemented reactively and many software managers are reluctant to implement SPI best practice models because of the associated costs.'	Software Process, Software Process Improvement, Grounded Theory, Factors Influencing Process.
[9]	Kajko-Mattsson (2008)	By applying their process in a start-up they obtained good results: manage requirements, define development release, and control releases, improve quality.	Process improvement, Software Process, Release Management, Communication, Maintenance, Founder, early-stage, Improve Quality.
[62]	Hásel (2010)	'[...] the competence profiles preferred by founders with innovative products differ from those preferred by founders with less innovative products.'	Founder Background, Founder Teams, IT competence, Team, Know-how.
[63]	Hanna (2010)	The model presented in this paper offers a powerful tool for understanding the challenges of offshoring	Outsourcing, Offshore, Know-how, Management, decision-making.
[64]	Deakins(2005)	'The dotcom development environment is highly volatile and requirements can change rapidly in response to competitor offerings and customer needs; customers are unreliable predictors of their future needs. Need of multi-skilled teams, adaptiveness over efficiency, rapid development, early-stage, experimentation, improvement based on customer'	Innovation, management, software Process, volatile environment, time-to-market, rapid Development,web-development.
[65]	Camel (1994)	'The presence of several other time-to-completion accelerators appears to be weak in software startups: they did not fully use development methodologies, they made little for increasing use of software tools to increase productivity, weak risk analysis and project control'	Time-to-market, Package Software, Founder, Software Process, Tools.
[66]	Silva (2005)	'We have successfully used all of XP practices, adopted most of them and even came up with some unique practices of our own.'	XP, Software Process, Agile methodologies, Practices, Adaption, Rapid Changes.
[67]	Midler (2008)	'While exploitation provides vital short-term resources,exploration enhances the adaptation of the organization to a changing environment because it increases the variance of organizational activities'	Management, Multi-project, Software Process, Maturity, Learning, Internet, Founder, Uncertainty.
[68]	Taipale (2010)	'Our workflow is predictable within acceptable variance and we can change direction of the business at any given time'	XP, Lean, Agile Methodologies, Software Process, Pivoting.
[23]	Chorev (2006)	'Marketing is very important for success and is underestimated by product startups [...] core team competences is crucial as well'	Management, Software Process, Influencing Factors, Success, Marketing, early-stage, Team Competences.
[69]	Zettel (2001)	'This paper proposes a lightweight software process for a specific application domain (i.e., database-and user-interface-oriented off-the-shelf e-business applications).'	Process Improvement, XP, Agile Methodologies, Lightweight process, IT, Project Management.
[70]	Jansen (2008)	'Here, we describe two start-ups that have opportunistically and pragmatically developed their products, reusing functionality from others that they could never have built independently.'	Reuse, Product-line, Unstructured, Method, third-party, Founder, IT, COTS.
[2]	Sutton (2000)	'Startups represent a software industry segment that has been mostly neglected in process studies, and it is possible that lessons drawn from start-ups also apply to other develop-ment organizations.'	Software Process, Process formation, early-stage,Time-to-market, Resources.
[19]	Heitlager (2007)	'These companies start very ad-hoc, trying to overcome the uncertainties of market, team and platform. The biggest struggle for these companies is to survive with only scarce resources.' - This paper provides a matrix to analyze the dynamics of the maturity of product development.	Software Process, early-stage, Innovation, Internet, Process Improvement, Product Development.
[71]	Tingling (2007)	'Small releases, on-site customer, continuous integration and refactoring were most vigorously advanced by management and adopted by developers. Paired programming on the other hand was culturally avoided.'	Rapid Development, XP, IT, Software Process, Internet, Practices, Agile methodologies.
[72]	Deias 2002)	'We are enthusiastic about XP, and it is difficult for us to imagine a software project where we should not try to use XP, at least in the domain of Internet development.'	Software Process, Agile Methodologies, XP, Web Development, Risk Management, Project Management, Internet, Quality Assurance, Business, Founder,Customer relation.
[73]	Stanfill (2007)	'To improve the chances of successfully adopting a new technological innovation and boosting entrepreneurial team performance, we propose an improved way to select suitable technologies, better timing for delivering market-driven requirements to product designers, and enhanced understanding of the implications of business and technical decisions with regards to impact on intellectual property.'	Team Performance, Market Feasibility, Management, Market-driven Requirements, early-stage, Founder,Technology-Driven decisions, IT.
[74]	Wood (2005)	'Taken together, these strategies provide guidance to entrepreneurs, board members and business and engineering managers of startups for the effective use of Open Source Software.'	Open Source, Release, Software development, early-stage,Strategy, Internet, Cost-reduction,License.

Table .15 – *Continued on next page*

Table .15 – *Continued from previous page*

Ref.	Author (year)	One-line contribution	Keywords
[75]	Steenhuis (2008)	'It is therefore unlikely that follower regions or nations are able to catch-up with the leading regions or nations unless the leading regions or nations enter the high portion of the S-curve, i.e. their economic growth slows down.'	Innovation, Technological Development, Critical Mass,Business,Economic Growth Internet, Business.
[76]	Yogendra (2002)	'With the role of technology varying from 'enabler' to 'driver' of the business strategy, business and technology strategies need be in close alignment'	Management, Technology Driven Decisions, Planning, Monitoring, Quality.
[77]	Ambler (2002)	'I wondered whether the rules of software development had also changed. Were we witnessing a paradigm shift in the way we develop software?'	Agile Methodologies, Web Development, RUP, Comparison, Paradigm-shift.
[4]	Crowne (2002)	'A model for the evolution of product development from startup to maturity is provided, consisting of three phases:phases: Startup ,Stabilization , Growth [...] Successful development of new software products is a key value driver for many startup companies.'	Software Process, Founder, Life-Cycle, Internet, Maturity.
[78]	Mater (2000)	'Short-time-to market, fast growth, changing requirement are Entrepreneurs dream and Engineer nightmare'	Management, Business,Web Development, Quality, UX, Time-to-Market.
[24]	Kakati (2003)	'Product uniqueness was shown not to be a significant factor in determining initial success, despite the tendency of high-tech firm to emphasize RnD and technological excellence'	Management, Success Criteria, Risk.
[79]	Kuvinka (2011)	'Scrum involves many meetings, much planning overhead, and time-consuming team collaboration. Is it possible for a single writer to keep up?'	Management, Internet, Business, Agile Methodologies, early-stage, Scrum, Kanban.
[80]	Su-Chuang (2007)	' [...] a properly constructed value proposition is essential to the value creation process in e-business, and value is essential to the value creation process in e-business, and value co-production is the building blocks for value protection mechanism in network economy'	Value Proposition, Web Development, E-Business, Internet, Business, Value,Co-production.
[81]	Sau-ling Lai (2010)	'Technology is not Alibaba's core competency (non tech-founder) [...] Customer First, employee next [...] Small is beautiful'	Web Development, Founder Background, Know-how, Value Proposition, Finance, Customer Relation, IT, Internet, Product Design , Business.
[82]	Mirel (2000)	' [...] usability improvements depend on more than innovative and user-centered technical designs and implementations. Equally important for creating useful and usable software are the social and political forces that shape the development context.'	Organizational Factors, Usability, Political Support, Sociology, IT, Conflicts, Internet, Innovation.
[83]	Himola (2003)	'On the basis of the results of this article, it is suggested that the improvement of product development lead time is one of the most important parameters in the software startup environment [...] all decisions related to product development are tradeoff situations.'	Time-to-market, Improvement,Business, Management, Finance.
[84]	Kim (2005)	'Initial trust is regarded as a critical factor for many e-businesses to succeed in the business-to-customer e-markets, especially startups, because it creates initial relationships with customers.'	Trust, Customer Relation, Initial Trust, b2c, E-commerce, Internet, early-stage, Quality.
[85]	Wall (2001)	'When money are scarce, OSS can help your business launch without breaking your budget'	Open Source, Tool, Java, Software development, License, Distributed Development, Cost-reduction.
[86]	Yoffie (1999)	'That youthfulness also helps to explain why most start-ups fail: exuberance can only get you so far. Jim Clark and Marc Andreessen made a conscious choice to scale the company with a different type of person. They targeted maturity as well as technical expertise.'	Management, Internet, Web Development, Team Formation.
[87]	Bean (2005)	'Smaller firms like Aperture Technologies Inc. are using wikis to brainstorm, track projects, write and edit documentation, and coordinate marketing. Software startups like Stata Lab-oratories Inc. are using wikis to lower teleconferencing costs for outsourced engineering to India!'	Open Source, Method, Tool, Internet, Communication, IT, Knowledge Management, Management.
[22]	Tanabian (2005)	'Because of the small size of the firm, the amount of uncertainty of its business, and lack of financial strength, Many practices in place may appear to be in contradiction with guidelines for a productive and healthy job.'	Management, Founder, Job Design, Business, Team Performance, IT, Features, Workload, Internet.
[88]	Fayad (1997)	'The process should be treated differently from startups to established companies [...] 'startup effect' in which new initiatives get much more highly qualified and motivated people than standard projects, and the idea of 'heroic efforts' '	Software Process, IT, Process Improvement, Developers Skill, Resources Scarcity, Internet, Motivation.

Table .15: Mapping study - One line content review

Appendix .3. Ranking quantification

In this appendix we refer to the process of ranking the selected studies as discussed in *Research Methodology*, specifically in Subsection **??**. The final score has been computed by summing up contributions from eight dimensions: age; rigor; relevance; venue; pertinence; contribution type; research type; and focus. For each dimension we defined conversion tables to quantify our criteria, i.e. - assigning higher scores to recent rigorous journal articles entirely devoted to the topic and presenting empirical results relevant to practitioners.

The left table (a) shows the scores assigned to each subcategory of the classification schema, while the table on the right side (b) shows the scores associated to the remaining dimensions.

Rigor	
Ri	Score
3	10
2.5	8.33
2	6.67
1.5	5
1	3.33
.5	1.67
0	0

Relevance	
Re	Score
4	10
3	7.5
2	5.
1	2.5
0	0

Age	
Age	Score
$[0, 2]$	10
$[3, 5]$	8
$[6, 9]$	6
$[10, 14]$	4
$[15, 40]$	1

Venue	
Venue	Score
Journal Article	10
Conference Proceeding	7
Magazine Article	6

(a) Other dimensions

Research	
Type	Score
Evaluation Research	10
Solution Proposal	6
Philosophical Papers	3
Opinion Papers	3
Experience Papers	3

Focus	
Type	Score
Software development	10
Process management	10
Tools and technology	8
Managerial/organizational	6

Pertinence	
Type	Score
Full	10
Partial	5
Marginal	3

Contribution	
Type	Score
Model	10
Theory	10
Framework/Methods	8
Guidelines	6
Lesson learned	6
Advice/Implications	6
Tool	6

(b) Classification schema

Figure .13: Conversion table for the scoring function

The results of the final ranking of studies is presented in Subsection **??**, while limitations of this approach are discussed in Subsection 3.8.3.

Appendix .4. Work practices extraction

Version one

References	One-line comment	Work practice	Category	Evidence provided
[85]	The open source community helped in support product development with no costs	Rely on open source community support	Managerial/organizational	1
[68]	To find subjects for improvements	Measure lead time and cycle time	Managerial/organizational	0
[71]	40-h workweek (be flexible but regular workdays)	40-h work week	Managerial/organizational	1
[71]	Code was officially shared but developers exhibited possessiveness	Collective Ownership	Managerial/organizational	1
[71]	Communication simple, formal but ambiguous	System metaphor	Managerial/organizational	1
[23]	High communication between development and marketing team to facilitate spotting product/market fit.	Communication marketing-engineering;	Managerial/organizational	2
[77]	Collective ownerships under SVN of the architectural docs	Collective Ownership of architecture	Managerial/organizational	2
[69]	Documentation specialist writes an online help and user guide book	Create user doc	Managerial/organizational	2
[69]	Documentation is not viable in the short term but essential for the long run	Documentation	Managerial/organizational	2
[78]	Costumer in the internet are not tollerant, and competitve solutions are just one click away	Plan for robustness	Managerial/organizational	1
[9]	Plan to release the product	Release deployment	Managerial/organizational	2
[9]	Assignation of tasks to developers	Release development	Managerial/organizational	2
[9]	Start with the release scope preparation defining functionalities to include. Delay has been decreased	Release management process	Managerial/organizational	2
[9]	prioritization of requirements, release cycle has been improved by planning	Release planning	Managerial/organizational	2
[77]	Chef-architect makes models visible to every developers and collect feedbacks	Visibility of the architectures	Managerial/organizational	2
[69]	Everyone collect metrics (testing, progress, velocity, quality)	Collect metrics	Managerial/organizational	2
[77]	Because of growth	Hiring new developers	Managerial/organizational	2
[69]	Project manager bases his decisions (effort estimations, schedules and priorities) on collected metrics	Project manager takes decisions for PM	Managerial/organizational	2
[73]	Engineering students developed a technology in parallel with the market team, and frequently independently. This was an issue that limited the project.	Separate development from market concerns	Managerial/organizational	1
[83]	Lead time as the main driver for product development	Time-to-market as driver for the development	Managerial/organizational	2
[22]	Early stage startups demand initial more working hours, more stress and responsabilities in comparison to established companies.	Working over time	Managerial/organizational	0
[33]	Necessity of skipping documentation, lead to excess of tacit knowledge	Documentation	Managerial/organizational	2
[33]	The size of the company influences the attitude towards documentation: the smaller the company the greater the hostility towards documentation	Documentation	Managerial/organizational	2
[17]	Necessity of skipping documentation, lead to excess of tacit knowledge	Documentation	Managerial/organizational	2
[65]	little project management	No project management	Managerial/organizational	1
[65]	No use of traditional project planning and poorly planned development		Managerial/organizational	1
[2]	Empower team improve quality of product, adaptability of process,efficiency and coordination.	Empower team;	Managerial/organizational	1
[4]	focus on single product, create a new version if need customizations.	Focus on single-product	Managerial/organizational	1
[4]	Plan objectives short-medium term	Plan objectives	Managerial/organizational	1
[76]	Monitor the process continuously to verify the alignment with the business strategy	Monitor process;	Managerial/organizational	1
[75]	Select the right skills in people	Skill team;	Managerial/organizational	0
[23]	Core Team expertise, diversified knowledge and harmony are essential for success. Many angels and VCs highlighted the assessment of the core team in investment decision making. Very often start-ups are founded by young people who themselves lack management skills and experience but do not hire suitable managers. This creates difficulties in both R&D and in the marketing processes. At certain stages, where the start-up lacks expertise, consultants can be useful.	Team skills vital to success	Managerial/organizational	2
[33]	Co-locate the development team to reduce the costs increasing the sharing of the tacit knowledge	Co-locate the development team	Managerial/organizational	2
[77]	Modelling and documentation were separated efforts.	Dedicated people to docs and modelling	Managerial/organizational	2
[77]	Sharing knowledge and collaboration	Developers and testers were close	Managerial/organizational	2
[77]	When scalability problems arise	Hire chief-architect	Managerial/organizational	2
[86]	Hire developers with self starter personalities with past experience in similar domain	Need of experience in similar domain and self starter personalities	Managerial/organizational	2
[86]	Youthness of the founders contribute to failure rates. Startups need to hire technical and expertise	Need of expertise	Managerial/organizational	2
[22]		No separation between designers and developers	Managerial/organizational	0
[62]	The competence of founders working with innovative products are different with non innovative	Prefer IT experts in the founding team	Managerial/organizational	2
[22]	People can and encouraged to work at home but with more control. Due to the large amount of work.	Work from home with flexible hours.	Managerial/organizational	0
[67]	Team capability should be able to absorb and learn from trial and error quickly to avoid prolonged time	Learn quickly from trial and error	Managerial/organizational	1
[7]	Empower team to Enhance creativity and flexibility	Empower team	Managerial/organizational	2
[17]	Empower team to Enhance creativity and flexibility	Empower team	Managerial/organizational	2
[2]	Skilled team, because developer should be able to adapt to new roles and face new challenges everyday	Skilled team;	Managerial/organizational	1
[65]	Importance of the core team: highly motivated, homogeneous, history of working together, small, loose organizational structure	Get the right team	Managerial/organizational	1
[65]	working extra hours	Work overtime	Managerial/organizational	1
[23]	Characteristic of the entrepreneur plays a primary role in maximising the scarce resources	Entrepreneur skills primary;	Managerial/organizational	2

Table .16 – *Continued on next page*

References	One-line comment	Work practice	Category	Evidence provided
[24]	Characteristic of the entrepreneur (Courage , enthusiasm, commitment, leadership) are more important than other technical factor for the success of a startup	Entrepreneur skills primary;	Managerial/organizational	2
[7]	Hire expertise with agile background to improve chances of success	Agile background	Managerial/organizational	2
[4]	Developers are inexperienced and they need to familiarize with software engineering practices to deal with multi-role full-stack challenges	Familiarize with all SE practices	Managerial/organizational	1
[33]	Hire expertise to supply lack of knowledge in critical moments	Hiring expertise	Managerial/organizational	2
[17]	Hire expertise to supply lack of knowledge in critical moments	Hiring expertise	Managerial/organizational	2
[4]	Promote developers showing both leadership and technical capacity	Endorse leadership AND technical capacity	Managerial/organizational	1
[86]	Makes specifications free available to everybody	Use of open standard	Managerial/organizational	2
[72]	high experience developers hard to buy-in XP practices	// // //	Managerial/organizational	1
[66]	The work week was not fully respected	40h work week	Managerial/organizational	1
[66]	Senior developers were more comfortable with this practice	Collective code ownership	Managerial/organizational	1
[66]	To solve conflicts among developers or with customers	Dirty laundry meeting	Managerial/organizational	1
[66]		Retrospectives	Managerial/organizational	1
[68]	The Agile process is predictable with acceptable variance, and change direction of the business of any given time	Agile	Process management	0
[68]	The lead time from idea to production is 8 days. Deployment every day	Fast release with continous deployment	Process management	0
[7]	Startups should tailor agile processes and practices since the cost of SPI (time and res.) is too high to afford	Agile process tailoring	Process management	2
[17]	Startups should tailor agile processes and practices since the cost of SPI (time and res.) is too high to afford	Agile process tailoring	Process management	2
[71]	weekly delivered, frequent build cycles	Small releases	Process management	1
[72]	Trying to tailor XP harms the effectiveness	(try) to fully follow XP	Process management	1
[9]	Poor documentation (no process documentation) led to poor control over the change request, that are postponed to next releases	Documentation	Process management	2
[64]	Just-in-time delivery practices, prototyping	Release fast	Process management	0
[79]		Definition of done	Process management	0
[69]		Plan iterations based on scenarios	Process management	2
[65]	Mock-up, prototypes, demo to customers in order to test functionalities	Prototype	Process management	1
[68]		Definition of done	Process management	0
[77]	6 to 12 weeks	Incremental release in short cycle	Process management	2
[68]	Product queue for listing the features, from 1 to 5	Kanban product queue	Process management	0
[79]		Kanban wall	Process management	0
[77]	Tailoring rup to meet specific necessity	RUP	Process management	2
[79]	2 weeks	Sprint base approach	Process management	0
[68]	Maintain the buffer small enough	The number of features up to seven	Process management	0
[79]		WIP metric, 3 tasks at the time	Process management	0
[77]	6 to 12 weeks	Highly iterative process	Process management	2
[88]	Process assessment is considered wasteful	Process assessment	Process management	0
[69]	Viewed as obsolete and irrelevant	Process improvement	Process management	2
[33]	Tailor process to startups' context	Process tailoring	Process management	2
[7]	Tailor process to startups' context	Process tailoring	Process management	2
[33]	XP is the most used across organizations because of the least process cost and low level of documentation requirements	Use XP	Process management	2
[69]	Tailoring of XP but pair programming but with documenting practices	XP practices	Process management	2
[2]	Process definition only high level: leave details at low level to the team.	High-level work-flow;	Process management	1
[17]	Minimum processes might work	Minimum process	Process management	2
[64]	Evolutionary prototyping with improvements	Prototyping	Process management	1
[64]	Rapid development processes	Rapid development;	Process management	1
[2]	Evolutionary work-flows at high level to allow prototype;	Evolutionary work-flows;	Process management	1
[4]	Start introducing repeatable processes when grow starts to parallelism	Introduce repeatable process in growth	Process management	1
[79]	The company was applying scrumban. Scrumban proved to be a great organizational and motivational tool, but working well only if every one is doing it. Scrum emphises the customers and their needs.	Scrumban	Process management	0
[65]	Little effort allocated formalizing specifications, design, documentation and testing	Minimal engineering activities	Process management	1
[19]	Start with a product centric approach (ad-hoc) to avoid uncertainty of team market and platform, but at certain point the company should be ready to implement some process and scale	From product to process-centric approach;	Process management	1
[19]	A process is necessary to scale the company activities, and the startup should monitor the state of evolution of process vs. product to be able to take decision on when to introduce some process	Measure process maturity;	Process management	1
[65]	Engineering;marketing in parallel	Overlapping phases	Process management	1
[4]	Begin SPI when market shares and customers are established (mature)	Begin SPI after market stabilized	Process management	1
[24]	After building a first functioning prototype the team should focus on marketing instead of technical.	Prototype and market first;	Process management	2
[66]		Full XP;	Process management	1
[66]	One release every two weeks with one week of bug fixing	Small-release	Process management	1
[66]	used XP help trying to reduce time-to-market	XP practices	Process management	1
[66]	Even trying to implement XP, developers were more like cowboy coders.	Cow-boy coding;	Process management	1
[64]	Solutions should remain "soft-coded" in the first phases until the optimal solution is found	Prototyping	Process management	1
[68]		Code refactoring	Software development	0
[68]		Document new features if applicable, especially for interaction with third party	Software development	0
[68]		Integration test with CI	Software development	0

Table .16 – *Continued on next page*

References	One-line comment	Work practice	Category	Evidence provided
[68]		Peer-review critical paths	Software development	0
[68]	Helps find actionable issues	Root cause analysis	Software development	0
[68]		Simple design	Software development	0
[68]	Those that are related to users.	Trace metrics of product usage	Software development	0
[71]	Coding standards initially avoided but later implemented	Coding standards	Software development	1
[71]	Code was rarely broken and was continuously compiled	Continuous integration	Software development	1
[71]	Testing was continuous but without advanced scripts	Continuous testing	Software development	1
[71]	CEO and analytic director acted as customers	On-site customer	Software development	1
[71]	Programming were independent most of the time	Pair-Programming	Software development	1
[72]	Actively push developers during planning game to participate	Planning game	Software development	1
[71]	Value engineering balanced features against time and budget	Planning game	Software development	1
[71]	Modules were constantly improved, periodic burst and dramatic improvement occurred.	Refactoring	Software development	1
[71]	Working softer was favoured	Simple design	Software development	1
[4]	Stay close to customers during product development	Proximity to customer	Software development	1
[79]		Documentation is approached at the end of the user stories, often at the end of the sprint.	Software development	0
[65]	Software architecture not tangible (lack of documentation of modeling)	Documentation	Software development	1
[64]	Generate alternatives solutions and evaluate them	Analysis	Software development	1
[72]	Developing an architecture is difficult	Lack of architecture	Software development	1
[4]	Startups should capture new requirements, prioritize them, capture feasibility and value!	Requirement Engineering;	Software development	1
[9]		Acceptance testing	Software development	2
[69]	Customers use scenarios to judge whether the systems needs anticipated tests	Acceptance tests	Software development	2
[69]	Viewed as obsolete and irrelevant	Code metrics	Software development	2
[70]	Startups take advantage of existing components saving time-to-market. Extending software functionalities from a third party suppliers	Code reuse	Software development	2
[69]	Customer and developers collect user stories effort is estimated.	Collect scenarios	Software development	2
[77]	Define an infrastructure	Enterprise architectural modelling process	Software development	2
[67]	Feedback driven patterns to have an high interaction between internal and external factors. Customer provides key to action	Feedback-driven RE	Software development	1
[78]	Looking at the basic requirement process and provide raccomendations where the risk exists and what to be done for mitigating. Assessment of the scalability risk for basic architecture and design is very important for internet startups.	Initial risk assessment	Software development	0
[65]	No evidences of risk analysis conducted	No risk analysis	Software development	1
[77]	Only overview of the architecture. No details	Only high-level architecture.	Software development	2
[65]	Family of products built close together enhancing re-use	Re-use	Software development	1
[69]	Include code for unique test cases and write necessary test files for documentation to improve traceability.	Realize scenarios with TDD	Software development	2
[77]	Because of growth	Redevelopment of the system	Software development	2
[77]	Because of growth	Refactor architecture	Software development	2
[69]	Ensure readability and understandability of the code	Refactor system	Software development	2
[69]		Release system	Software development	2
[69]	Developers fix a defect described in an open issue report. Or improve quality	Rework code	Software development	2
[70]	SOA saved time and resources using third party services. But changing a component introduce substantial rework.	SOA using third party services	Software development	2
[9]	Contributed to improve faulty releases.	System testing	Software development	2
[65]	Formulating trade-off at the end of the development process in order to speed-up time to market	Trade-off analysis only in retrospective	Software development	1
[65]	Object orient development orientation for package firms	Object oriented development	Software development	1
[64]	Adjust requirements collecting customer feedback	Customer development	Software development	1
[77]	Benefits in having time of discussing and reflect with co-workers	Architecture conducted in parallel with development	Software development	2
[68]		Minimum marketable features (MMF)	Software development	0
[86]	The system should scale with the company size in harmony, early but not late	Prepare for scalability	Software development	2
[78]	Expecially scalability, robustness, speed of changing in technology	Understing key failure conditions	Software development	1
[2]	Developers should maximize the amount of code re-used that they are familiar with	Code re-use;	Software development	1
[80]	Co-create value with the customer	Costumer involvment	Software development	0
[68]		Customer development	Software development	0
[82]	But it didn't allow a proper feasibility study	Use unpredictability	Software development	1
[64]	high-quality product	Quality product	Software development	1
[67]	Postpone technological choices that are restrictive and can limit flexibility	Post-pone limiting decisions;	Software development	1
[78]	Quality assurance team are not formed until the product is not ready to be delivered. But at that time the practices are already established and difficult to change. QA is tipically inexperienced, lack of SE training and no business experience.	Elevate the software assurance function to report directly to the CEO	Software development	1
[86]	User adopters as quality assurance team	Involve users for QA	Software development	2
[78]	Increased quality from outsourcing	Outsourcing QA	Software development	0
[69]	Time spent on QA is reduced because of time-to-market pressure	QA	Software development	2
[22]	Not clear requirements. The design team must adapt to this frequent and changing	no risk analysis	Software development	0
[78]	Introduce a process to understand the costumer quality needs	Costumer feedbacks	Software development	1
[82]	Go to customers to learn their needs. It was a good approach to enhance usability of the product.	Customer involvement	Software development	1
[85]	To rank the companies' priorities around the customers' needs	Customer Involvement	Software development	2
[80]	Monitor and collect feedbacks	Customers feedback	Software development	0
[80]	Define what product or service you are going to provide	Value proposition	Software development	0
[80]	Differentiate your product from customers	Value proposition	Software development	0

Table .16 – *Continued on next page*

References	One-line comment	Work practice	Category	Evidence provided
[80]	Indentify costomer target are	Value proposition	Software development	0
[68]		Prioritization with time boxing. WIP is 2 at the time	Software development	0
[69]	Requirement engineering only practiced in abbreviated session	Requirement Engineering;	Software development	2
[77]	Requirement documents based on marketing requirement produced iteratively	Update requirement document quickly	Software development	2
[82]	When user had request the prioritization was conflicting between designers and developers	Balance conflicts in prioritization between usability and product team	Software development	1
[77]	Fixing architectural issues	Specialists for architecture	Software development	2
[72]	Refactoring was difficult, since developers had no experience	Refactoring	Software development	1
[85]	Using standard APIs makes switching to another project is an advantage if switching to another project	Standard APIs	Software development	1
[70]	Use third party but offline components than online third party services	Third party offline components	Software development	2
[70]	Reusing cots reinforce the architectural structure of the product	Use of cots	Software development	2
[68]		Automated acceptance test	Software development	0
[68]		Unit tests	Software development	0
[82]	Usability test on proxy instead of real users.	Proxy users	Software development	1
[78]	To risk management for making decisions. Include test planning process, test case development, documentation of testing and reporting.	Traditional testing for making decisions	Software development	1
[82]	Usability test is important for product/market fit	Usability test most important to achieve product/market fit	Software development	1
[66]	1 customer inside the company writing user stories during stand-up meetings	Customer-on site	Software development	1
[66]		Manual deploy	Software development	1
[66]		Pair programming	Software development	1
[66]		Planning game	Software development	1
[66]	Developers didn't see immediate value in refactoring in using abstract concepts	Refactoring	Software development	1
[66]		Simple design	Software development	1
[66]	Was hard to implement, lack of supporting technologies, lack of experience made TDD hard to implement	TDD	Software development	1
[66]	Problems with CI because of low experience with version control system.	Use tools for Continuous Integration	Software development	1
[66]		Code standards	Software development	1
[65]	QA and testing largely absent to reduce development lead time	Quality Assurance	Software development	1
[66]		Cutting-edge technologies;	Tools and Technology	1
[4]	Choose tools which do not conflict with strategic plan	Right tools	Tools and Technology	1
[2]	Use technology that allows to change quickly	Adopt technologies easy to adapt;	Tools and Technology	1
[77]	Official documentation was HTML base for portability and accessibility	HTML documentation	Tools and Technology	2
[85]	Save costs	Open source databases	Tools and Technology	1
[85]	Open source ensures more compatibility with other components	Open source technologies	Tools and Technology	1
[74]	When a company wants to test the viability of a new technology, requiring a large pool of tester, rapidly evolving academic research, cheaply expand market, faces large number of evaluators and few buyers, Open Source can help overcoming this difficulty	Release Open Source code	Tools and Technologies	1
[74]	Use open source COTS instead of closed source components to have access to the code	Use Open Source components	Tools and Technologies	1
[87]	Using online wiki tool for synchronize the docs and accessability is a good practice. It is also easy to implement.	Use of wiki	Tools and Technologies	1
[77]	Support models and diagrams on the whiteboard. Good because requires little training and enforce teamness	Use of whiteboards	Tools and Technologies	2
[77]	When they needed permanent documentation they simply scanned sketch drawings	Use simple tools	Tools and Technologies	2

Table .16: Work practices

Version two

References	One-line comment	Work practice	m	Category provided	Evidence
[68]	Helps find actionable issues	Root cause analysis	Software development	Analysis	0
[72]	Actively push developers during planning game to participate	Planning game	Software development	Analysis	1
[71]	Value engineering balanced features against time and budget	Planning game	Software development	Analysis	1
[64]	Generate alternatives solutions and evaluate them	Analysis	Software development	Analysis	1
[78]	Looking at the basic requirement process and provide raccomendations where the risk exists and what to be done for mitigating. Assessment of the scalability risk for basic architecture and design is very important for internet startups.	Initial risk assessment	Software development	Analysis	0
[65]	No evidences of risk analysis conducted	No risk analysis	Software development	Analysis	1
[65]	Formulating trade-off at the end of the development process in order to speed-up time to market	Trade-off analysis only in retrospective	Software development	Analysis	1
[68]		Minimum marketable features (MMF)	Software development	Analysis	0
[78]	Expecially scalability, robustness, speed of changing in technology	Understing key failure conditions	Software development	Analysis	1
[82]	But it didn't allow a proper feasibility study	Use unpredictability	Software development	Analysis	1
[67]	Postpone technological choices that are restrictive and can limit flexibility	Post-pone limiting decisions;	Software development	Analysis	1

Table .17 – *Continued on next page*

References	One-line comment	Work practice	m	Category provided	Evidence
[68]		Prioritization with time boxing. WIP is 2 at the time	Software development	Analysis	0
[82]	When user had request the prioritization was conflicting between designers and developers	Balance conflicts in prioritization between usability and product team	Software development	Analysis	1
[66]		Planning game	Software development	Analysis	1
[69]		Release system	Software development	Deployment	2
[66]		Manual deploy	Software development	Deployment	1
[68]		Simple design	Software development	Design	0
[71]	Working softer was favoured	Simple design	Software development	Design	1
[72]	Developing an architecture is difficult	Lack of architecture	Software development	Design	1
[70]	Startups take advantage of existing components saving time-to-market. Extending software functionalities from a third party suppliers	Code reuse	Software development	Design	2
[77]	Define an infrastructure	Enterprise architectural modelling process	Software development	Design	2
[77]	Only overview of the architecture. No details	Only high-level architecture.	Software development	Design	2
[65]	Family of products built close together enhancing re-use	Re-use	Software development	Design	1
[77]	Because of growth	Refactor architecture	Software development	Design	2
[70]	SOA saved time and resources using third party services. But changing a component introduce substantial rework.	SOA using third party services	Software development	Design	2
[77]	Benefits in having time of discussing and reflect with co-workers	Architecture conducted in parallel with development	Software development	Design	2
[86]	The system should scale with the company size in harmony, early but not late	Prepare for scalability	Software development	Design	2
[2]	Developers should maximize the amount of code re-used that they are familiar with	Code re-use;	Software development	Design	1
[77]	Fixing architectural issues	Specialists for architecture	Software development	Design	2
[85]	Using standard APIs makes switching to another project is an advantage if switching to another project	Standard APIs	Software development	Design	1
[70]	Use third party but offline components than online third party services	Third party offline components	Software development	Design	2
[70]	Reusing cots reinforce the architectural structure of the product	Use of cots	Software development	Design	2
[66]		Simple design	Software development	Design	1
[71]	Coding standards initially avoided but later implemented	Coding standards	Software development	Implemantation	1
[71]	Programming were independent most of the time	Pair-Programming	Software development	Implementation	1
[69]	Viewed as obsolete and irrelevant	Code metrics	Software development	Implementation	2
[65]	Object orient development orientation for package firms	Object oriented development	Software development	Implementation	1
[66]		Pair programming	Software development	Implementation	1
[66]		Code standards	Software development	Implementation	1
[69]	Documentation is not viable in the short term but essential for the long run	Documentation	Managerial/organizational	M Documentation	2
[33]	Necessity of skipping documentation, lead to excess of tacit knowledge	Documentation	Managerial/organizational	M Documentation	2
[33]	The size of the company influences the attitude towards documentation: the smaller the company the greater the hostility towards documentation	Documentation	Managerial/organizational	M Documentation	2
[17]	Necessity of skipping documentation, lead to excess of tacit knowledge	Documentation	Managerial/organizational	M Documentation	2
[68]		Code refactoring	Software development	Maintainance	0
[71]	Modules were constantly improved, periodic burst and dramatic improvement occurred.	Refactoring	Software development	Maintainance	1
[77]	Because of growth	Redevelopment of the system	Software development	Maintainance	2
[69]	Ensure readability and understandability of the code	Refactor system	Software development	Maintainance	2
[69]	Developers fix a defect described in an open issue report. Or improve quality	Rework code	Software development	Maintainance	2
[72]	Refactoring was difficult, since developers had no experience	Refactoring	Software development	Maintanance	1
[66]	Developers didn't see immediate value in refactoring in using abstract concepts	Refactoring	Software development	Maintenance	1
[9]	Poor documentation (no process documentation) led to poor control over the change request, that are postponed to next releases	Documentation	Process management	PM Documentation	2
[68]		Integration test with CI	Software development	QA	0
[68]		Peer-review critical paths	Software development	QA	0
[71]	Code was rarely broken and was continuously compiled	Continuous integration	Software development	QA	1
[71]	Testing was continuous but without advanced scripts	Continuous testing	Software development	QA	1
[9]		Acceptance testing	Software development	QA	2
[69]	Customers use scenarios to judge whether the systems needs anticipated tests	Acceptance tests	Software development	QA	2
[69]	Include code for unique test cases and write necessary test files for documentation to improve traceability.	Realize scenarios with TDD	Software development	QA	2
[9]	Contributed to improve faulty releases.	System testing	Software development	QA	2
[64]	high-quality product	Quality product	Software development	QA	1
[78]	Quality assurance team are not formed until the product is not ready to be delivered. But at that time the practices are already established and difficult to change. QA is tipically inexperienced, lack of SE training and no business experience.	Elevate the software assurance function to report directly to the CEO	Software development	QA	1
[86]	User adopters as quality assurance team	Involve users for QA	Software development	QA	2
[78]	Increased quality from outsourcing	Outsourcing QA	Software development	QA	0
[69]	Time spent on QA is reduced because of time-to-market pressure	QA	Software development	QA	2
[68]		Automated acceptance test	Software development	QA	0
[68]		Unit tests	Software development	QA	0
[82]	Usability test on proxy instead of real users.	Proxy users	Software development	QA	1

Table .17 – *Continued on next page*

References	One-line comment	Work practice	m	Category provided	Evidence
[78]	To risk management for making decisions. Include test planning process, test case development, documentation of testing and reporting.	Traditional testing for making decisions	Software development	QA	1
[82]	Usability test is important for product/market fit	Usability test most important to achieve product/market fit	Software development	QA	1
[66]	Was hard to implement, lack of supporting technologies, lack of experience made TDD hard to implement	TDD	Software development	QA	1
[66]	Problems with CI because of low experience with version control system.	Use tools for Continuous Integration	Software development	QA	1
[65]	QA and testing largely absent to reduce development lead time	Quality Assurance	Software development	QA	1
[68]	Those that are related to users.	Trace metrics of product usage	Software development	RE	0
[71]	CEO and analytic director acted as customers	On-site customer	Software development	RE	1
[4]	Stay close to customers during product development	Proximity to customer	Software development	RE	1
[4]	Startups should capture new requirements, prioritize them, capture feasibility and value!	Requirement Engineering;	Software development	RE	1
[67]	Feedback driven patterns to have an high interaction between internal and external factors. Customer provides key to action	Feedback-driven RE	Software development	RE	1
[64]	Adjust requirements collecting customer feedback	Customer development	Software development	RE	1
[80]	Co-create value with the customer	Costumer involvment	Software development	RE	0
[68]		Customer development	Software development	RE	0
[22]	Not clear requirements. The design team must adapt to this frequent and changing	Changing requirements	Software development	RE	0
[78]	Introduce a process to understand the costumer quality needs	Costumer feedbacks	Software development	RE	1
[82]	Go to customers to learn their needs. It was a good approach to enhance usability of the product.	Customer involvement	Software development	RE	1
[85]	To rank the companies' priorities around the customers' needs	Customer Involvement	Software development	RE	2
[80]	Monitor and collect feedbacks	Customers feedback	Software development	RE	0
[80]	Define what product or service you are going to provide	Value proposition	Software development	RE	0
[80]	Differentiate your product from customers	Value proposition	Software development	RE	0
[80]	Indentify costomer target are	Value proposition	Software development	RE	0
[69]	Requirement engineering only practiced in abbreviated session	Requirement Engineering;	Software development	RE	2
[77]	Requirement documents based on marketing requirement produced iteratively	Update requirement document quickly	Software development	RE	2
[66]	1 customer inside the company writing user stories during stand-up meetings	Customer-on site	Software development	RE	1
[69]	Customer and developers collect user stories effort is estimated.	Collect scenarios	Software development	RE and Analysis	2
[68]		Document new features if applicable, especially for interaction with third party	Software development	SD Documentation	0
[79]		Documentation is approached at the end of the user stories, often at the end of the sprint.	Software development	SD Documentation	0
[65]	Software architecture not tangible (lack of documentation of modeling)	Documentation	Software development	SD Documentation	1
[69]	Documentation specialist writes an online help and user guide book	Create user doc	Managerial/organizational	Usability	2
[85]	The open source community helped in support product development with no costs	Rely on open source community support	Managerial/organizational		1
[68]	To find subjects for improvements	Measure lead time and cycle time	Managerial/organizational		0
[71]	40-h workweek (be flexible but regular workdays)	40-h work week	Managerial/organizational		1
[71]	Code was officially shared but developers exhibited possessiveness	Collective Ownership	Managerial/organizational		1
[71]	Communication simple, formal but ambiguous	System metaphor	Managerial/organizational		1
[23]	High communication between development and marketing team to facilitate spotting product/market fit.	Communication marketing-engineering;	Managerial/organizational		2
[77]	Collective ownerships under SVN of the architectural docs	Collective Ownership of architecture	Managerial/organizational		2
[78]	Costumer in the internet are not tollerant, and competitve solutions are just one click away	Plan for robustness	Managerial/organizational		1
[9]	Plan to release the product	Release deployment	Managerial/organizational		2
[9]	Assignation of tasks to developers	Release development	Managerial/organizational		2
[9]	Start with the release scope preparation defining functionalities to include. Delay has been decreased	Release management process	Managerial/organizational		2
[9]	prioritization of requirements, release cycle has been improved by planning	Release planning	Managerial/organizational		2
[77]	Chef-architect makes models visible to every developers and collect feed-backs	Visibility of the architectures	Managerial/organizational		2
[69]	Everyone collect metrics (testing, progress, velocity, quality)	Collect metrics	Managerial/organizational		2
[77]	Because of growth	Hiring new developers	Managerial/organizational		2
[69]	Project manager bases his decisions (effort estimations, schedules and priorities) on collected metrics	Project manager takes decisions for PM	Managerial/organizational		2
[73]	Engineering students developed a technology in parallel with the market team, and frequently independently. This was an issue that limited the project.	Separate development from market concerns	Managerial/organizational		1
[83]	Lead time as the main driver for product development	Time-to-market as driver for the development	Managerial/organizational		2
[22]	Early stage startups demand initial more working hours, more stress and responsabilities in comparison to established companies.	Working over time	Managerial/organizational		0
[65]	little project management	No project management	Managerial/organizational		1
[65]	No use of traditional project planning and poorly planned development		Managerial/organizational		1
[2]	Empower team improve quality of product, adaptability of process,efficiency and coordination.	Empower team;	Managerial/organizational		1

Table .17 – *Continued on next page*

References	One-line comment	Work practice	m	Category provided	Evidence
[4]	focus on single product, create a new version if need customizations.	Focus on single-product	Managerial/organizational		1
[4]	Plan objectives short-medium term	Plan objectives	Managerial/organizational		1
[76]	Monitor the process continuously to verify the alignment with the business strategy	Monitor process;	Managerial/organizational		1
[75]	Select the right skills in people	Skill team;	Managerial/organizational		0
[23]	Core Team expertise, diversified knowledge and harmony are essential for success. Many angels and VCs highlighted the assessment of the core team in investment decision making. Very often start-ups are founded by young people who themselves lack management skills and experience but do not hire suitable managers. This creates difficulties in both R&D and in the marketing processes. At certain stages, where the start-up lacks expertise, consultants can be useful.	Team skills vital to success	Managerial/organizational		2
[33]	Co-locate the development team to reduce the costs increasing the sharing of the tacit knowledge	Co-locate the development team	Managerial/organizational		2
[77]	Modelling and documentation were separated efforts.	Dedicated people to docs and modelling	Managerial/organizational		2
[77]	Sharing knowledge and collaboration	Developers and testers were close	Managerial/organizational		2
[77]	When scalability problems arise	Hire chief-architect	Managerial/organizational		2
[86]	Hire developers with self starter personalities with past experience in similar domain	Need of experience in similar domain and self starter personalities	Managerial/organizational		2
[86]	Youthness of the founders contribute to failure rates. Startups need to hire technical and expertise	Need of expertise	Managerial/organizational		2
[22]		No separation between designers and developers	Managerial/organizational		0
[62]	The competence of founders working with innovative products are different with non innovative	Prefer IT experts in the founding team	Managerial/organizational		2
[22]	People can and encouraged to work at home but with more control. Due to the large amount of work.	Work from home with flexible hours.	Managerial/organizational		0
[67]	Team capability should be able to absorb and learn from trial and error quickly to avoid prolonged time	Learn quickly from trial and error	Managerial/organizational		1
[7]	Empower team to Enhance creativity and flexibility	Empower team	Managerial/organizational		2
[17]	Empower team to Enhance creativity and flexibility	Empower team	Managerial/organizational		2
[2]	Skilled team, because developer should be able to adapt to new roles and face new challenges everyday	Skilled team;	Managerial/organizational		1
[65]	Importance of the core team: highly motivated, homogeneous, history of working together, small, loose organizational structure	Get the right team	Managerial/organizational		1
[65]	working extra hours	Work overtime	Managerial/organizational		1
[23]	Characteristic of the entrepreneur plays a primary role in maximising the scarce resources	Entrepreneur skills primary;	Managerial/organizational		2
[24]	Characteristic of the entrepreneur (Courage , enthusiasm, commitment, leadership) are more important than other technical factor for the success of a startup	Entrepreneur skills primary;	Managerial/organizational		2
[7]	Hire expertise with agile background to improve chances of success	Agile background	Managerial/organizational		2
[4]	Developers are inexperienced and they need to familiarize with software engineering practices to deal with multi-role full-stack challenges	Familiarize with all SE practices	Managerial/organizational		1
[33]	Hire expertise to supply lack of knowledge in critical moments	Hiring expertise	Managerial/organizational		2
[17]	Hire expertise to supply lack of knowledge in critical moments	Hiring expertise	Managerial/organizational		2
[4]	Promote developers showing both leadership and technical capacity	Endorse leadership AND technical capacity	Managerial/organizational		1
[86]	Makes specifications free available to everybody	Use of open standard	Managerial/organizational		2
[72]	high experience developers hard to buy-in XP practices		Managerial/organizational		1
[66]	The work week was not fully respected	40h work week	Managerial/organizational		1
[66]	Senior developers were more comfortable with this practice	Collective code ownership	Managerial/organizational		1
[66]	To solve conflicts among developers or with customers	Dirty laundry meeting	Managerial/organizational		
[66]		Retrospectives	Managerial/organizational		1
[68]	The Agile process is predictable with acceptable variance, and change direction of the business of any given time	Agile	Process management		0
[68]	The lead time from idea to production is 8 days. Deployment every day	Fast release with continous deployment	Process management		0
[7]	Startups should tailor agile processes and practices since the cost of SPI (time and res.) is too high to afford	Agile process tailoring	Process management		2
[17]	Startups should tailor agile processes and practices since the cost of SPI (time and res.) is too high to afford	Agile process tailoring	Process management		2
[71]	weekly delivered, frequent build cycles	Small releases	Process management		1
[72]	Trying to tailor XP harms the effectiveness	(try) to fully follow XP	Process management		1
[64]	Just-in-time delivery practices, prototyping	Release fast	Process management		0
[79]		Definition of done	Process management		0
[69]		Plan iterations based on scenarios	Process management		2
[65]	Mock-up, prototypes, demo to customers in order to test functionalities	Prototype	Process management		1
[68]		Definition of done	Process management		0
[77]	6 to 12 weeks	Incremental release in short cycle	Process management		2
[68]	Product queue for listing the features, from 1 to 5	Kanban product queue	Process management		0
[79]		Kanban wall	Process management		0
[77]	Tailoring rup to meet specific necessity	RUP	Process management		2
[79]	2 weeks	Sprint base approach	Process management		0

Table .17 – *Continued on next page*

References	One-line comment	Work practice	m	Category provided	Evidence
[68]	Maintain the buffer small enough	The number of features up to seven	Process management		0
[79]		WIP metric, 3 tasks at the time	Process management		0
[77]	6 to 12 weeks	Highly iterative process	Process management		2
[88]	Process assessment is considered wasteful	Process assessment	Process management		0
[69]	Viewed as obsolete and irrelevant	Process improvement	Process management		2
[33]	Tailor process to startups' context	Process tailoring	Process management		2
[7]	Tailor process to startups' context	Process tailoring	Process management		2
[33]	XP is the most used across organizations because of the least process cost and low level of documentation requirements	Use XP	Process management		2
[69]	Tailoring of XP but pair programming but with documenting practices	XP practices	Process management		2
[2]	Process definition only high level: leave details at low level to the team.	High-level work-flow;	Process management		1
[17]	Minimum processes might work	Minimum process	Process management		2
[64]	Evolutionary prototyping with improvements	Prototyping	Process management		1
[64]	Rapid development processes	Rapid development;	Process management		1
[2]	Evolutionary work-flows at high level to allow prototype;	Evolutionary work-flows;	Process management		1
[4]	Start introducing repeatable processes when grow starts to parallelism	Introduce repeatable process in growth	Process management		1
[79]	The company was applying scrumban. Scrumban proved to be a great organizational and motivational tool, but working well only if every one is doing it. Scrum emphises the customers and their needs.	Scrumban	Process management		0
[65]	Little effort allocated formalizing specifications, design, documentation and testing	Minimal engineering activities	Process management		1
[19]	Start with a product centric approach (ad-hoc) to avoid uncertainty of team market and platform, but at certain point the company should be ready to implement some process and scale	From product to process-centric approach;	Process management		1
[19]	A process is necessary to scale the company activities, and the startup should monitor the state of evolution of process vs. product to be able to take decision on when to introduce some process	Measure process maturity;	Process management		1
[65]	Engineering;marketing in parallel	Overlapping phases	Process management		1
[4]	Begin SPI when market shares and customers are established (mature)	Begin SPI after market stabilized	Process management		1
[24]	After building a first functioning prototype the team should focus on marketing instead of technical.	Prototype and market first;	Process management		2
[66]		Full XP;	Process management		1
[66]	One release every two weeks with one week of bug fixing	Small-release	Process management		1
[66]	used XP help trying to reduce time-to-market	XP practices	Process management		1
[66]	Even trying to implement XP, developers were more like cowboy coders.	Cow-boy coding;	Process management		1
[64]	Solutions should remain "soft-coded" in the first phases until the optimal solution is found	Prototyping	Process management		1
[66]		Cutting-edge technologies;	Tools and Technology		1
[4]	Choose tools which do not conflict with strategic plan	Right tools	Tools and Technology		1
[2]	Use technology that allows to change quickly	Adopt technologies easy to adapt;	Tools and Technology		1
[77]	Official documentation was HTML base for portability and accessibility	HTML documentation	Tools and Technology		2
[85]	Save costs	Open source databases	Tools and Technology		1
[85]	Open source ensures more compatibility with other components	Open source technologies	Tools and Technology		1
[74]	When a company wants to test the viability of a new technology, requiring a large pool of tester, rapidly evolving academic research, cheaply expand market, faces large number of evaluators and few buyers, Open Source can help overcoming this difficulty	Release Open Source code	Tools and Technologies		1
[74]	Use open source COTS instead of closed source components to have access to the code	Use Open Source components	Tools and Technologies		1
[87]	Using online wiki tool for synchronize the docs and accessability is a good practice. It is also easy to implement.	Use of wiki	Tools and Technologies		1
[77]	Support models and diagrams on the whiteboard. Good because requires little training and enforce teamness	Use of whiteboards	Tools and Technologies		2
[77]	When they needed permanent documentation they simply scanned sketch drawings	Use simple tools	Tools and Technologies		2

Table .17: Work practices

References

[1] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering – A systematic literature review, Information and Software Technology 51 (2009) 7–15.

[2] S. M. Sutton, The role of process in software start-up, IEEE Software 17 (2000) 33–39.

[3] T. Kane, The Importance of Startups in Job Creation and Job Destruction, Technical Report, Kauffman Foundation, 2010.

[4] M. Crowne, Why software product startups fail and what to do about it, in: Proceedings of 2002 IEEE International Engineering, pp. 338–343.

[5] A. Maccormack, How Internet Companies Build Software, MIT Sloan Management Review (2001) 75–84.

[6] K. M. Eisenhardt, S. L. Brown, Time pacing: competing in markets that won't stand still., Harvard Business Review 76 (1998) 59–69.

[7] G. Coleman, R. V. O'Connor, An investigation into software development process formation in software start-ups, Journal of Enterprise Information Management 21 (2008) 633–648.

[8] S. Blank, The four steps to the epiphany, Cafepress. com, 2005.

[9] M. Kajko-Mattsson, N. Nikitina, From Knowing Nothing to Knowing a Little: Experiences Gained from Process Improvement in a Start-Up Company, in: 2008 International Conference on Computer Science and Software Engineering, IEEE, 2008, pp. 617–621.

[10] G. Coleman, An empirical study of software process in practice, in: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, volume 00, p. 315c.

[11] D. Storey, Entrepreneurship and the New Firm, Croom Helm, 1982.

[12] A. B. Perkins, M. C. Perkins, The Internet Bubble: Inside the Overvalued World of High-Tech Stocks – And What You Need to Know to Avoid the Coming Catastrophe, HarperInformation, 1999.

[13] M. Marmer, B. L. Herrmann, E. Dogrultan, R. Berman, C. Eesley, S. Blank, Startup Genome Report Extra: Premature Scaling, Technical Report, Startup Genome, 2011.

[14] R. Grimaldi, A. Grandi, Business incubators and new venture creation: an assessment of incubating models, Technovation 25 (2005) 111–121.

[15] C. M. Christensen, The Innovator's Dilemma, Harvard Business School Press, 1997.

[16] G. Chroust, What is a software process?, Journal of Systems Architecture 42 (1996) 591–600.

[17] G. Coleman, R. Oconnor, Investigating software process in practice: A grounded theory perspective, Journal of Systems and Software 81 (2008) 772–784.

[18] J. Bach, Microdynamics of process evolution, Computer 31 (1998) 111–113.

[19] I. Heitlager, R. Helms, S. Brinkkemper, A tentative technique for the study and planning of co-evolution in product, in: Software Evolvability, 2007 Third International IEEE Workshop on, pp. 42 –47.

[20] K. Martin, B. Hoffman, An open source approach to developing software in a small organization, Software, IEEE 24 (2007) 46 –53.

[21] A. Cockburn, Surviving Object-Oriented Projects, Addison-Wesley Professional, 1998.

[22] M. Tanabian, Building high-performance team through effective job design for an early stage software start-up, in: Management Conference, pp. 789–792.

[23] S. Chorev, A. R. Anderson, Success in Israeli high-tech start-ups; Critical factors and process, Technovation 26 (2006) 162–174.

[24] M. Kakati, Success criteria in high-tech new ventures, Technovation 23 (2003) 447–457.

[25] M. Marmer, B. L. Herrmann, E. Dogrultan, R. Berman, C. Eesley, S. Blank, The Startup Ecosystem Report 2012, Technical Report, Startup Genome, 2012.

[26] C. Alves, S. Pereira, J. Castro, A Study in Market-Driven Requirements Engineering, Technical Report, Universidade Federal de Pernambuco, 2006.

[27] O. D. Johan Natt, Elicitation and management of user requirements in market-driven software development, Ph.D. thesis, Department of Communication Systems Lund Institute of Technology, 2002.

[28] P. Sawyer, I. Sommerville, G. Kotonya, Improving market-driven re processes, in: International Conference on Product-Focused Software Process Improvement (Profes '99).

[29] C. Potts, Invented requirements and imagined customers: requirements engineering for off-the-shelf software, in: Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on, pp. 128 – 130.

[30] L. Karlsson, Å. G. Dahlstedt, J. N. O. Dag, B. Regnell, A. Persson, Challenges in market-driven requirements engineering - an industrial interview study, in: Proceedings of the Eighth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02.

[31] D. A., Study of current practices in marketdriven requirements engineering, in: Third Conference for the Promotion of Research in IT, University Colleges Sweden.

[32] M. Keil, E. Carmel, Customer-developer links in software development, Commun. ACM 38 (1995) 33–44.

[33] G. Coleman, R. O'Connor, Using grounded theory to understand software process improvement: A study of Irish software product companies, Information and Software Technology 49 (2007) 654–667.

[34] W. Cunningham. The WyCash Portfolio Management System. [Online]. Available: `http://c2.com/doc/oopsla92.html` (Accessed : Sep. 15, 2012)., ????

[35] A. Nugroho, J. Visser, T. Kuipers, An empirical model of technical debt and interest, in: Proceedings of the 2nd Workshop on Managing Technical Debt, MTD '11, ACM, New York, NY, USA, 2011, pp. 1–8.

[36] C. Izurieta, A. Vetrò, N. Zazworka, Organizing the technical debt landscape, in: Workshop on Managing Technical Debt (MTD), 2012 Third International, pp. 23–26.

[37] Third international workshop on Managing Technical Debt. [Online]. Available: `http://www.sei.cmu.edu/community/td2012/` (Accessed : Sep. 15, 2012)., ????

[38] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. Ozkaya, R. Sangwan, C. Seaman, K. Sullivan, N. Zazworka, Managing technical debt in software-reliant systems, in: Proceedings of the FSE/SDP workshop on Future of software engineering research, New York, NY, USA, pp. 47–52.

[39] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic Mapping Studies in Software Engineering, in: 12th International Conference on Evaluation and Assessment in Software Engineering, volume 17, pp. 1–10.

[40] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[41] T. Dybå, V. B. Kampenes, D. I. Sjø berg, A systematic review of statistical power in software engineering experiments, Information and Software Technology 48 (2006) 745–755.

[42] D. Budgen, M. Turner, P. Brereton, B. Kitchenham, Using Mapping Studies in Software Engineering, in: Proceedings of PPIG 2008, Lancaster University, 2008, pp. 195–204.

[43] S. Rumsey, How to find information : a guide for researchers, McGraw-Hill, 2008.

[44] Engineering Village. [Online]. Available: `http://www.engineeringvillage2.org/` (Accessed : Aug. 25, 2012)., ????

[45] ACM Search. [Online]. Available: `http://dl.acm.org/advsearch.cfm` (Accessed : Aug. 25, 2012)., ????

[46] IEEExplore. [Online]. Available: `http://ieeexplore.ieee.org/` (Accessed : Aug. 25, 2012)., ????

[47] Web of Knowledge. [Online]. Available: `http://wokinfo.com/` (Accessed : Aug. 25, 2012)., ????

[48] Scopus. [Online]. Available: `http://www.scopus.com/` (Accessed : Aug. 25, 2012)., ????

[49] Google Scholar. [Online]. Available: `http://scholar.google.com/` (Accessed : Aug. 25, 2012)., ????

[50] BibDesk. [Online]. Available: `http://bibdesk.sourceforge.net/` (Accessed : Aug. 25, 2012)., ????

[51] T. Dybå, T. Dingsø yr, Empirical studies of agile software development: A systematic review, Information and Software Technology 50 (2008) 833–859.

[52] M. Jorgensen, M. Shepperd, A systematic review of software development cost estimation studies, Software Engineering, IEEE Transactions on 33 (2007) 33 –53.

[53] A. Sayers, Tips and tricks in performing a systematic review, Br J Gen Practice 1 (2007) 542–759.

[54] M. Ivarsson, T. Gorschek, A method for evaluating rigor and industrial

relevance of technology evaluations, Empirical Software Engineering 16 (2010) 365–395.
[55] M. Ivarsson, T. Gorschek, Technology transfer decision support in requirements engineering research: a systematic review of rej, Requir. Eng. 14 (2009) 155–175.
[56] M. Unterkalmsteiner, T. Gorschek, A. Islam, C. K. Cheng, R. Permadi, R. Feldt, Evaluation and measurement of software process improvement – a systematic literature review, Software Engineering, IEEE Transactions on 38 (2012) 398 –424.
[57] T. Saracevic, Evaluation of evaluation in information retrieval, in: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '95, ACM, New York, NY, USA, 1995, pp. 138–146.
[58] M. Wood, J. Daly, J. Miller, M. Roper, Multi-method research: an empirical investigation of object-oriented technology, Journal of Systems and Software 48 (1999) 13–26.
[59] M. Kasunic, Designing an Effective Survey, Technical Report, Carnegie Mellon Software Engineering Institute, 2005.
[60] R. Wieringa, N. Maiden, N. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, Requir. Eng. 11 (2005) 102–107.
[61] M. Shaw, Writing good software engineering research papers, in: International Conference on Software Engineering, Proceedings. 25th, pp. 726 – 736.
[62] M. Häsel, N. Breugst, T. Kollmann, IT Competence in Internet Founder Teams An Analysis of Preferences and Product Innovativity, Business & Information System Engineering 52 (2010) 210–217.
[63] R. Hanna, T. U. Daim, Information technology acquisition in the service sector, International Journal of Services Sciences 3 (2010) 21.
[64] E. Deakins, S. Dillon, A helical model for managing innovative product and service initiatives in volatile commercial environments, International Journal of Project Management 23 (2005) 65–74.
[65] E. Carmel, Time-to-completion in software package startups, Proceedings of the System Sciences, 1994. (1994) 498–507.
[66] A. da Silva, F. Kon, Xp south of the equator: An experience implementing xp in brazil, Extreme Programming and Agile Processes (2005) 10–18.
[67] C. Midler, P. Silberzahn, Managing robust development process for high-tech startups through multi-project learning: The case of two European start-ups, International Journal of Project Management 26 (2008) 479–486.
[68] M. Taipale, Huitale - A story of a Finnish lean startup, in: Proceedings of the First International Conference, volume 65 LNBIP, Helsinki, Finland, pp. 111–114.
[69] J. Zettel, F. Maurer, J. Münch, L. Wong, LIPE: a lightweight process for e-business startup companies based on extreme programming, Product Focused Software Process Improvement (2001) 255–270.
[70] S. Jansen, S. Brinkkemper, I. Hunink, Pragmatic and Opportunistic Reuse in Innovative Start-up Companies, Software, IEEE (2008) 42–49.
[71] P. Tingling, Extreme programming in action: a longitudinal case study, Proceedings of the 12th international conference (2007) 242–251.
[72] R. Deias, G. Mugheddu, Introducing XP in a start-up, European Internet Services Company (2002).
[73] R. Stanfill, T. Astleford, Improving Entrepreneurship Team Performance through Market Feasibility Analysis, Early Identification of Technical Requirements, and Intellectual Property Support, in: American Society for Engineering Education Annual Conference.
[74] D. Wood, Open Source Software Strategies for Venture-Funded Startups, Technical Report TR-MS1287, MIND Laboratory, University of Maryland, 2005.
[75] H.-J. Steenhuis, E. de Bruijn, Innovation and technology based economic development: Are there short-cuts?, in: Management of Innovation and Technology, 2008. ICMIT 2008. 4th IEEE International Conference on, pp. 837 –841.
[76] S. Yogendra, Aligning business and technology strategies: a comparison of established and start-up business contexts, in: Management Conference.
[77] S. Ambler, Lessons in agility from Internet-based development, IEEE Software (2002) 66–73.
[78] J. Mater, B. Subramanian, Solving the software quality management problem in Internet startups, in: Proceedings of the 18th annual pacific northwest software quality conference, pp. 297–306.
[79] K. Kuvinka, Scrum and the Single Writer, in: Proceedings of Technical Communication Summit, May, pp. 18–19.
[80] S.-C. Li, The role of value proposition and value co-production in new internet startups: How new venture e-businesses achieve competitive advantage, in: Management of Engineering and Technology, Portland International Center for, pp. 1126 –1132.
[81] S.-l. Lai, Chinese Entrepreneurship in the Internet Age : Lessons from Alibaba.com, World Academy of Science, Engineering and Technology 72 (2010) 405–411.
[82] B. Mirel, Product, process, and profit: the politics of usability in a software venture, ACM Journal of Computer Documentation (JCD) 24 (2000) 185–203.
[83] O.-P. Hilmola, P. Helo, L. Ojala, The value of product development lead time in software startup, System Dynamics Review 19 (2003) 75–82.
[84] E. Kim, S. Tadisina, Factors impacting customers' initial trust in e-businesses: an empirical study, in: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, volume 07, pp. 1–10.
[85] D. Wall, Using open source for a profitable startup, Computer 34 (2001) 158 –160.
[86] D. Yoffie, Building a company on Internet time: Lessons from netscape, California Management Review 4 (1999).
[87] L. Bean, D. D. Hott, Wiki: A speedy new tool to manage projects, Journal of Corporate Accounting & Finance 16 (2005) 3–8.
[88] M. Fayad, Process assessment considered wasteful, Communications of the ACM 40 (1997) 125–128.
[89] D. Šmite, C. Wohlin, T. Gorschek, R. Feldt, Empirical evidence in global software engineering: a systematic review, Empirical Software Engineering 15 (2009) 91–118.
[90] N. Ali, H. Edison, Towards innovation measurement in software industry, Master's thesis, Blekinge Institute of Technology, 2010.
[91] S. Blank, B. Dorf, The Startup Owner's Manual: The Step-by-Step Guide for Building a Great Company, K&S Ranch Publishing LLC, 2012.
[92] E. Ries, The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Business, 2011.
[93] Colin Robson, Real World Research: A Resource for Social Scientists and Practitioner-Researchers, John Wiley and Sons, 2009.
[94] J. G. Brodman, D. L. Johnson, What small business and small organizations say about the cmm: experience report, in: Proceedings of the 16th international conference on Software engineering, ICSE, IEEE Computer Society Press, Los Alamitos, CA, USA, 1994, pp. 331–340.
[95] T. Dyba, B. Kitchenham, M. Jorgensen, Evidence-based software engineering for practitioners, Software, IEEE 22 (2005) 58 – 65.
[96] B. Kitchenham, T. Dyba, M. Jorgensen, Evidence-based software engineering, in: Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on, pp. 273 – 281.
[97] C. von Wangenheim, A. Anacleto, C. Salviano, Helping small companies assess software processes, Software, IEEE 23 (2006) 91 –98.
[98] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering: an introduction, Kluwer Academic Publishers, Norwell, MA, USA, 2000.
[99] G. Coleman, eXtreme Programming (XP) as a "Minimum" Software Process : A grounded theory, in: Proceedings of the 28th Annual International Computer Software and Applications Conference, pp. 4–5.
[100] G. Coleman, Practice Not Process : Improving the Capability of Software Startups, 2004. Proposal at Department of Computing and Mathematics, Duldalk Institute of Technology.
[101] M. Lehman, Programs, life cycles, and laws of software evolution, in: Proceedings of the IEEE, volume 68, pp. 1060 – 1076.
[102] R. Banker, G. Davis, Software development practices, software complexity, and software maintenance performance: A field study, Management Science (1998).
[103] P. Thiel. Notes on CS183. [Online]. Available: `http://blakemasters.tumblr.com/post/21742864570/peter-thiels-cs183-startup-class-6-notes-essay` (Accessed : Aug. 25, 2012)., ????
[104] K. Beck, C. Andres, Extreme Programming Explained: Embrace

Change (2nd Edition), Addison-Wesley Professional, 2004.
[105] N. Gautam, N. Singh, Lean product development: Maximizing the customer perceived value through design change (redesign), International Journal of Production Economics 114 (2008) 313–332.
[106] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, Agile software development methods, Relatório Técnico, Finlândia (2002).
[107] Rob Walling, Start Small, Stay Small: A developer's Guide to Launching a Startup, The Numa Group, 2010.
[108] M. Marmer, B. L. Herrmann, E. Dogrultan, R. Berman, C. Eesley, S. Blank, Startup Genome Report Extra: Premature Scaling, Technical Report, Startup Genome, 2011.
[109] C. F. Kurtz, D. J. Snowden, The new dynamics of strategy: Sense-making in a complex and complicated world, IBM Systems Journal 42 (2003) 462 –483.