Chapter 7

Continuous Practices and DevOps: Beyond the Buzz, What Does It All Mean?


Abstract: DevOps and continuous practices are attracting steadily growing attention by both practitioners and researchers in the software engineering community. The terms are often used inconsistently, interchangeably and with unclear meaning, however. Taking the position that this ambiguity and miscommunication renders the community great harm, in that it impedes our ability to critically appraise these practices, their effects and interplay between them, we analyze how published literature on both continuous practices and DevOps treat the terms. Based on this analysis, along with statements by often cited sources in the community as well as personal experience from researching and practicing these concepts, we propose guidelines to help authors reduce ambiguity in their publications. Additionally, definitions designed to reflect mainstream interpretation while disentangling the terms from one another are presented.

7.1 Introduction

Continuous practices in software engineering have long since ceased to be a curious novelty, but have become firmly established in the industry mainstream. The precise moment of their birth is up for debate: depending on whether one counts the introduction of continuous integration as a term by Booch (1994), the definition of Extreme Programming in 1997 or its subsequent popularization by Beck (1999), these practices are now in their early twenties or late teens. During that time they have not only greatly influenced industry practice, but also diversified and stirred substantial interest in the research community, not least in their effect on related aspects of software engineering, such as configuration management, traceability, communication, testing and planning.

As a more recent development, DevOps has risen to the fore as a prominent trend in the software engineering community and attracted significant and increasing attention from researchers in the last few years, particularly since 2014.

The actual meaning and the internal relationships of these practices to one another is far from clear, however. In previous work (Stähl and Bosch 2014b) we have found that there is a high degree of divergence in how continuous integration, specifically, is interpreted and implemented. This confusion is not limited to continuous integration, but is arguably even more pronounced in the case of continuous delivery and continuous deployment – terms which are often used interchangeably (Rodriguez et al. 2016). Similarly, DevOps is notorious for its ambiguity and lack of established definitions, and a large overlap with continuous practices exists. This has been
recognized in literature (Lwakatare et al. 2015) as well as by forefront figures in the
industry: in the words of Andrew Phillips, “some people say continuous delivery is the
goal of DevOps, some people say DevOps is the [enabler of] continuous delivery” and
“unfortunately […] people use the terms almost interchangeably” (interview on SE
Radio 2016).

In response to this state of confusion, we have conducted a systematic mapping of
literature to assess the current state of understanding of DevOps and continuous
practices, particularly in relation to one another. Based on the results of this, in the
interest of greater short-term clarity in published literature we propose guidelines for
researchers, and in the hopes of long-term consensus propose a disentanglement of
these related, but ultimately distinct, concepts.

The key contribution of this paper is three-fold. First, it highlights a critical problem
in the software engineering community by assessing the state of understanding and
level of consensus regarding the meaning of continuous practices and DevOps in
contemporary literature. Second, it proposes recommendations for researchers to
improve the level of conceptual clarity in published literature. Third, it proposes a set
of definitions which offer a way forward to disentangle DevOps and continuous
practices from one another.

The remainder of the paper is structured as follows. The next section discusses the
problem statement and its importance in greater depth. This is followed by the research
method in Section 7.3 and the results of the systematic mapping study in Section 7.4.
These results are then analyzed in Section 7.5. Finally, the validity of the study is
discussed in Section 7.6 whereupon the paper is concluded in Section 7.7.

7.2 Problem Statement

While the lack of a consensus regarding the exact meaning of continuous practices and
DevOps – not least in relation to one another – is difficult to refute, one can take
different views on it. Indeed, one may argue that “having no clear definition [of
DevOps] has its advantages” because it raises questions and allows a high degree of
freedom in combining methods and tools “within one high level development method”
(Olszewska and Waldén 2015).

We, on the other hand, take the opposite and arguably more conservative view.
Similarly to the divergence observed and documented previously with regards to
continuous integration – and which we believe has slowly improved since then – the
current lack of consensus regarding the actual meaning of DevOps, continuous delivery
and continuous deployment impedes the systematic building of knowledge regarding
pros, cons, challenges and possible solutions. As long as the statement “By adopting
DevOps, we have achieved benefit X” semantically equates to “By changing something
related to our software production we have achieved benefit X” it is a near impossible
task to analyze, evaluate and ultimately evolve the practice through systematic research
efforts. Similarly, for a practitioner, as long as any attempt to investigate how DevOps
might benefit one’s software development efforts begins by deciding what DevOps is,
learning by example and making informed decisions becomes very challenging indeed.
As one source puts it, as long as “little has been presented to describe and formalize
what [DevOps] constitutes [...] the phenomenon will not be effectively communicated” (Lwakatare et al. 2015).

Consequently, we seek to address this challenge by answering the following research question: *What, if any, are the consensus definitions of continuous practices and DevOps, particularly in relation to one another, in literature on continuous practices and DevOps, and can such definitions be found which reflect mainstream usage yet cleanly separates the terms?*

### 7.3 Research Method

To answer the research question outlined in Section 7.2 we chose to first conduct a systematic mapping study (Petersen et al. 2008) of the joint field of continuous practices and DevOps, followed by a more in-depth review of relevant papers. The purpose of this mapping study was to investigate how the research field of DevOps has developed in relation to the highly related field of continuous practices, and to identify candidates for in-depth analysis.

The steps proposed by Petersen et al. (2008) when conducting a systematic mapping study are:

- Definition of Research Question
- Conduct Search for Primary Studies
- Screening of Papers for Inclusion and Exclusion
- Keywording of Abstracts
- Data Extraction and Mapping of Studies

Each of these steps is discussed individually below. The step-by-step process is also depicted in Figure 24.

![Figure 24: The step-by-step mapping study process.](image)

#### 7.3.1 Definition of Research Question

The first step of the study was to phrase the research question presented in Section 7.2.
7.3.2 Conduct Search for Primary Studies

To identify published literature dealing with continuous practices and DevOps, we wanted to ensure that we did not unwittingly exclude or misrepresent literature because of any personal mis- or preconceptions with regards to nomenclature. Hence we opted for as automated and open-ended an approach as possible.

The first step of this approach was to search for continuous practices. While the most popularized terms are continuous integration, delivery and deployment, several other continuous practices have been proposed over the years; Fitzgerald and Stol (2015) provides an overview of nine practices which we have based our search on without modification. To identify published literature dealing with one or more of these practices, a Scopus search was conducted.

The decision to use only one indexing service was made based on two factors. First, in previous work (Ståhl and Bosch 2014b, Ståhl and Bosch 2016a, Ståhl et al. 2016b) we have found Scopus to cover a large majority of published literature in the field, with other search engines only providing very small result sets not already covered by Scopus. Second, restricting oneself to a single source has the benefit of consistency in nomenclature and available meta-data – enabling our automated approach to analysis – as well as avoiding the risk of introducing errors when merging results from multiple sources.

The search string was developed in a two-step process. First, TITLE-ABS-KEY("continuous integration" OR "continuous delivery" OR "continuous release" OR "continuous deployment" OR "continuous verification" OR "continuous testing" OR "continuous compliance" OR "continuous security" OR "continuous evolution") AND PUBYEAR > 1996 AND PUBYEAR < 2017 AND SUBJAREA(COMP) AND DOCTYPE("ar" OR "bk" OR "ch" OR "cp" OR "rp" OR "re") was searched to find all items related to any of the identified continuous practices within computer science. This search, however, returns a large number of “false positives”, such as research on drug release mechanisms in medicine. In the interest of reproducability of the study we decided to attempt to eliminate as many of these unrelated items as possible by refining the search string, rather than by manual assessment of the result.

To improve the accuracy of the result set, the keywords of all items were parsed, counted and sorted according to number of publications including them. All keywords occurring in ten or more of the publications were reviewed, and of these all which were related to software engineering in general (e.g. “Software Design”) or continuous and/or agile practices in particular (e.g. “Continuous Integration”), were added to the search string (... LIMIT-TO(EXACTKEYWORD, "...") OR LIMIT-TO(EXACTKEYWORD, "...") ...), thereby excluding all items not listing at least one of these keywords. The full resulting string is not included due to space limitations. The search was performed on October 10, 2016 and yielded 458 hits.
7.3.3 Keywarding of Abstracts

While Petersen et al. (2008) present recommendations on analysis and classification of included papers, we chose a fully automated approach to avoid any human error. For each of the 458 publications, the following data was collected: title, abstract, keywords (both author keywords and indexed keywords, where present), authors, affiliations, publication year, type of publication and number of citations. The data set was then automatically parsed using two methods.

First keywords, titles and abstracts were searched for phrases, which were sorted by number of papers making mention of them. In the case of keywords this is straightforward, but when analyzing running text one can not only analyze individual words: e.g. “continuous integration” would not show up in such an analysis. Consequently we performed n-gram analysis (Kohonen and Somervuo 1998) of title and abstract for N values 1, 2 and 3, respectively (i.e. single words, bigrams and trigrams). In the next step the resulting list of phrases was analyzed, and similar phrases were thematically grouped. To exemplify, the common keyword “continuous integrations” and the common bigram “continuous integration” were grouped into a single theme. These themes were then used to represent a single semantic concept, with DevOps being one of these themes, containing a total of 35 items. This process of extracting themes is depicted in Figure 25.

The end result of this approach is equivalent to appending AND TITLE–ABS–KEY () with all phrases of the theme to the original search string. The benefit, however, rather than that of manually selecting filter criteria up-front, is that we do not run the risk of accidentally filtering out relevant publications because of alternative spelling, syntax or terminology. To exemplify, common n-grams in the result set were “automated testing”, “test automation”, “automated test” and “automated tests”. Unaware of this divergence, a researcher constructing an up-front search query on test automation may inadvertently miss out on relevant publications.

Other metrics, such as number of affiliations, number of authors, publication type et cetera were parsed and tabulated by bespoke scripts.
7.3.4 Data Extraction and Mapping of Studies

Following the above steps, the resulting table of classified publications was analyzed, first to create an overview of current trends in the research field and second to investigate definitions – explicit or implicit – among the publications within the DevOps theme. Such definitions were extracted and grouped as pertaining to one or more continuous practices, to DevOps or to the relationship between these, respectively.

In Section 7.4 the results of this collation is presented without in-depth analysis, whereas in Section 7.5 we analyze it within a broader context, including often cited authors in the continuous practices and DevOps space as well as our own experiences and observations as researchers and practitioners.

7.4 Results

This section presents the results of the mapping study.

7.4.1 Overview and Trends

One advantage of the automated approach described in Section 7.3 is that it generates a richer data set than is practical using strictly manual analysis and classification. In turn, this allows for creating an overview and studying trends.

Figure 26 shows the number of publications over time per extracted continuous practice related theme, as well as in total. A quick look at the compiled data shows that the research field of continuous practices is steadily growing, as has been reported in related work (Rodriguez et al. 2016), and that the three practices that are currently attracting the most attention are continuous integration, delivery and deployment. Particularly the latter two are rising sharply, while continuous integration is actually on the decline, relative to the total number of publications.

Figure 26: Number of publications related to individual practices over time.
Figure 27, on the other hand, shows trends in the methodologies addressed in literature on continuous practices. Here the Agile theme includes a wide range of non-DevOps phrases, such as “agile methods”, “agile manufacturing systems”, “scrum”, “agile practices”, “extreme programming” and “lean software development”. Similarly, the Other theme includes multiple phrases related to methodology in general, e.g. “iterative methods” and “development practices”, but which cannot be classified as strictly Agile (interestingly enough, not a single common phrase made reference to any particular non-Agile methodology). The figure clearly shows that while the interest in Agile has stayed strong over time, literature on continuous practices has increasingly begun addressing DevOps as well. Indeed, relative to the total number of publications, the DevOps theme is increasing much more rapidly than the Agile theme and began to take off at exactly the same time as continuous delivery and continuous deployment, which could imply a close link between these particular practices.

![Figure 27: Number of publications related to methodologies over time.](image)

To summarize, we find evidence that interest in the research community regarding continuous practices is strong and growing. Particularly continuous delivery and deployment are on the rise, which coincides with a similarly rapid increase in attention to DevOps.

### 7.4.2 Definitions

While it can be shown that literature on continuous practices is increasingly mentioning continuous delivery, continuous deployment and DevOps, given the confusion regarding definitions outlined in Sections 7.1 and 7.2, that does not necessarily mean they are discussing the same subjects. Consequently, the more fundamental question is how these publications define the terms, explicitly or implicitly.

1) **DevOps**: A full review of the 35 publications in the DevOps theme (see Section 7.3) reveals that some actually do not address – much less discuss the meaning of – DevOps at all. Indeed, some may include DevOps as a keyword or index term without a single mention in the article itself (Chen 2015a, Bass et al. 2015) or only mention it in passing (Zhu et al. 2015, Xu et al. 2014, De Gouw et al. 2016).
That being said, a larger number do define or at least describe the concept. e.g. stating that it is “a software development method that combines quality assurance mechanisms with IT operations within software engineering practices” (Olszeweska and Waldén 2015), whereas others argue that it is “not a method”, but an approach (Erich et al. 2014).

Meanwhile, others take a much more technical stance, stating that DevOps aims “to automate the complete deployment process from the source code in version control to the production environment” (Wettinger et al. 2014a) or discuss the “DevOps toolchain” and the DevOps practices of “fast feedback, small batch sizes, and independent releases” (Callanan and Spillane 2016). Similarly, Virmani (2015) finds that DevOps is a "set of principles towards software delivery [...] speed of delivery, continuous testing in production like environment, be in shippable state at any day, continuous feedback, ability to react to change more quickly, teams working to accomplish a goal instead of a task”.

We find the most common interpretation, however, to be that DevOps is about culture: a “culture that stresses collaboration and integration between software developers, operations personnel, and everyone involved in the design, creation, development, and delivery of software” (Gotimer and Stiehm 2016) or “an emerging culture in which development, testing, operations teams collaborate to deliver outcome in a continuous and effective manner” (Soni 2015) or that it “includes organizational and cultural aspects” (Ahmadighohandizi and Systä 2015) and constitutes “a culture shift toward collaboration between development, quality assurance, and operations” (Ebert et al. 2016) where “engineering activities [...] are shifted towards earlier stages of development” (Liu et al. 2016).

A final perspective is, arguably, a more holistic one. Humble and Molesky (2011) argues that while “Devops is about aligning the incentives of everybody involved in delivering software”, it does that in four ways: “culture, automation, measurement, and sharing”. Similarly, based on interviews and review of 22 publications, Lwakatare et al. (2015) identifies four elements of DevOps: collaboration, automation, measurement and monitoring. While these two lists of elements are not identical, they do provide a clue to the underlying nature of the divergence: while some regard DevOps as a very broad and multi-faceted concept, others zoom in one part or other – such as culture or automation – and instead think of that as DevOps. Indeed, this also provides a lens through which we may regard the relationship between continuous practices and DevOps, discussed further in Section 7.5.

2) Continuous Practices: Similarly to the case of DevOps, it is not uncommon for the 35 identified publications in the DevOps theme to only list one or more continuous practices in keywords or mentioning them in the abstract, without actually addressing them.

As found by Rodriguez (2016), many sources treat continuous delivery and continuous deployment as one and the same, using the terms interchangeably. To exemplify, Ebert et al. (2016) includes deployment of software to products in the field in continuous delivery, yet also discusses continuous deployment in the same paper without clarifying the distinction between the two concepts. Similarly, one source considers continuous delivery to also involve actual release of the software and
consequently argues that “continuous deployment is a prerequisite for continuous delivery, but the reverse is not necessarily the case” (Fitzgerald and Stol 2015). Others define the practices separately, but there is a high degree of divergence between sources. To exemplify, Virmani (2015) states that continuous delivery “tries to optimize the infrastructure management and the critical need to balance out time and resources”, the benefits of which include “no need for dedicated hardware” and reliable and scalable infrastructure provisioning – considerations and benefits arguably more related to cloud infrastructure engineering than to continuous delivery practice, per se.

That being said, the most common interpretation as we find it is that continuous delivery is about ensuring the software can be released and/or deployed to production at any time (Chen 2015a, Soni 2015, Chen 2015b, Krusche and Alperowitz 2014), but may not actually be thus released and/or deployed, and that the continuous delivery pipeline helps build “confidence that the software is a viable candidate” (Gotimer and Stiehm 2016). Furthermore, while this is “often erroneously considered synonymous with automated deployment” (Gotimer and Stiehm 2016), there is an important distinction. This also serves to “significantly shorten software release cycles” (Wettinger et al. 2015a) and enable more frequent releases (Krusche and Alperowitz 2014). Continuous deployment, on the other hand, is to actually put those candidates into production (Zhu et al. 2016), “with as much automation as possible” (Shahin 2015).

In the case of continuous integration, it is described as “integrate early, don’t keep changes localized to your workspace for long” (Virimani 2015) or “a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily” (Soni 2015), which in turn is a direct quote from Martin Fowler’s often cited article (Fowler 2006). It is also found that this typically “comprises interconnected steps such as compiling code, running unit and acceptance tests, validating code coverage, checking compliance with coding standards, and building deployment packages” (Fitzgerald and Stol 2014). That does not mean that alternative interpretations are not published, however. Fitzgerald and Stol (2015) present continuous integration as a superset where “a number of further modes of continuous activities can be identified, namely continuous deployment and continuous delivery”, while Fitzgerald and Stol (2014) argues that continuous integration “has emerged to eliminate discontinuities between development and deployment”.

Finally, Fitzgerald and Stol (2015) defines a larger set of continuous practices, expanding the scope to include business strategies, most of which have not (yet) come into popular use in literature and are therefore not discussed here.

3) DevOps vs Continuous Practices: Studying how sources treat DevOps in relation to continuous practices reveals further divergence in literature. A number of sources consider DevOps to be an enabler of continuous practices (Fitzgerald and Stol 2015), e.g. stating that “DevOps [enables] rapid, continuous deployment and delivery” (Wettinger et al. 2015b), that “the main promise of DevOps is to enable continuous delivery” (Wettinger 2014b) or implying that continuous delivery is a superset of DevOps (Krusche and Alperowitz 2014).

The opposite perspective, that DevOps is enabled by continuous practices and/or constitutes a superset of continuous practices, is far more common, however (Wettinger et al. 2015a, Gotimer and Stiehm 2016, Olszewska and Waldén 2015, Wettinger et al.
2015b). To exemplify, it is stated that continuous delivery “promises to radically reduce frictions in DevOps processes” (Austel et al. 2015), that DevOps “builds on” continuous practices (Ahmadighohandizi and Systä 2015), that continuous deployment is the “heart of DevOps” (Virmani 2015) and that DevOps drives the need for continuous integration and delivery (Wright and Druta 2014). While differences between DevOps and continuous practices are rarely discussed explicitly, Ahmadighohandizi and Systä (2015) finds that DevOps, unlike continuous integration and delivery, also “includes organizational and cultural aspects”.

Related to the fact that most sources do not explicitly address the relationship and/or differences between continuous practices and DevOps is the finding that apparently conflicting views are sometimes presented in the same source. To exemplify, Shahin (2015) states that DevOps exists “in the context of Continuous Deployment”, and later states that continuous integration, delivery and deployment are “DevOps practices”.

7.5 Analysis and Discussion

This section presents the analysis of the results presented in Section 7.4.

7.5.1 Assessment of Divergence

From the review of the studied sources it is evident that even though we find what we consider to be a mainstream interpretation in line with influential sources such as Jez Humble (Humble and Farley 2010, Humble 2010) and Martin Fowler (Fowler 2006, Fowler 2013), there is a high degree of divergence – not to say confusion – with regards to the actual meaning of continuous practices, DevOps, and how these terms relate to one another. We agree with others in the community who have made similar observations that this renders both researchers and practitioners great harm. There is a clear danger that if we are unable to agree on the meaning of these buzzwords and they ultimately become meaningless: we end up in a situation where everybody ostensibly practices continuous delivery and DevOps, though in practice very little has actually changed, resulting in backlash and disillusionment until the next buzz word comes along and the cycle repeats. In this section, we present one short term and one long term approach to avoid that scenario.

Before delving into these approaches, however, we would pause to reflect on the reasons for the current state of disagreement. We posit that it is not altogether surprising: the software industry is, after all, highly diverse. Software is developed in different contexts, using different processes, in different technology domains, targeting different types of users and operating in under different regulatory circumstances and in different business contexts. Depending on the experiences of the individual author, it is to be expected that not all have had reason to reflect on the actual differences between these practices or that they may seem similar to the point of conflation. To exemplify, given certain SaaS (Software as a Service) contexts deploying to production and releasing to users may seem like one and the same, given certain business policies being able to deploy a given version of the software may not be different from actually doing it, and given small to medium sized projects, the distinction between continuous
integration and delivery may seem vague (Ståhl et al. 2017b). That being said, another possible explanation is one of simple demographics: as more people enter the domain of software engineering in general, and continuous practices in particular, the finer points of the terminology and underlying concepts may appear less clear than to newcomers than to old-timers, as hinted at by Andrew Phillips (interview on SE Radio 2016).

7.5.2 Short Term Approach: Explicitness

As a short term step to remedy the current situation we strongly encourage colleagues and fellow authors in the community writing and speaking about continuous practices and DevOps to be explicit regarding the meaning they attach to those terms – not only in a positive sense, but also in a negative sense.

In other words, it is important to communicate not only what one intends by e.g. “continuous delivery”, but also what one does not intend. To exemplify, if by DevOps one means a culture of collaboration between development and operations, does one also include the automation and infrastructure that enables and fosters that culture?

7.5.3 Long Term Approach: Viable Definitions

Ideally, the shared understanding of continuous practices and DevOps would be such that explicit definitions would be superfluous. In support of that and to take our own advice from Section 7.5.2, we wish to propose what we consider to be a set of viable definitions. We base this proposal on what we interpret to be common perspectives in the reviewed literature, as well as influential sources in the industry, along with years of personal experience applying these practices at operative, tactical and strategic levels in organizations ranging in size from dozens to tens of thousands of engineers. In doing so, our goal is not to break new ground or to redefine anything, but rather the opposite: to cause minimal disruption and reuse what can be reused, while hopefully adding a certain level of clarity by disentangling separate but frequently conflated concepts.

We argue that a conducive set of a definitions would be one where the terms are either mutually exclusive or explicit supersets of one another, in order to avoid the current situation where continuous integration, continuous delivery, continuous deployment and DevOps are frequently used as vaguely defined expanded subsets of one another.

Starting with the three most popular continuous practices, we would suggest that they may be understood as follows:

- **Continuous integration** is a developer practice where developers integrate their work frequently, usually each person integrates at least daily, leading to multiple integrations per day. This is taken almost verbatim from the often cited definition by Martin Fowler (Fowler 2006), with two important exceptions. First, considering it a developer practice, as opposed to a development practice, emphasizes that it is ultimately about developer behavior and helps distinguish it from continuous delivery. In a small scale context the distinction may be subtle, but in large scale
development we see that continuous delivery does not necessarily imply continuous integration (Ståhl et al. 2017b). In other words, a state of the art automated software pipeline does not guarantee that developers actually integrate continuously, although it may well enable and encourage that practice. The second exception is the substitution of developers for members of a team, which implies a single-team setup and causes unnecessary confusion. It is a well established fact that continuous integration is often practiced at much larger scale in the industry; consequently we argue for defining the term without connotations of organizational units.

- **Continuous delivery** is a development practice where, in the words of Humble and Farley (2010), every change is treated as a potential release candidate to be frequently and rapidly evaluated through one’s continuous delivery pipeline, and that one is always able to deploy and/or release the latest working version, but may decide not to, e.g. for business reasons. Emphasis is placed on potential, because particularly in large-scale implementations of the practice, it is not feasible to apply the full test and verification battery to every change in source code – rather, changes are often batched throughout the pipeline, meaning that commits are essentially sampled, in the sense that only certain source code commits end up being built into release candidates that consequently evaluated as such. This sampling, however, is a completely heuristic process, rather than a planned or otherwise pre-ordained one. Also, to distinguish continuous delivery from continuous integration it is important to note the difference between developer practice and development practice: an organization may practice continuous delivery and implement a continuous delivery pipeline, yet at the same time fail to convince its individual developers to adopt the practice of continuously integrating. The scope of continuous delivery varies from case to case, and is defined by whatever one needs to do to confidently release and/or deploy the software. This typically involves activities such as code analysis, documentation generation, acceptance testing, regulatory compliance assessments, license scanning and requirement verification.

- **Continuous deployment** is an operations practice where release candidates evaluated in continuous delivery are frequently and rapidly placed in a production environment, the nature of which may differ depending on technological context. This often, but not necessarily, implies making it generally available to users, while in other contexts it is not even applicable as a concept.

In addition to these, based on our own industry experience and along the lines of Jez Humble’s reasoning (interview on SE Radio 2015), we see a strong reason to include continuous release:

- **Continuous release** is a business practice where release candidates evaluated in continuous delivery are frequently and rapidly made generally available to users/customers. This is critically different from continuous deployment, depending on the context. In many SaaS environments, release is achieved through deployment, but this is not necessarily the case (consider e.g. dark launches). In other contexts, where an organization is deploying for its own internal operational needs and thereby is its own user, the concept of a release is arguably not even applicable. Conversely, in the case of user installed software, continuous deployment is not an applicable
concept even though continuous release may very well be. For these reasons, we find that the inclusion of continuous release as a separate concept often helps clarify otherwise muddled conversations.

- **DevOps**, on the other hand, is a more expansive concept, and therefore arriving at a concise definition is much more difficult. As found in Section 7.4.2, it is commonly referred to as a culture or mindset, but is also defined as being more holistic: yes, it is about culture and values, but also about tooling, processes and concrete practices. This view is explicitly or implicitly advocated by several forefront figures of the movement (Humble and Molesky 2011, Debois 2012) and elaborated in depth by Mueller (2016), structuring DevOps into values, principles, methods, practices and tools, similarly to how Agile is often described. This view of DevOps is depicted in Figure 28.

\[
\text{Figure 28: DevOps as a combination of values, principles, methods, practices – including continuous practices – and tools.}
\]

Admittedly, there is a certain level of uncertainty as to exactly which the included values, principles, methods, practices and tools are, but continuous practices can be thought of as constituting certain “specific techniques used as part of implementing the [DevOps] concepts and processes” (Mueller 2016). There is already a number of attempts at listing the constituent parts of DevOps (Mueller 2010, Honor 2010, Willis 2010), and making another such listing in this context is arguably of limited value. Instead, while strongly promoting the notion of DevOps as a superset, we return to our short term approach (see Section 7.5.2) and encourage authors to be very explicit in what they intend when addressing DevOps, particularly with regards to actual methods, practices and/or tools. This is because we argue that not only are values and principles very difficult to pin down and quantify, but they are only of indirect importance: one’s behavior affects external reality, whereas one’s intrinsic values and motivations for that behavior do not. In this sense principles and values are only important to the extent that
they influence behavior. Consequently, we argue that while principles and values are useful in conveying an idea, in trying to understand, investigate and evaluate DevOps, the community must go beyond that and focus on actual behaviors and their consequences.

That being said, we argue that such a definition, where DevOps forms of a superset of continuous practices along with other values, principles, methods and practices, is highly useful in that it provides a conceptually clear framework of how DevOps and continuous practice relate to one another: it enables practitioners to reason about what to adopt and how, and it provides researchers the opportunity to systematically approach the problem of investigating the interplay between specific continuous practices and DevOps principles, processes and methods.

Returning to the original research question (see Section 7.2), it is our conclusion that no clear consensus regarding definitions of continuous practices or DevOps exists in published literature addressing both subjects. However, definitions that cleanly separate the concepts yet reflect the most common views expressed in that literature are indeed possible.

7.6 Discussion of Validity

This study has investigated published literature on continuous practices and DevOps, in order to discern how these terms are used and defined – implicitly or explicitly. One may object that there is a substantially larger body of literature on only continuous practices and on only DevOps, respectively. Additionally, there is even greater activity on various blogs, podcasts and social media, where the shared understanding of these terms is arguably decided to a higher degree than in academic publications. While not arguing the point, the intersection of continuous practices and DevOps is of particular interest to the purposes of the study, and based on our experience as researchers and practitioners, we consider the publications we have reviewed to be largely representative to general usage of the terms in the industry.

Another relevant concern is whether the algorithm used to classify publications is sound. The first aspect of this is whether it produced correct results. To protect against this, we have verified that its results are identical to including the discovered phrases in the original search string for each of the themes it produces. The second aspect is conceptual soundness. We argue that it is superior to manual up-front determination of phrases and keywords to search for, as it relies on unprejudiced discovery of phrases and therefore promises to be more inclusive of relevant concepts unexpected by or unknown to the researcher. In this vein, we have in other work applied the same process to research on other areas related to continuous practices, and thereby successfully identified relevant publications we did not discover in parallel searches using manually selected keywords.
7.7 Conclusion

In this paper we have started out from the recognized problem that both continuous practices and the term DevOps are vaguely defined and loosely used in the software engineering community. This is highly problematic, as such confusion regarding semantics stands in the way of critical and systematic analysis of the concepts, their effects and how they interact with one another. This in turn prevents the community from building upon a solid foundation of proven experience, rather than being seduced by a series of buzzwords which may turn out to affect very little in terms of real improvement in the daily lives of software professionals.

We have conducted a systematic mapping study to identify published literature addressing both continuous practices and DevOps, and then reviewed the identified publications to investigate usage of continuous practices and DevOps. Based on this review we find that there is indeed a high degree of confusion and contradictions – often internally in the same source – regarding the meanings attached to these terms. We encourage authors to reduce ambiguity in published literature. Furthermore, putting these results into the larger context of statements by often cited sources in the community as well as our experiences as researchers and practitioners, we present a set of definitions that closely corresponds to what we consider to be a mainstream interpretation, while disentangling the terms, thereby providing a conceptual framework that supports systematic study and evaluation of the practices.

Finally, we consider the highly automated process of literature mapping and classification used in this study to be of great value, as it relies on unprejudiced discovery of phrases and is therefore able to include relevant concepts unexpected by or unknown to the researcher.

Further work remains, however. In this paper we have found that the software engineering community is indeed far from reaching a well established consensus regarding the definition of continuous practices and DevOps – particularly their relations to one another. While this can be considered an area for further work, it is also important to recognize that this is a development that largely comes down to non-academic arenas: social media, blogs, industry talks, books et cetera. That being said, researchers can play an important role in nudging the community in a more scrupulous direction when it comes to terminology, or at the very least not contribute to the confusion.

In this paper we have also made the conscious decision not to build a taxonomy of DevOps values, principles, methods, practices and tools, but rather to focus on what one might consider a “meta definition” of the concept. That does not mean such a taxonomy lacks value, but we argue that it would need to be underpinned by further work: research into individual methods, practices and tools often associated with DevOps, and the interplay between them. Important questions to answer in this area include “Which practices, tools and methods are typically used together?”, “How are they typically implemented?” and “Which values and principles are confessed to, and how do they correlate with methods, practices and tools?”. Finding the answers to such questions, supported by a clear “meta definition” such as the one presented in Section 7.5.3, would enable the software engineering community to progress by transforming
DevOps from a buzzword to something more like a recipe. A recipe that offers the individual engineer clear guidance: given a certain situation and a certain set of goals, which values, principles methods, practices and tools have been shown to be conducive in achieving those goals?

Finally, we welcome any further development and improvement of the automated approach to systematic literature mapping used in this study, as we believe it has the potential to add significant value to the community, given the rapidly increasing volume of publications.