Developing a Taxonomy for Software Process Context

Diana Kirk[1] and Jim Buchan[2]
[1]*Technology Academy, EDENZ Colleges, 85 Airedale Street, Auckland 1010, New Zealand*
[2]*Software Engineering Laboratory (SERL), Auckland University of Technology (AUT), Private Bag 92006, Auckland 1142, New Zealand*

Keywords: Software Process Context, Taxonomy, Evidence Accumulation.

Abstract: When developing software intensive products, practitioners adapt software practices to suit their specific environment. In order to evaluate and compare practices in an evidence based way, researchers must report the context in which the practice was enacted. This is problematic, as the discipline lacks an agreed classification structure for software context. In this paper, we re-position earlier investigations into software context for the purpose of practice evaluation by mapping the evolved framework as a taxonomy. The purpose of the taxonomy is to support discussions about situated software practices with a view to guiding researchers in the specification of context. We conducted an initial evaluation by classifying existing context structures into the taxonomy, and by implementing a small trial study. In future work, we will refine the taxonomy in conjunction with software researchers and practitioners, and use the taxonomy for evidence accumulation.

1 INTRODUCTION

There are many different approaches to creating computer software, each comprising a set of specified *practices*. The need to more deeply understand the relationships between a practice and the context in which it is enacted is a pressing one. This need arises from the established fact that practitioners do not follow software methodologies as architected, but rather adapt these to suit the project environment (Avison and Pries-Heje, 2008; MacCormack et al., 2012). Lengnick-Hall and Griffith identify risks in such ad-hoc practice adaptation. They suggest that applying knowledge in an intuitive or experimental way introduces a lack of fit between type of knowledge and how it is applied. The implication is that many organisations are functioning with reduced effectiveness at best (Lengnick-Hall and Griffith, 2011).

If we are to mitigate the risks inherent in adaptation, it is clear that empirical studies investigating software practices must specify the study's context (Avison and Pries-Heje, 2008; Fitzgerald, 1997; Hansson et al., 2006; MacCormack et al., 2012). This is problematic because our understanding of the factors that must be included in context is immature. Thus, researchers who carry out formal experiments are unable to confidently interpret the scope of applicability of their results because the role of contextual factors is insufficiently understood (Basili et al., 1999; Carver et al., 2004; Runeson et al., 2014; Sjøberg et al., 2005). The goal of amalgamating studies to achieve a more holistic understanding of a technique is compromised by a lack of contextual information (Basili et al., 1999; Kitchenham et al., 2002). The number of possible contexts is immense, perhaps infinite. The problem is exacerbated by the informal use of terms within the industry. For example, it is extremely difficult to establish in an exact way what commonly-used terms such as 'agile' and 'global software development' (Šmite et al., 2014) actually *mean*. In order to address these issues, the problem space must be abstracted in a way that is both meaningful and useful. A *taxonomy* supports classification of items according to stated characteristics, i.e. meaning, and thus is suitable for abstraction. In this paper, we present the first phase of the development of a taxonomy for software process context. The taxonomy represents a re-positioning of our earlier work (Kirk and MacDonell, 2014a; Kirk and MacDonell, 2016), where we evolved a candidate framework for software process context. The aim was to complete the exploratory, model-building phase of a three-phase effort to conceptualise, refine and apply a framework for context (Routio, 2007) i.e. we aimed to create "a starting point (e.g. a framework) that identifies aspects of a topic" (Stol and Fitzgerald, 2015). As this

Kirk, D. and Buchan, J.
Developing a Taxonomy for Software Process Context.
DOI: 10.5220/0006885303120319
In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pages 312-319
ISBN: 978-989-758-320-9

phase involved both empirical observation and human experience, we adopted a research world view of *pragmatism* (Creswell, 2014). The pragmatic viewpoint considers theories as "the products of a consensual process ... to be judged for their utility" (Easterbrook et al., 2008). We intend that our proposed taxonomy, based on the candidate framework, will next enter the refinement phase in conjunction with researchers and practitioners (Routio, 2007) Key understandings from our previous research are a) terms commonly named in the literature as describing context in fact relate to different *kinds* of context, and b) we need to be really clear about scoping. For a), we found that factors named as contextual related to *organisation level strategies* (for example, 'globalise'), *project objectives* (for example, 'user acceptance'), aspects of the *process* (for example 'tool support') and *local operational context* (for example, 'developer experience'). We also exposed that many named terms are simply not sufficiently detailed. For example, 'user participation' may mean the user helped in requirements definition, is available throughout the project, carried out beta testing, etc. 'Non-colocated team' may mean testers are in a different country or the team is split between two locations in the same building, etc. 'Company size' may involve all of constraints on local process choices, staff satisfaction levels and locational organisation. Each meaning has different connotations for practice tailoring. For b), we understood that, for evaluation of practices, the focus must be on *local operational context* i.e. we are always interested in local effect. A factor such as 'globalise' certainly may have an impact on the project, but in an *indirect* way, for example, by causing teams to be set up remotely or placing external constraints on the project. These then become the local context for the project. We applied the taxonomy development method proposed by Nickerson et al. (Nickerson et al., 2013) and mapped the taxonomy to the design structure suggested by Usman et al. (Usman et al., 2017). Our contribution is an initial taxonomy for software process context for the purpose of supporting discussions about situated software practices. In section 2, we overview related work. In section 3 we present the taxonomy design and in section 4 we define the taxonomy. In section 5, we discuss initial evaluation efforts. In section 6, we summarise the paper and discuss future work.

2 RELATED WORK

There are two areas of related work for this paper, research to categorise context and taxonomy design.

2.1 Categorising Context

Many researchers have proposed frameworks that categorise software contextual factors along various dimensions. We overview a selection here.

Avison and Pries-Heje aimed to support selection of a suitable, project-specific methodology (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, its scope is 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 framework again relates to semantically inconsistent classifications. 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 would suggest different kinds of practice.

Petersen and Wohlin provide a checklist for representing context for the purpose of aggregating studies in industrial settings (Petersen and Wohlin, 2009). 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, there appear to be missing contexts, for example, relating to cultural mis-matches between and within teams.

2.2 Developing Taxonomies

The main related area concerns taxonomy development within the industry. For reasons of space, we limited our coverage. However, Usman et al. effectively summarise the work in this area.

Usman et al. carried out a systematic mapping study of taxonomies in SE and applied insights to revise an earlier taxonomy development approach (Usman et al., 2017). Discovered taxonomies spanned the software life-cycle, software management and supporting processes. Context for software practices was

not represented. Our main interest in this work was the revised structure for developing taxonomies. We apply this structure in section 3.

Nickerson et al. proposed a method for taxonomy development (Nickerson et al., 2013). Key aspects of the method include the need for a Design Science approach i.e. iteration with specified ending conditions, determination of the over-riding characteristic to be applied for classification based on the expected purpose and user base, and support for different development approaches. The authors state the need for providing insight into the nature of the items to support explanation and warn against a purely descriptive classification. We follow this advice by aiming for clarity of conceptual meaning in the categories of our proposed taxonomy, and by planning an iterative refinement as future work.

Britto et al. extended an earlier taxonomy for global software engineering (GSE) (Britto et al., 2016). Extensions address the recognition that cultural and process-related characteristics are relevant in a GSE setting. This work is highly relevant as GSE represents a subset of the space of software engineering and this means our taxonomy should be applicable for GSE projects. We address this in future work.

3 TAXONOMY DESIGN

The term 'taxonomy' was originally applied to the naming and classification of organisms in the natural sciences. The term has more recently been applied in a broader sense. We adopt the definition of "... classification according to a predetermined system, with the resulting catalog used to provide a conceptual framework for discussion..." (TechTarget, nd). The rules for inclusion in a category must be clearly stated i.e. mutually exclusivity is indicated. Categories ideally span the problem space.

Nickerson et al., state that the identification of meta-characteristics for the taxonomy is a key issue. The researcher should not examine a "large number of related and unrelated characteristics ... in the hope that a pattern will emerge" (Nickerson et al., 2013). Meta-characteristics should be based on the purpose of the taxonomy and this requires identification of the expected users. The authors describe two development methods. In the *conceptual* approach, dimensions are conceptualised by the researcher. The *empirical* approach involves selecting a subset of items and establishing common characteristics.

We expect SE researchers and practitioners to use the taxonomy when discussing situated software practices. Our meta-characteristic is *contextual meaning*. As we are re-positioning an existing framework, developed under a *pragmatic* world view (Routio, 2007), our approach is *conceptual*.

3.1 Design Mapping

Usman et al. proposed a taxonomy development method for software engineering. The approach has thirteen activities, shown in the first three columns of table 1. The last column shows our mapping.

3.1.1 Planning

The SE knowledge area for the proposed taxonomy is *software process* (A1). The objectives of the taxonomy are to support investigations into, and discussions about, situated software practices (A2). The scope thus includes all situational factors that may affect practice efficacy. The subject matter of the taxonomy is *context for situated software practices* (A3).

Classification structures are informed by studies into the role of classification in knowledge representation. Kwasnik suggests that "a good classification functions in much the same way as a theory does, connecting concepts in a useful structure" (Kwasnik, 1999). We summarise the options below.

Hierarchies: are a common form of representing knowledge in fields that have theoretical foundations such as biological evolution. The characteristic relationship between entities is the *is-a* relationship, with its connotations of inheritance, transitivity and mutual exclusivity.

Trees: commonly capture a *part/whole* relationship. In this representation, sibling entities may have little in common, but rather the tree "lays out the entities comprising a domain in a pattern that ... makes evident the important or defining relationships among them" (Kwasnik, 1999).

Paradigms: are appropriate for describing entities by the intersection of two attributes.

Faceted analysis: adopts the viewpoint that a domain is constantly changing and must be viewed from several perspectives. The example provided by Kwasnik is that of *material culture* with one entity described as "19th century, Japanese, ceramic, vase". Items are thus described by concatenating the description for each facet and this means that new items may appear with as-yet unused combinations of facet values (for example, "19th century, American, ceramic, vase").

The initial thought would be that, as *context* has many aspects with a huge number of possible variations, a facet-based classification would be most ap-

Table 1: Taxonomy development activities - Usman et al.

Phase	*Activity*	*Description*	*Kirk et al.'s taxonomy*
Planning	A1	Define SE knowledge area	Software process
	A2	Describe the objectives of the taxonomy	Support discussions about situated practices.
	A3	Describe the subject matter to be classified	Context for situated software practices.
	A4	Select classification structure type	Tree
	A5	Select classification procedure type	Qualitative
	A6	Identify the sources of information	SE literature
Identification, extraction	A7	Extract all items	Theory+data mapping+pragmatism
	A8	Perform terminology control	Abstraction
Design, construction	A9	Identify and describe taxonomy dimensions	Software initiative
	A10	Identify and describe dimension categories	See figure 1, table 1 and section 4
	A11	Identify and describe the relationships	Part-whole
	A12	Define the guidelines for using the taxonomy	See Appendix
Validation,	A13	Validate the taxonomy	See section 5

propriate. However, the goal for the taxonomy is not to provide a means of representing every possible contextual factor, but is rather to abstract the information in a meaningful way. Our taxonomy is best described as a *Tree* (A4). The abstraction process we applied is *qualitative* in nature (A5). The taxonomy is based on concepts from the SE literature (A6).

3.1.2 Identification and Extraction

Extraction of terms from the source data was achieved by a mix of theory, data mapping and pragmatism (Kirk and MacDonell, 2014b; Kirk and MacDonell, 2014a; Kirk and MacDonell, 2016). The initial structure was based on early observations that tailoring requires consideration of goals and environment (Basili and Rombach, 1987), with environment characterised by a set of mutually exclusive dimensions (conceptual approach). As terms from the literature were mapped, issues were encountered, and the result was a rethinking of the dimensions and the introduction of new categories. For example, many commonly used terms were found to be ambiguous with regard to contextual meaning. Such terms must remain in the taxonomy if we are to successfully discuss context with practitioners and researchers, and so a new category, 'ambiguous', was formed (A7). All found terms were abstracted into one of the conceptual categories (A8).

3.1.3 Design and Construction

The top level of the taxonomy is the *Software initiative* (A9). The structure is presented in figure 1 and table 2 and structure elements are described in section 4 (A10). Relationships are part-whole. For example, a *Software Initiative* comprises some *Objectives*, an *Operational Context*, a *Strategic Context* and a *Process Solution* (A11). Instructions for use by researchers establishing context for situated software practices are shown in the Appendix (A12).

3.1.4 Testing and Validation

See section 5 (A13).

4 TAXONOMY DESCRIPTION

In this section, we describe the categories included in the taxonomy.

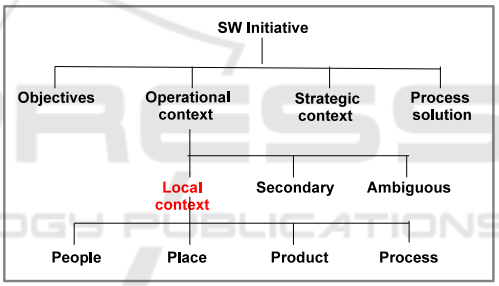

Figure 1: Taxonomy top level.

The taxonomy is depicted in figure 1 and table 2.

4.1 Software Initiative

A *Software initiative* is any endeavour that involves defining, creating, delivering, maintaining or supporting software intensive products or services. It thus encompasses the more recent client-focussed delivery paradigm and subsumes the traditional 'project'.

4.2 Objectives, Strategic Context, Process Solution

Objectives represent the goals of the initiative at an operational level. For example, as the result of a strategic decision, a project manager may be tasked with delivering a product within a short time-frame. (S)he may also be expected to keep staff happy i.e. a single initiative may have several objectives.

Table 2: Local operational context factors.

People	Entity	Capability	High ... Low
		Motivation	High ... Low
		Empowerment	High ... Low
		Cultural cohesion	High ... Low
	Interface	Cultural cohesion	High ... Low
Place	Entity	Physical distance	Same room, same blg, same city, same country, different country
		Temporal distance	Same time zone, <4 hrs overlap, <6 hrs overlap, >= 6 hrs overlap
		Availability	High ... Low.
	Interface	Physical distance	Same room, same blg, same city, same country, different country
		Temporal distance	Same time zone, <4 hrs overlap, <6 hrs overlap, >= 6 hrs overlap
		Availability	High ... Low.
Product	Product type		*Safety critical; embedded; developer tools.*
	Lifecycle stage		NPD, adolescence, maturity, old-age, in-retirement
	Standards		*Safety; security; licencing.*
	Requirements		*Well-understood; changing/emergent; conflicting; missing.*
	Implemented		*Consistent; complete; quality; complexity.*
External Process	Client		*Specification; delivery.*
	Parent org		
	Legal		
	Financial		

Strategic context includes the many factors that require decision makers to manipulate the Objectives or Operational context for the initiative. Examples are the organisation's need to gain consumer trust, or to expand into a global marketplace. The first may result in a project where product quality is stated as the key objective. The second may result in the establishment of off-shore teams (operational context). In both cases, the affect on tailoring is indirect.

Process solution represents the set of practices and techniques enacted within the initiative to meet the stated objectives. This is the element to be tailored according to Objectives and Operational context.

4.3 Operational Context

These are the contextual factors that apply at the operational level and that are fixed for the initiative. Of course, a factor that is fixed for one initiative (e.g. using an external test team) may be under project management control for another.

4.3.1 Secondary, Ambiguous

Secondary factors are factors that comprise multiple ideas. An example is 'outsourcing', which has many possible scenarios relating to *what* is outsourced by *whom*, *where*.

Ambiguous factors require deeper consideration. An example is 'uncertain requirements', which may exist because the client is unclear about what is wanted, client processes result in delays before requirements decisions can be made, or the relevant client isn't available. Each of these meanings has different connotations for practice tailoring.

4.3.2 Local Context

These are the operational contextual factors that are directly applicable i.e. are the real factors of interest for process adaptation. The subcategories are *People*, *Place*, *Product* and *External Process*.

People: relates to cultural characteristics which affect how well a team performs. Identify the culturally-disparate *entities* (for example, agile team, analysts, client) and the *interfaces* between entities. For each entity, establish

- capability - ability to perform.
- motivation - desire to perform well.
- empowerment - degree of autonomy.
- cultural cohesion - degree of shared understanding within the entity.

For each interface, we establish *cultural cohesion*. For example, the shared understanding between client and development team can be low.

Place: addresses the availability of people, affecting practices relating to logistics and communication. For all entities and interfaces, we establish

- physical distance.
- temporal distance.
- level of availability.

Product: relates to characteristics of the product that may impact practice selection. By *product*, we mean either the product to be built or the product-as-is. Characteristics include

- product type (e.g. life-critical, embedded).
- maturity (for example, new product, stable).

- standards that impact the product.
- characteristics of product definitions (for example, conflicting requirements) .
- characteristics of as-implemented product (for example, consistency between representations).

Process: addresses constraints due to processes external to the software initiative. Sources include

- client (for example, delivery expectations).
- parent organisation (for example, cultural expectations on process).
- legal constraints (for example, licencing) .
- financial constraints (e.g. on tools).

Italicised and missing items in the last column of table 2 represent unfinished branches. In some cases, we are not confident that the sub-categories shown exhibit orthogonality. In other cases, we believe the branch is incomplete. These are areas for future work.

5 EVALUATION

Development of a taxonomy is expected to be iterative (Nickerson et al., 2013). Approaches to demonstrating utility include illustration (examples, scenarios and cases), case studies, experiments and expert opinion (Usman et al., 2017). Šmite et al. suggest that a taxonomy can be validated by "demonstrating orthogonality of its dimensions, benchmarking against existing classifications and ... (classifying) existing knowledge". Britto et al. demonstrated utility of an extended GSE taxonomy by classifying existing GSE projects (Britto et al., 2016).

As a first iteration, and initial evaluation, we classified two context structures from the literature (Avison and Pries-Heje, 2008; Petersen and Wohlin, 2009) and implemented a small, informal trial study.

5.1 Classifying Existing Structures

In tables 3 and 4, we show the results of classifying the elements of the target structures (columns 1 and 2) into our taxonomy (column 3). We first note that the source elements represent different kinds of meaning. For example, *Quality* in table 4 classifies as an *Objective* and *Time critical* in table 3 as *Strategic context* (as decisions concerning scope, manpower, etc. must be made). Many elements classify as *Ambiguous* i.e. cannot be used for practice tailoring without further investigation. For example, *Language* in table 4 may mean 'the language the product is coded in affects developer capability' or 'there is an external constraint on the language to be used for coding'.

Table 3: Classification - Avison and Pries-Heje.

Task	Large	Prod:Req
	Unclear	Prod:Req
	Complex	Prod:Req
Knowledge	Domain	Pple:Ent:Capablty
	Development	Pple:Ent:Capablty
	Tools	Pple:Ent:Capablty
Individual	Project experience	Pple:Ent:Capablty
	Forced into project	Pple:Ent:Motivn
	Part time	Place:Ent:Avail
Environmt	Far apart	Secondary
	Interruptions	Ambiguous
	Spacial conditions	Secondary
Team	Large,backgrounds	Pple:Ent:Cohesn
	Personality mix	Pple:Ent:Cohesn
	Just met	Pple:Ent:Cohesn
	Calendar time	Strategic context
	Time critical	Strategic context
Stakehldrs	Conflicts	Pple:IF:Cohesn
	Many	Ambiguous
	Mgmnt attentn	Ambiguous
	Unclear decisions	Ambiguous
Criticality	Life threatening	Prod:Type
	Process constrained	Process
	User needs unknown	Prod:Req
	Complex req.	Prod:Req

Table 4: Classification - Petersen and Wohlin.

Product	Maturity	Product:Lifecycle
	Quality	Objectives
	Size	Product:Req.
	System type	Product:Type
	Customisation	Product:Type
	Language	Ambiguous
Processes	Activities	Ambiguous
	Work-flow	Ambiguous
	Artifacts	Ambiguous
Practices	CASE tools	Ambiguous
	Techniques	Ambiguous
People	Roles	Pple:IF:Cohesn
	Experience	Pple:Ent:Capablty
Organisation	Org model	Secondary
	Org unit	Ambiguous
	Certification	Ambiguous
	Distribution	Secondary
Market	Num customers	Process:Client
	Mkt segment	Strategic context
	Strategy	Strategic context
	Constraints	Strategic context

We further observe that some of our taxonomy categories are not represented. Examples are product lifecycle stage in table 3 and the people motivation and empowerment categories in table 4. It is possible that these categories are covered by the elements classified as *Ambiguous* and *Secondary*.

5.2 Trial Study

Participants (industry practitioners) were provided with instructions (see Appendix), to be read in conjunction with the definitions above. In summary, they were asked to

- Select a specific practice from a recent software initiative and establish objectives.
- Identify factors believed to have contributed towards success or failure.
- Classify the factors into the taxonomy.

Several practitioners were positive about the possibility of a decision support system to help with practice selection. However, most found it difficult to align their thinking with the categories of the taxonomy. This is not surprising, as thus far there appears to have been little thought given to exactly what is meant by 'context', with the result that different kinds of factor tend to be viewed in a generic way. This caused us to understand that the taxonomy is primarily a tool for *researchers*, at least until a common terminology has been established.

One researcher, with expertise in the area of human aspects in agile projects, felt that the terms 'cultural cohesion' and 'shared understanding' in the 'People' category had two different meanings. He suggested the term 'team cohesion' was a more appropriate one. He also suggested the addition of the term 'willingness to change' as having a different meaning than 'motivation' i.e. both are required as basic ideas.

6 SUMMARY

In this paper, we have proposed a taxonomy for software process context. The taxonomy represents a repositioning of our earlier investigations and our contribution is a preliminary conceptualisation of context to support discussion and evidence accumulation. We applied the taxonomy development method proposed by Nickerson et al. (Nickerson et al., 2013) and mapped the taxonomy to the design structure suggested by Usman et al. (Usman et al., 2017). The main limitation of this contribution is that, at present, the taxonomy is in the *conceptual* stage and so evaluation thus far is minimal. We classified two existing context models into the taxonomy and conducting a small industry trial. In the next stage of our research, we will formally and iteratively refine the taxonomy (Nickerson et al., 2013; Routio, 2007) in collaboration with researchers and practitioners.

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, *ICEIS 2007*, pages 212–224. Springer.

Basili, V. R. and Rombach, H. D. (1987). Tailoring the Software Process to Project Goals and Environments. In *Proc. Ninth International Conf. on SW Engineering*, pages 345–357. IEEE.

Basili, V. R., Shull, F., and Lanubile, F. (1999). Building Knowledge through Families of Experiments. *IEEE Trans. on Software Engineering*, 25(4):456–473.

Britto, R., Wohlin, C., and Mendes, E. (2016). An extended global software engineering taxonomy. *Jrnl. Software Engineering Research and Development*, 4:3.

Carver, J., Voorhis, J. V., and Basili, V. (2004). Understanding the Impact of Assumptions on Experimental Validity. In *Proc. ISESE'04*, pages 251–260. IEEE.

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.

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.

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.

Fitzgerald, B. (1997). The use of systems development methodologies in practice: a field study. *Information Systems Journal*, pages 201–212.

Hansson, C., Dittrich, Y., Gustafsson, B., and Zarnak, S. (2006). How agile are industrial software development practices? *Jnl. Systems and Software*, 79:1295–1311.

Kirk, D. and MacDonell, S. G. (2014a). Categorising software contexts. In *Proceedings of 20th Americas Conference on Information Systems, AMCIS 2014*.

Kirk, D. and MacDonell, S. G. (2014b). Investigating a conceptual construct for software context. In *Proceedings of the Conference on Empirical Assessment in Software Engineering (EASE)*, number 27.

Kirk, D. and MacDonell, S. G. (2016). An Ontological Analysis of a Proposed Theory for Software Development. In *Software Technologies - ICSOFT 2015*, volume 586 of *CCIS*, pages 1–17. Springer Intl.

Kitchenham, B. A., Pfleeger, S. L., Hoaglin, D. C., El Emam, K., and Rosenberg, J. (2002). Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Trans. on SW Eng.*, 28(8):721–734.

Kwasnik, B. H. (1999). The role of classification in knowledge representation and discovery. *Library Trends*, 48.

Lengnick-Hall, C. A. and Griffith, R. J. (2011). Evidence-based versus tinkerable knowledge as strategic assets: A new perspective on the interplay between innovation and application. *Journal of Engineering and Technology Management*, 28:147–167.

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.

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.

Nickerson, R. C., Varshney, U., and Muntermann, J. (2013). A method for taxonomy development and its application in information systems. *European Journal of Information Systems*, 22:336–359.

Petersen, K. and Wohlin, C. (2009). Context in Industrial Software Engineering Research. In *Proc. ESEM 2009*, pages 401–404, Orlando, Florida. IEEE.

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

Runeson, P., Stefic, A., and Andrews, A. (2014). Variation factors in the design and analysis of replicated controlled experiments. *Empirical Software Engineering*, 19:1781–1808.

Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N.-K., and Rekdal, A. C. (2005). A Survey of Controlled Experiments in Software Engineering. *IEEE Transactions on Software Engineering*, 31(9):733–753.

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 *Proc. PACIS 2010*, pages 1297–1308.

TechTarget (n.d.). Definition - Taxonomy.

Usman, M., Britto, R., Börstler, J., and Mendes, E. (2017). Taxonomies in software engineering: A Systematic mapping study and a revised taxonomy development method. *Inf. and SW Technology*, 85:43–59.

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.

APPENDIX

Scope the Study

- Select a recent or current software initiative (for example, a project).
- Select a specific practice from the initiative.
- Establish the objectives for the initiative. Classify as 'Objectives'. Note that these are the local, operational objectives.

Classify named factors

For the selected software initiative and practice, ask the practitioner (or ascertain from the literature) the factors believed to have been important in the success or failure of the practice in meeting one of the objectives. For each named factor, establish:

- If the factor refers to an organisation level strategy, (for example, 'increase market share'), classify as 'Strategic context'. Discuss with the practitioner what this means locally for the initiative and identify any local factors.
- If the factor represents a local objective, classify as 'Objectives'.
- If the factor describes the attributes of a process or practice, classify as 'Process solution'. Some examples are 'tool support', 'agile process'.
- If the factor is believed to directly affect the efficacy of the named practice in meeting one of the objectives, classify as 'Operational Context' and continue with *classify operational factors*, below.
- If the factor does not classify as any of the above, report back to the taxonomy owner, as this may indicate a gap in the taxonomy.

Classify operational factors

An 'Operational Context' factor must be classified into one of the four base dimensions ('People', 'Place', 'Product', 'Process') and then sub-classified according to the structure shown in Table 2. Elements with italicised or missing values in the last column represent 'unfinished' elements.

- If the factor does not have a clear classification according to the descriptions in section 4, analyse it as a possible secondary or ambiguous factor, and classify as such.
- Classify the factor according to table 2. For elements with uncertain values, ascertain any values that are believed to be relevant, using the examples to support identification.
- Analyse all secondary and ambiguous factors in conjunction with the practitioner, and add emerging factors to the list of factors to be analysed.
- Create two diagrams for the selected initiative, objective and practice, one for the context that supported practice efficacy, and the other for the context that was detrimental to practice efficacy. In each, show values for all context leaf nodes that have emerged during the analysis.