Software Development Context: Critiquing Often Used Terms

Diana Kirk
Independent Researcher, Auckland, New Zealand

Keywords: Software Development Context, Situated Software Practices.

Abstract: Software development practices are carried out in a range of different contexts and there is little evidence to support the likelihood that a specific practice will be effective in a specific context. The need to understand the relationships between practice and context is pressing if we are to support industry in selecting a set of practices that will provide maximum benefit for the project. An evidence based approach requires that, for each practice, we associate a 'context profile' that describes the operating parameters for the practice. Our earlier explorations resulted in a categorisation of the huge number of terms used when describing software development context. We found that some terms could not be included in the context profile because the term described context at a high level, was vague or its meaning was ambiguous. In this position paper, we overview the findings from our explorations of context and present and critique the inadmissible terms with respect to their application to situated software practices. The cataloguing of these terms will benefit researchers by supporting discussion and thus a deeper understanding of context for situated software practices.

1 POSITION

Our position for this paper is stated as:

1. We are currently not in a position to support organisations make decisions about implementing a software development practice because we do not have evidence linking practice efficacy to specific project context.
2. Accumulating such evidence requires a suitable abstraction for context for situated software practices.
3. Such an abstraction is crucial if we are to progress with supporting organisational decision-making.
4. There are many terms used to describe context that are unsuitable for understanding situated software practices.
5. A consideration of these terms will benefit researchers by supporting discussion and thus deeper understanding of context for situated software practices.

2 INTRODUCTION

The architects and proponents of software processes and methodologies have lagged behind industry in understanding that, rather than being implemented as prescribed, methodologies are inevitably adapted for use in specific project environments (Avison and Pries-Heje, 2008; Kuhrmann and Münch, 2019; MacCormack et al., 2012; Müller et al., 2009; Petersen and Wohlin, 2009a; de Azevedo Santos et al., 2011; Turner et al., 2010). Adaptation is often at the project level and generally involves tailoring of specific practices based on the prior experience of project team members. The inference is that investigating adaptation and tailoring at the level of the process or methodology is probably not helpful and attention is better focused at the level of the *practice*.

One result of the persistence of the prescriptive approach we have witnessed over the years is that the issue of *context* has been either ignored or treated somewhat informally. For example, the term does not appear in the IEEE Standard Glossary of Software Engineering Terminology (Institute of Electrical and Electronic Engineers., 1990) and, as far as we are aware, context is not represented in the Software Engineering Body of Knowledge (SWEBOK). The lack of inclusion of quality terms in the Standards is also reported by an ITiCSE Working Group (Börstler et al., 2018). The relationship between process and context has only recently become popular as a focus of serious investigation in mainstream research (Klünder et al., 2020).

Kirk, D.
Software Development Context: Critiquing Often Used Terms.
DOI: 10.5220/0010469903400347
In *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2021)*, pages 340-347
ISBN: 978-989-758-508-1

As highlighted in Section 1, we believe a deeper understanding of context is crucial if we are to support researchers investigating situated software practices with the longer term aim of advising practitioners. Our earlier work on this topic resulted in the creation and refinement of a dimensional model for context (Kirk and MacDonell, 2018). As the number and possible combinations of contextual factors is enormous, our approach was to *abstract* the problem space as a set of dimensions onto which each factor might be mapped. The resulting model is now in the 'evaluate-and-refine' stage of development (see Section 4).

During model creation, we, as expected. encountered many terms describing contextual factors. However, when considering each term with respect to how it might be applied in the case of a specific situated software practice, we realised that many of the commonly stated terms were unhelpful or irrelevant, In some cases, the factor described a high level notion that might certainly affect decision-making about project strategy, but was not *directly* applicable to a specific implementation instance of a practice. For example, the factor 'going global' might result in a decision to establish off-shore teams. This would certainly affect practice selection, but in an *indirect* way. We categorised these as *strategic* factors. In other cases, the *meaning* of the term was unclear. There were two types of unclear terms. Some were vague in meaning. For example, the term 'off shore development' does not help us understand whether or not a practice might be useful, because we do not know which teams are offshore, the degree of time difference or whether political constraints apply. As such factors can be broken down into more basic elements, we categorised them as *secondary*. Other terms were *ambiguous* in meaning. For example, 'uncertain requirements' might mean the client doesn't really know what (s)he wants, or might mean the communication between client and developers is problematic. Each meaning would indicate different practices.

In this paper, we collect the unhelpful terms exposed during earlier investigations, informally categorise similar terms and present these as a catalogue of common terms. The objective is to provide a deeper clarity in the area of situated software context and to prove a starting point for discussion by other researchers in this area.

In Section 3, we overview other work on establishing software context. In Section 4, we present the model evolved from earlier investigations and briefly discuss the model elements. In Section 5, we present and discuss the terms that represent strategic, vague and ambiguous ideas and provide rationale and counter-examples In Section 6, we summarise the contribution.

3 RELATED WORK

There have been many efforts to relate Software Engineering (SE) outcomes to specific key factors. We overview a selection here.

Avison and Pries-Heje aimed to support selection of a suitable methodology that is project-specific (Avison and Pries-Heje, 2008). For a given project, the authors plotted position along each of eight dimensions on a radar graph and inferred an appropriate methodology from the shape of the graph. We see two limitations. First, the abstraction is based on a specific organisation, resulting in missing contexts, for example, temporal distance. Second, it is based at the level of the *project* and so is inapplicable to, for example, a 'customer-driven' environment, where the on-going relationship between development group and customer becomes key (Dingsøyr and Lassenius, 2016; Munezero et al., 2017; Stuckenberg and Heinzl, 2010).

Clarke and O'Connor propose a reference framework for situational factors affecting software development (Clarke and O'Connor, 2012). The framework includes eight classifications: *Personnel*, *Requirements*, *Application*, *Technology*, *Organisation*, *Operation*, *Management* and *Business*, further divided into 44 factors. Our critique of this approach is that the *meanings* assigned to sub-factors do not represent a consistent set with respect to practice suitability. For example, the factor 'Cohesion' includes "team members who have not worked for you", "ability to work with uncertain objectives" and "team geographically distant", each of which might indicate different kinds of practice. The framework may indeed provide a comprehensive list of factors. However, the approach remains discrete in nature and is unsuitable for classifying factors in a theoretical way, as the categories are semantically inconsistent and there are no clear rules on which to base abstraction.

Petersen and Wohlin provide a checklist for representing context for the purpose of aggregating studies in industrial settings (Petersen and Wohlin, 2009b). The facets of the structure include *Product*, *Processes*, *Practices*, *People*, *Organisation* and *Market*. The facets and context elements are presented as a given, without justification. While likely useful, our earlier critique exposed issues of clarity and completeness with the checklist. For example, the term 'Language' is ambiguous and could mean 'the language the product is coded in affects developer effi-

cacy' or 'there is an external constraint on the language to be used for coding'. In the 'People' category, team member experience is included but experience is only one of the aspects that might determine efficacy. For example, terms relating to team member motivation and empowerment are missing (Kirk and MacDonell, 2018).

Klünder et al. apply a statistical approach to investigate context for hybrid development methods (Klünder et al., 2020). Data for the study was sourced from a comprehensive questionnaire in which a set of contexts was included for participant selection. The authors created clusters of practices that correlated with common contexts. They found that method (practice) selection was influenced by only a few factors, for example, target application domain. As the contexts were provided by the authors, the study does not represent an exploration of context.

4 MODEL EVOLUTION

In this Section, we overview our earlier research into context for software development and present the resulting research framework. The study is reported fully elsewhere (Kirk and MacDonell, 2018)

Although there exist several proposed frameworks for context, we rejected these for two reasons. First, none emphasises the properties that define category membership and so categorisations are inconsistent from a meaning perspective (see Section 3). Second, we were concerned that the result would not be sufficiently general given the fast-changing nature of software development. For example, newer paradigms such as software-as-a-service (Stuckenberg and Heinzl, 2010) and continuous value delivery (Dingsøyr and Lassenius, 2016) have raised the need to rethink software process. We believed a more conceptual approach would result in a more comprehensive model. This perspective is in keeping with the exploratory process where the researcher begins with a "preliminary notion" of the object of study. During the study the "provisional concepts ... gradually gain precision" until a suitable conceptualisation is achieved (Routio, 2007). Routio suggests that the journey may involve some "creative innovation" (Routio, 2007). For this research, we adopted a mixed method, sequential exploratory paradigm (Creswell, 2014), appying a combination of approaches to better understand the problem space (Easterbrook et al., 2008).

We scoped our research to a *software initiative* which we define as "any endeavour that involves defining, creating, delivering, maintaining or supporting software intensive products or services". In Table 1, we overview the activities carried out during the evolution of our proposed framework. The initial concept was based on existing ideas (Dybå et al., 2012; Orlikowski, 2002; Zachman, 2009). The second step involved a small pilot where we categorised into the structure contextual factors named in three software engineering literature studies. We wanted to test that our conceptualisation represented "a starting point (e.g. a framework) that identifies aspects of a topic" (Stol and Fitzgerald, 2015). This step resulted in two main findings (Kirk and MacDonell, 2018). First, we found huge issues with terminology, a problem more recently addressed by Clarke et al., who suggest that the "proliferation of language and term usage" warrants the establishment of an ontological model for software process terminology (Clarke et al., 2016). This is a position we agree with and have explored in relation to some software process constructs (Kirk and MacDonell, 2016). Second, we realised that named terms related to different *kinds* of context i.e. had different meanings. This was a crucial discovery as it led to the exposure of three kinds of term that cannot be applied as-is when discussing context in relation to situated practices. These categories are the topic of this paper, and we discuss in Section 5. The result was an extension of the framework to include these categories.

Table 1: Steps in model evolution.

Activity	Source
Initial concept	Prior work
Pilot categorisation	Literature studies
Extend framework	Results of pilot
Literature categorisation	Literature studies
Small evaluation	Industry projects

The pilot was followed by a more extensive examination of the literature. From each of the included documents, we extracted into a dedicated document words or terms that could be viewed as stating or describing a contextual factor. As in the pilot, our strategy was to be as comprehensive as possible in our identification of contextual factors. This meant that we wanted to expose factors that may not be typically considered as context. For example, the software-as-a-service paradigm has revealed the need for different kinds of practice, but this is not normally viewed as a contextual factor. We thus chose to include studies that contain any thoughts or description about what might affect practice efficacy. We did not evaluate the studies in which the elements were mentioned for quality. We also did not 'tidy up' the found elements by making value judgements about whether

two elements had the same meaning. We felt that such evaluations would effectively remove some of the nuances of identification and would thus compromise our efforts. The underlying issue here is one of a lack of common, agreed vocabulary for software projects. Analysis resulted in a modification of the base dimensions of the framework. This became the *Working context* i.e. the dimensions that include those terms that can be directly applied as 'context' for situated software practices. To evaluate the framework, we carried out a small industry trial involving two organisations.

We overview the framework in Figure 1. In essence, a *Software Initiative* is a set of *Practices* that take place within an *Operational context*.

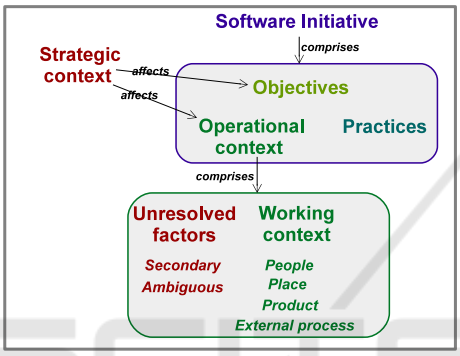

Figure 1: Model for situated software practices.

The explorations described above led us to understand that, in addition to the base dimensions of context (*Working context*), there were many terms mentioned that can not be considered as directly applicable for a consideration of practice efficacy. These are the terms-of-interest for this paper and are shown in red and decribed in Section 5. We overview *working context* in Tables 2 and 3.

Table 2: Working context dimensions.

People	Cultural characteristics affecting peoples' ability to perform
Place	Peoples' availability affecting logistics and communication
Product	Characteristics of the product that is being developed
Process	Processes external to the initiative

5 STRATEGIC AND UNRESOLVED FACTORS

The three categories that we describe in this paper are shown in Figure 1 as *Strategic context* and *Unresolved factors*. *Strategic context* includes factors that are stated at a high level and affect practice efficacy only in an indirect way, as a basis for decisions about *Objectives* and/or *Operational context*. For example, a factor such as 'globalise' certainly may have an impact on development, but in an *indirect* way, for example, by causing teams to be set up remotely. *Unresolved factors* are the operational factors that not sufficiently detailed for direct use. They include *Secondary* factors (factors that can be broken down to more basic elements) and *Ambiguous* factors (factors with multiple possible meanings). For example, 'Company size' is often cited as a contextual factor, but when a specific practice within a specific context is considered, the relevant factors relate to the distribution of teams and organsational constraints on process. 'Company size' is thus categorised as *Secondary* i.e. we need to know more detail. Another commonly cited factor is 'User participation', but this may mean, for example, the user helped in requirements definition, is available throughout the project or carried out beta testing. Each of these meaning has different repercussions for different practices and so the term is *Ambiguous* i.e. we need to know which meaning is intended.

Table 3: Working context factors.

People	Entity	Capability
		Motivation
		Empowerment
		Team cohesion
	Interface	Team cohesion
Place	Entity	Physical distance
		Temporal distance
		Availability
	Interface	Physical distance
		Temporal distance
		Availability
Product	Product type	e.g. embedded
	Lifecycle stg	eg new, mature
	Standards	eg safety
	Requiremts	eg clear, complete
	Implementns	eg consistent
Process	Client	eg specifcn, delivery
	Parent org	eg cultural
	Legal	eg licencing
	Financial	

In Table 4, we catalogue these factors. As the number of terms found during our study was extremely large, we have grouped them into a smaller number of umbrella concepts. For each concept, we show its categorisation as Strategic, Secondary or Ambiguous, some examples of the terms found, and a short rationale for our viewing as not directly ap-

plicable when considering practice effectiveness. The terms in bold font in the rationales map back to the model described in in Figure 1 and Tables 2 and 3. For example, the term 'Proj. structure' is an umbrella term for the many terms we found that describe structure, for example, 'globally distributed project' and 'decentralised approach'. We classify as *secondary* and provide our reasons for this classification i.e. in order to apply this structure to understand a specific software practice, we need specifics about which teams are where and any stakeholder constraints. For example, for a requirements elicitation practice, we would need to know if the analysts and customers in different locations with cultural differences, and any customer constraints on availability for discussions.

The catalogue includes a representative selection only, with the objective of exposing the problem and detailing the kinds of terms found.

As a consequence of the nature of the catalogue entries i.e. many terms are similar and terms are often used in the literature in different ways, each may have multiple aspects i.e. a term may appear as representing more than one category. This is a consequence of the lack of clarity of meaning discussed above. We emphasise that the entries in the catalogue are not definitions, but rather serve to illustrate and raise awareness of the kinds of term found in the literature that claim to be contextual factors for software process, but which cannot be used as-is when considering situated software practices.

5.1 Illustrative Examples

5.1.1 Strategic

These factors often relate to organisational context that manifests as an organisaional goal.

The organisational goal to *go global* might result in a decision to establish teams in the target countries i.e. *Operational context* is affected.

The organisational goal to *gain competitive leverage* might result in a decision to establish product quality as the key objective, to hire design experts, to upskill staff to increase application area expertise, or to overhaul development processes with a view to shortening delivery times. *Objectives* and *Operational context* may be affected.

5.1.2 Secondary

The term *large company* does not inform us about what this means for the *project*. A large traditional company might have a) a small project to trial agile principles, b) a small number of large projects with teams in different places, c) a mix of large and small projects with variations on how teams are constituted. At the operational level, we need to know how the teams are made up (e.g. cross-functional), b) where they are located (spacial and temporal distance) and c) whether the company places any constraints on the project, for example, the need to adhere to company processes.

The term *product domain - health* does not inform us of the kind of product or any expectations and constraints from the client base. For example, software to support patient administration will likely have different product quality expectations than software that will be embedded in a machine to deliver radiation treatment. In addition, the former may have many users with expectations of regular updates whereas the latter may have few users with little desire for upgrades. Each of these aspects must be understood in greater detail.

5.1.3 Ambiguous

The term *uncertain requirements* might mean any of a) the customer doesn't know what they want, b) there are many customers and they want different things, c) the communication between customer and team is problematic, d) the communication among teams is problematic, e) the customer knows what (s)he wants, but is waiting for a third party before decisions can be finalised. Creating a prototype for discussion with the customer will be counter-productive if the issue is d) or e).

The term *stakeholder involved throughout* might mean a) the single customer is readily available for consultation about requirements, b) the single customer has expertise in software development and is essentially part of the team, participating in design decisions, c) the company stakeholder provides support, d) some users are available for testing.

5.2 Discussion

Our long term objective is to provide decision support to practitioners involved in the creation and evolution of software-intensive products by supplying evidence based information about the indicated and contra-indicated contexts for practice efficacy. The objective for this paper is to raise awareness within the software engineering community of the problems involved in pinning down what are relevant contextual factors and by highlighting and cataloguing terms that are often used but are, in actual fact, unhelpful when used as-is. Our intent is that researchers investigating situated software practices will use this catalogue as support when aiming to understand how a practice works within a specific context. Note that we

Table 4: Examples of strategic and unresolved factors.

Context factor / Type	**Examples / Rationale**
Business	Hyper-competitive, innovative, dynamic, fast-moving, inflation, market maturity
Strategic	*Influence decisions relating to project **Objectives**.*
Org. goals	Desire to expand, gain competitive leverage, go global, new line of business, increase portfolio, accelerate innovation
Strategic	*Influence decisions on **Objectives**, **Product type**, team locations (**Place**)*
Org. support	Num. IT professionals, num employees, govt. support, conflicting political beliefs
Strategic	*Influence **Objectives**, staffing, legal and financial decisions*
Secondary	*Depends upon **Process constraints** from the org. and team locations (**Place**)*
Org. type	Software house, IT dept., wholly owned subsidiary, multi-national, product line company, public/private sector
Secondary	*Depends upon **Process constraints** resulting from the org, team locations (**Place**) and **Product type**.*
Org. structure	Flat, matrix, hierarchical, decentralised
Secondary	*Depends upon **Process constraints** from the org. and team locations (**Place**)*
Org. culture	Agile, traditional, cooperative, learning culture, maturity, supports quality
Secondary	*Depends upon **Process constraints** from the org. and team locations (**Place**)*
Proj. size	Project size, project complexity, number of clients
Strategic	*Influence decisions about how to structure and resource project*
Secondary	*Depends upon team locations (**Place**)*
Ambiguous	*Many stakeholders? the product is large? new technologies are being used?*
Proj. structure	Globally distributed, geographically dispersed teams, decentralised, offshore
Secondary	*Depends upon **Process constraints** from the stakeholders. and team locations (**People, Place**)*
Proj. stability	Technological progress, market change rate, dynamism, uncertainty
Strategic	*Influence **Objectives**, **Product type***
Ambiguous	*Is the instability due to fast moving technlogy? unproven teachnologies? lack of experienced staff?*
Proj. defn	Clarity of proposal, requirements stability, conflicting requirements
Ambiguous	*Is the issue due to stakeholders with different viewpoints? client unsure about what is wanted? technology changing?*
Proj. mgmnt	Style, well balanced staffing, PM capabiliy, informal communication, agile
Secondary	*Depends upon **Process constraints** from the PM*
Ambiguous	*stakeholders with different viewpoints? unsure client? technology changing?*
Proj. strategies	Communication mechanism, design approach, work contracted out, on-the-fly team, OS project
Secondary	*Depends upon **Process constraints***
Stakeholderst	User involvement, client involved throughout, participate in design, capability
Secondary	*Depends upon **Client availability and capability**, the nature of participation and **Process constraints***
Ambiguous	*Client available throughout for requirements consultation? users available at specific times for testing?*
Clients	OS developers, system administrators, individuals, inhouse, early adopter charlities
Strategic	*Influence decisions about **Objectives**, and team locations*
Secondary	*Depends upon **Process constraints** from client base and team locations (**Place**)*
Product domain	Medical health, education, insurance, research, corporate, telecommunications
Secondary	*Depends upon **Product type** and **Process constraints** from client base*

do not view the catalogue as a taxonomy. The kinds of term included are inherently lacking in strict definition and cannot be used for the purpose of formal categorisation. The catalogue is an informal listing of problematic terms aimed at raising awareness and engendering discussion and understanding. We believe this represents an important contribution to evidence gathering.

A major limitation of the catalogue as presented is, of course, a lack of formal evaluation of its contents. Rather, we have made a case based on argument and illustrative example. However, the catalogue is part of a larger model that is currently subject to refinement as part of an accepted process for developing models, i.e. where an initial model is created based in a pragmatic way and then undergoes ongoing evaluation and refinement until stabilisation occurs (Routio, 2007). We are currently implementing an industry-focused project to deepen our understanding of software development context and would expect both framework and catalogue to be modified and/or extended according to findings.

6 SUMMARY

Software developers do not implement software processes as-is, but rather adapt these according to specific project circumstances. This means we must understand the relationships between specific practices and contextual factors. Earlier investigations revealed that a large number of contextual factors claimed as being relevant for tailoring are unhelpful in that the terms applied cannot be used as-is, but must be understood more deeply. Some terms represent high level factors that affect decisions about operational strategy and objectives but only indirectly affect practice efficacy at the project level. Others are vague in that they can be split into several more basic factors or are ambiguous in meaning. For example, the use of vague terminology in the software engineering literature has been observed in the area of Global Software Engineering (GSE), where a lack of definition of terms such as 'outsourcing' and 'offshoring' causes confusion in meaning which results in an "inability to judge the applicability and thus transferability of the research into practice" (Šmite et al., 2014).

In this paper, we have catalogued an example set of these unhelpful terms and suggested some reasons for their requiring further attention before use. The main contribution is to expose the plethora of unhelpful terms commonly stated as 'contextual factors'. The objective is to support discussion by providing an illustrative set of terms that we suggest cannot be directly applied for understanding situated software practices. We hope the outcome will be increased clarity. This study is part of a set of studies in which we will evaluate and refine the model presented in Section 4. We would expect the catalogue presented in this paper will be modified and/or extended as new terms and understandings arise.

REFERENCES

Avison, D. and Pries-Heje, J. (2008). Flexible information systems development: Designing an appropriate methodology for different situations. In Filipe, J., Cordeiro, J., and Cardoso, J., editors, *Enterprise information systems : 9th International Conference, ICEIS 2007*, pages 212–224, Berlin, Heidelberg. Springer.

Börstler, J., Störrle, H., Toll, D., van Assema, J., Duran, R., Hooshangi, S., Jeuring, J., Keuning, H., Kleiner, C., and MacKellar, B. (2018). "i know it when i see it" perceptions of code quality: Iticse '17 working group report. In *Proceedings of the 2017 ITiCSE Conference on Working Group Reports*, ITiCSE-WGR '17, page 70?85, New York, NY, USA. Association for Computing Machinery.

Clarke, P., Mesquida, A.-L., Ekert, D., Ekstrom, J., Gornostaja, T., Jovanovic, M., Johansen, J., Mas, A., Messnarz, R., Villar, B. N., O'Connor, A., O'Connor, R. V., Reiner, M., Sauberer, G., Schmitz, K.-D., and Yilmaz, M. (2016). An Investigation of Software Development Process Terminology. volume 609 of *Communications in Computer and Information Science (CCIS)*, pages 351–361. Springer International Publishing, Switzerland.

Clarke, P. and O'Connor, R. V. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology*, 54:433–447.

Creswell, J. W. (2014). *The Selection of a Research Approach*, pages 31–55. Sage Publications Inc.

de Azevedo Santos, M., de Souza Bermejo, P. H., de Oliveira, M. S., and Tonelli, A. O. (2011). Agile practices: An assessment of perception of value of professionals on the quality criteria in performance of projects. *Journal of Software Engineering and Applications*, 4:700–709.

Dingsøyr, T. and Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*, 77:56–60.

Dybå, T., Sjøberg, D. I., and Cruzes, D. S. (2012). What Works for Whom, Where, When and Why? On the Role of Context in Empirical Software Engineering. In *Proceedings of the 6th International Symposium on Empirical Software Engineering and Measurement (ESEM 2012)*, pages 19–28, Lund, Sweden.

Easterbrook, S., Singer, J., Storey, M., and Damian, D. (2008). Selecting empirical methods for software engineering research. In F. Shull and J. Singer and D.I.K Sjøberg, editor, *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer International Publishing, London, UK.

Institute of Electrical and Electronic Engineers. (1990). Std 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology. In *IEEE Standards Collection - Software Engineering*. The Institute of Electrical and Electronic Engineers, Inc., New York, USA.

Kirk, D. and MacDonell, S. G. (2016). An Ontological Analysis of a Proposed Theory for Software Development. In et al., P. L., editor, *Software Technologies*

- *ICSOFT 2015*, volume 586 of *Communications in Computer and Information Science (CCIS)*, pages 1–17. Springer International Publishing, Switzerland.

Kirk, D. and MacDonell, S. G. (2018). Evolving a Model for Software Process Context: An Exploratory Study. In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 18)*, pages 296–303, Porto, Portugal. SCITEPRESS.

Klünder, J., Karajic, D., Tell, P., Karras, O., Münkel, C., Münch, J., MacDonell, S. G., Hebig, R., and Kuhrmann, M. (2020). Determining Context Factors for Hybrid Development with Trained Models. In *ICSSP '20, Proceedings of the 2013 International Conference on Software and System Processes*, pages 61–70. Association for Computing Machinery.

Kuhrmann, M. and Münch, J. (2019). SPI is Dead, isn't it? Clear the Stage for Continuous Learning! In *ICSSP '19, Proceedings of the 2019 International Conference on Software and System Processes*, pages 9–13, Montréal, Canada. Association for Computing Machinery.

MacCormack, A., Crandall, W., Henderson, P., and Toft, P. (2012). Do you need a new product-development strategy? *Research Technology Management*, 55(1):34–43.

Müller, S. D., Kræmmergaard, P., and Mathiassen, L. (2009). Managing Cultural variation in Software Process Improvement: A Comparison of Methods for Subculture Assessment. *IEEE Transactions on Engineering Management*, 56(4):584–599.

Munezero, M., Yaman, S., Fagerholm, F., Kettunen, P., Mäenpää, H., Mäkinen, S., Tiihonene, J., Riungu-Kalliosaari, L., Tuovinen, A.-P., Oivo, M., Münch, J., and Männistö, T. (2017). *Continuous Experimentation Cookbook*. DIMECC Oy, Helsinki, Finland.

Orlikowski, W. (2002). Knowing in Practice: Enabling a Collective Capability in Distributed Organizing. *Organization Science*, 13(3):249–273.

Petersen, K. and Wohlin, C. (2009a). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software*, 82:1479–1490.

Petersen, K. and Wohlin, C. (2009b). Context in Industrial Software Engineering Research. In *Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement (ESEM 2009)*, pages 401–404, Orlando, Florida. The Institute of Electrical and Electronic Engineers, Inc.

Routio, P. (2007). Models in the Research Process. http://www2.uiah.fi/projects/metodi/177.htm.

Stol, K.-J. and Fitzgerald, B. (2015). Theory-oriented software engineering. *Science of Computer Programming*, 101:79–98.

Stuckenberg, S. and Heinzl, A. (2010). The Impact of the Software-as-a-Service concept on the Underlying Software and Service Development Processes. In *Proceedings of the 2010 Pacific Asia Conference on Information Systems (PACIS 2010)*, pages 1297–1308.

Turner, R., Ledwith, A., and Kelly, J. (2010). Project management in small to medium-sized enterprises: Matching processes to the nature of the firm. *International Journal of Project Management*, 28:744–755.

Šmite, D., Wohlin, C., Galviņa, Z., and Priladnicki, R. (2014). An empirically based terminology and taxonomy for global software engineering. *Empirical Software Engineering*, 19:105–153.

Zachman, J. A. (2009). Engineering the Enterprise: The Zachman Framework for Enterprise Architecture. http://www.zachmaninternational.com/index.php/the-zachman-framework.