This is the published version of a paper published in *Safety Science*.

Citation for the original published paper (version of record):

Asplund, F. (2020)
Conflict as software levels diversify: Tactical elimination or strategic transformation of practice?
*Safety Science*, 126: 104682
https://doi.org/10.1016/j.ssci.2020.104682

Access to the published version may require subscription.

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-269100
Conflict as software levels diversify: Tactical elimination or strategic transformation of practice?

Fredrik Asplund⁎, Greg Holland⁎, Saleh Odeh

⁎ KTH Royal Institute of Technology, Department of Machine Design, Division of Mechatronics, Brinellvägen 83, 10044 Stockholm, Sweden
b Rolls-Royce plc, PO Box 31, Moor Lane, Derby DE24 8BJ, United Kingdom

ARTICLE INFO

Keywords:
Communities of practice
Safety standards
Software levels
Standardization

ABSTRACT

Communities of Practice create a shared consensus on practice. Standards defining software levels enable firms to diversify practice based on a software component’s contribution to potential failure conditions. When industrial trends increase the importance of lower software levels, there is a risk that the consensus on practice for software engineers used to primarily working at higher levels of assurance is eroded. This study investigates whether this might lead to conflict and – if so – where this conflict will materialize, what the nature of it is and what it implies for safety management.

A critical case study was conducted: 33 engineers were interviewed in two rounds. The study identified a disagreement between designers with different roles. Those involved in the day-to-day activities of software development advocated elimination of practice (dropping or doing parts less stringently), while those involved in expert advice and process planning suggested transforming practice (adopting realistic alternatives).

This study contributes to practice by showing that this conflict has different implications for firms that do not lead vs those that lead the early adoption of technology. At the majority of firms, safety management might need to support the organisation of informal opinion leaders to avoid vulnerability. At early adopters, crowdsourcing could provide much-needed help to refine the understanding of new practice. Across entire industries, crowdsourcing could also benefit entire engineering standardization processes. The study contributes to theory by showing how less prescriptive standardization in the context of engineering does not automatically shift rule-making towards allowing engineers to act more autonomously.

1. Introduction

Cyber-Physical Systems (CPS), enabling interaction with physical processes through information technology, have become well established in application domains such as healthcare, transportation, energy and manufacturing (Törngren, et al., 2017). Despite efforts to simplify their engineering, entering the CPS market still requires expertise in a wide set of engineering disciplines (Geisberger, et al., 2015). Safety engineering is frequently one of these disciplines, since safety is often a critical characteristic of CPS. This expertise is codified in standards that directly or indirectly provide guidance on ensuring safety, such as DO-178C (aerospace software) (RTCA Inc, 2011), ISO 26262 (automotive) (International Organization for Standardization, ISO, 2011), IEC 60987 (nuclear hardware) (International Electrotechnical Commission, 2007), ECSS-Q-ST-40C (space safety assurance) (ESA-ESTEC, 2009) and EN 50129 (rail electronics) (CENELEC, 2003). As it is relatively costly to follow these standards, several of them allow for treating product components differently depending on their impact on safety. As an example, DO-178C recognizes and offers guidance for five software levels based on a software component’s contribution to potential failure conditions. The system safety assessment process determines the software level of a software component by identifying the level associated with the most severe failure condition to which it can contribute (RTCA., 2011). Software levels thus influence technological choices, both in regard to components and system architectures. However, the differences between levels are often found in the methods used in (or goals of) the development processes, rather than the technology employed.

In an engineering context, those implementing software according to software levels organise in various Communities of Practice (CoP), i.e. “groups of interdependent participants [that] provide the work context within which members construct both shared identities and the social context that helps those identities to be shared” (Brown and Duguid, 2001). CoP carry strong implications for their members: they...
introduce new practitioners to established practitioners and others related to their work (Lesser and Storck, 2001), and can develop their own pool of tacit knowledge and associated practices (Leonard and Sensiper, 1998). Software engineering practice thus involves alignment not only through mechanically adhering to processes, but through negotiation of meaning between those engaged in the practice (Wenger, 2010). In other words, the way work tasks are carried out will depend on what the collective of engineers engaged with them believe is required, to ensure the manufacture of safe products, for instance.

There are thus three efforts in the development process that establish the implications of the software level determination of a software component: firstly, the effort to handle the consequences of component failure identified during safety assessment; secondly, the redesign of the product to handle software levels efficiently in regard to both safety and cost; finally, the way work tasks are later influenced during implementation by how software levels are perceived by engineers. Of these, this paper focuses on the way work tasks are influenced by the perception of the engineers. More specifically, it focuses on the changes to this perception as technological trends increase the demand for software components at lower software levels. As the importance of levels that do not necessitate the strictest software assurance thus increase for development organisations, it can challenge the consensus on practice for software engineers used to primarily working at higher levels of assurance. This can lead to changes in the way CoP interact with each other and internally. Engineers might disagree on whether practice needs to change, and – if so – how it should be changed. Such disagreement can lead to a struggle, or conflict, for the right to decide on the future of practice within a firm. An important responsibility of safety management is to be pro-active in regard to such social interactions driven by change. Other organisational values than the wish to increase safety can drive practice, or at least be a source of resistance to such change (Groote, 2012). Unfortunately, standards that influence safety can currently be complex, open to interpretation and even ambiguous (Youn and Yi, 2014; Nair, 2014; Graydon, 2015). This means that even with a thorough understanding of a firm it could be difficult to deduce how firms should safely approach conflict regarding practice tied to different software levels. This motivates a case study into a context where the importance of lower software levels are increasing, challenging the consensus on practice for software engineers used to primarily working at higher levels of assurance. We explore conflicts due to this change and address three associated research questions:

Where do such conflicts materialize?
What is the nature of these conflicts?
Considering these conflicts, what are the implications of change for safety management?

The paper starts by constructing a framework from several academic discourses to serve as an analytical lens for the study. The next section describes our case as that of a large firm with a history of working primarily with highly safety-critical systems, positions the research as a critical case study and details the associated data collection, data analysis and validation concerns. This is followed by a description and summary of the results, clarified by examples. The results are then analysed to interpret the results in relation to our analytical framework. Based on our interpretation, the wider implications of the analysis is evaluated by discussing it in relation to the existing state-of-the-art.

2. Theoretical framework

This section starts by describing the discourses of relevance to the context of the study, i.e. those concerning CoP, software engineering and standards that provide guidance on ensuring safety. Findings from the different discourses are summarized continuously throughout the section. The section ends by using these summaries to construct an analytical framework for the analysis and discussion of the results.
establish what it means to be competent. Experiences by individual members outside of their CoP are brought back into the communities to change this collective understanding of competence. Organisational learning is thus dependent on the ability of individual members to suspend their identities as they cross boundaries between CoP, but also on their ability to negotiate the meaning of competency within their CoP (Wenger, 2000). Change can thus be difficult, especially if it challenges the currently dominant ideas on identity and practice (Roberts, 2006). Successful boundary spanning is a topic of research on its own, for instance in regard to gatekeepers who bring knowledge from external sources into an organisation and ensure that it is both understood and used (Paul and Whittam, 2010). The concept of CoP underlines findings in this discourse that suggest it is not possible to simply assign someone to the gatekeeping role, but rather requires identifying already well connected individuals and improving their networks and skill sets (Nochur and Allen, 1992). As the structure of CoP can be different from the formal structures at a firm, holding a formal position might not ensure that an individual is able to interact in a fruitful way with other experts. The different groups defined by formal and informal organisational structures can even be analysed from a political perspective (Tushman, 1977). The organisation of a firm into distinct parts can, for instance, create its own logic, leading to a self-reinforcing cycle where the flow of information is increasingly controlled (Vince, 2001). The CoP concept has been criticized for not taking such issues into account, focusing more on harmony and homogeneity (Wenger, 2010). This misses the point that informal structures centred on identity are inherently associated with power. One of the most basic implications for power in regard to CoP is that people tend towards homophilous communication, i.e., they favour communicating with those that are similar to themselves (Rogers, 1995). This stems from homophilous communication being more effective and perceived as more rewarding than heterophilous communication. Arguably it will be difficult to communicate new ideas between dissimilar CoP; since their differences imply both a high effort for cross-boundary communication and few connections between the groups. Trying to change practice as agreed by a community of practice might thus be a futile exercise if the right set of boundary spanners are not first persuaded to fall in line.

Not all suggestions on how to change practice that influences safety will thus have equal value. The details of best practice are defined by the practitioners working according to it. A well-known example is how management often form their own communities and might not consider the knowledge held by other CoP to be legitimate enough to merit consideration (Yanow, 2004). However, certain practitioners within a firm can, by force of their networking, be in a position to influence their community – especially if they are in a position to bring in and interpret knowledge from external sources. These experts can thus champion new ideas, and their existence explains how different CoP can influence each other – possibly leading to consensus between communities on cross-organisational issues such as safety. Such agreements between CoP could help identify the informal hierarchies within an organisation.

2.2. Software engineering

There are many high-level perspectives from which the practice of software development can be described, such as those based on knowledge areas (IEEE Computer Society, 2014) and sets of processes for building life-cycles (International Organization for Standardization, ISO, 2008). As the scope of these descriptions indicates, contemporary software engineering practice is large enough to encourage specialization into work functions focused on project management, design, testing, quality assessment, etc. Studies on knowledge management in software engineering have also highlighted the importance of tacit knowledge, that processes as they are performed can differ significantly from how they were formally designed, and the need for informal structures to support learning (Bjørnson and Dingsøyr, 2008). This implies the need for software engineering studies to take proper account of organisational roles in conjunction with their work practice. Unfortunately it is dubious whether this is usually the case. On the one hand, the academic discourse on software engineering has concerned itself mostly with technology and formal process, sometimes recognizing organisational roles but usually ignoring human and organisational factors (Lenberg et al., 2015). In some knowledge areas this is evident in the separate communities of academia and industry, with little evidence being generated by academia on several topics important to contemporary practice (Garousi and Mäntylä, 2016; Garousi and Felderer, 2017). On the other hand, studies on individual characteristics frequently fail to consider the software engineering profession as heterogeneous, focusing rather on distinguishing between software engineering and other professions (Cruz et al., 2015; Beecham, 2008). The software engineering field also includes several high visibility concepts that may hide the importance of organisational and human factors, since they lead to different types of knowledge sharing (Ghobadi, 2015). As an example, agile development is acknowledged to rely more on the interpersonal skills of software engineers, blurring the line between organisational roles as loyalty to the team and organisation takes precedence (Ghobadi, 2015; Hoda et al., 2013). Arguably, more time has been spent on generalizing the difference between agile and traditional development than teasing out their different knowledge-sharing patterns and then generalizing the resulting implications.

The discourse thus suggests that CoP within software engineering can have important implications, but are understudied. This is likely due to software engineers being treated as a homogeneous group and the existence of other important phenomena that affect engineers and engineering processes. The influence of different organisational roles has been noted, however – for instance in regard to how the difference in the practice of software development and testing may lead to communication difficulties (Zhang, 2014). Even though these groups work closely together they may thus form communities with different perspectives on each other’s skills and knowledge (Zhang, 2017). Interestingly enough, in the study by Hoda, Noble and Marshall on self-organizing teams in an agile development where organisational roles are transcended, testers are used both to make the point that the agile methodology needs to be reinforced to remain in place and as an example on how certain “personalities” might have to be removed in order to not hamper the agile methodology (Hoda et al., 2013). There is also evidence of software engineering managers forming a group distinct from other software engineers, for instance in regard to the possibility to codify knowledge (Dingsøyr and Ryrvik, 2003), the attribution of value (Taylor, 2016) and the importance of technical vs social skills (Kalliamvakou, 2017).

There are thus differences in the perspectives and practice of designers, testers and managers engaged in software engineering. This implies that CoP can emerge centred on these organisational roles. Similarly, what limited evidence there is concerning the importance of organisational culture in software engineering seems to carry implications that transcend individuals and development methodologies: livari and Huisman identify how a culture oriented towards stability and internal focus is related to the deployment of systems development methodologies (livari and Huisman, 2007); livari and livari reflect on how organisations with a different orientation towards change vs. stability and internal vs. external focus have different implications for the efficiency of ad hoc, agile and traditional development methods (livari and livari, 2011); and Siakas and Siakas suggest that agile development is suitable for organisations with a democratic culture (Siakas and Siakas, 2007).

Organisational values will thus have an effect on practice across organisational roles in software development. CoP within a firm cannot ignore its organisational values when defining software engineering practice. This suggests that the influence from the wider inter-organisational communities of designers, testers and managers will be interpreted locally in view of how much a firm emphasises safety.
2.3. Safety-relevant standards

Standards fill many roles in the development of advanced technological products, forming an infrastructure that has implications for both technology and economy (Tassey, 2015). Safety standards are of the type meant to specify acceptable product or service performance (Tassey, 2000), but are arguably unique due to their implications: failure to follow their guidance may not only impact the delivery of products but also the well-being of customers. While safety standards provide the same lowering of transaction costs as other standards that specify acceptable performance, they thus also have strong economic implications through liability: they provide a useful template for e.g. technical risk argumentation and best practice for avoiding hazardous errors (Kelly, 2014).

In other words, if a firm has adhered to the way a safety standard defines best practice, it may offer important protection should an accident occur. Safety standards are thus not only a strong moral argument for certain practice, but also an economic one.

However, the standards themselves are codified knowledge, which has to be absorbed and used by an organisation in order to have any impact. While not all safety standards define processes, it is arguably likely that an organisation will make use of safety standards by incorporating their knowledge in its processes. In this way the evidence required by a safety standard is generated as engineers perform their day-to-day duties. We note two implications of this approach related to knowledge handling. Most obviously it means that engineers might only be exposed to the bits and pieces of the safety evidence that relate to their processes. They may thus fail to appreciate the sum of what a safety standard strives for. Less obviously, as safety standards can be complex, open to interpretation or even ambiguous (Youn and Yi, 2014; Nair, 2014; Graydon, 2015), they often do not codify in complete detail the methods they stipulate. This can even be deliberate to allow alternative means of compliance. Internally to each organisation complying with a standard, there will thus be associated tacit knowledge known only to the engineers carrying out the associated methods. Software engineering organisations in particular rely on being able to efficiently share such tacit knowledge across the organisation ( Bjørnson and Dingsøyr, 2008).

Engineers can thus form a selective view regarding the overall intent of the standards. Standards also only provide guidance to a certain level of detail. Ultimately, processes based on the same safety standards might thus be carried out in very different ways, as CoPs have leeway to interpret the guidance on safety.

Furthermore, safety standards frequently come in sets with other standards that are meant to be applied together (Youn and Yi, 2014; Nair, 2014). The "systems" aspect of standards, where several standards together create an overall impact on technology or economy (Tassey, 2000), is thus often relevant to safety standards. We use the term safety-relevant standards to indicate both safety standards and the standards they rely on to offer complete guidance. As an example, DO-178C for aerospace assumes that a set of complete, correct and consistent software requirements are given and offers detailed guidance on how to ensure that these are met (RTCA Inc., 2011). This means that DO-178C is typically applied together with ARP4754A (systems development) (SAE S-18, 2010) and ARP4761 (safety assessment) (SAE S-18, 1996). Combined, they form a large part of the recognized guidelines on how to structure the development of an aircraft to arrive at the evidence necessary for safety certification. Although this complexity and scope often means that safety-relevant standards are seen as incurring a high cost, some of the perceived cost could be avoidable through the use of contemporary software methods and tools (Youn and Yi, 2014; Wong et al., 2011).

Although standards such as DO-178C are not safety standards per se, software engineers thus often perceive them as such and they do indeed contain the guidance related to safety that is important to them. This "system" of standards might seem daunting and overly costly, but it could possibly be handled efficiently through contemporary or new practice.

Furthermore, engineers without deep expertise in applicable standards might over- or under-estimate the limitations these impose on development practices. This might not hamper engineers working with safety-relevant standards that prescribe the methods and processes to apply, as these often stipulate which techniques should be used at different software levels to establish confidence commensurate to the contribution of the software to system risk (Kelly, 2014). However, other, goal-based standards instead set out high level objectives for manufacturers to prove through the submission of a safety argument (Hawkins, 2013). Where goal-based standards are used, safety cases must explicitly justify the claim that the evidence it contains justifies such confidence, perhaps through the use of a separate confidence argument (Hawkins et al., 2011). Such justification can align with the guidance on levels, but will still have to argue for why the confidence established at a particular software level is appropriate. The likelihood of engineers influencing development practices negatively by over- or under-estimating the limitations that safety-relevant standards impose on them is then aggravated by two aspects of these standards: firstly, that the process for creating standards excludes relevant experts from learning through participation (Habli, 2017); secondly, that the rationale behind many safety-relevant standards is implicit (Habli, 2017). Engineering firms will most likely come under increased pressure to handle these aspects to achieve an increased understanding of the rationale underpinning the guidance provided by safety-relevant standards.

In other words, if standardization moves towards goal-based standards in the future, it increases the chance that any argument for the use of software levels will be required to be explicit. This also increases the pressure on engineering firms to establish ways through which their engineers can learn the underlying intent behind the safety-relevant standards they use.

2.4. Analytical framework

This subsection summarizes the most important findings presented so far. It provides a succinct base for understanding our approach to generating and analysing results.

Our analytical framework focuses on the primary CoP that we can expect to encounter in association with software development, i.e. software designers, testers and managers. Internally to firms, members of each community form a consensus on best practice through organisational learning primarily under the influence of two sources: that of the internal informal power structure and that of the entire external corresponding community that spans the software development industries. Standards important to organisational values, such as safety-relevant ones, have implications for both the internal and external paths of influence on consensus.

Internally to an organisation, the interactions between members of the primary CoP will lead to a partial overlap of perspectives reflecting the informal power structure and boundary spanning in each firm. An intra-organisational community that holds significant informal power will export its perspectives to other intra-organisational communities. As safety-relevant standards are perceived as codified best practice, members of an intra-organisational community can use them as moral and economic motivation when struggling to change the consensus on best practice according to their perspective. In other words, in the process of organisational learning of practice within a firm, standards are leverage in conflicts between communities.

Externally to an organisation, CoP continue evolving their definition of best practice. As new practices emerge and become increasingly accepted, this can lead to new means for complying with a standard becoming seen as a realistic alternative by practitioners. Members of an intra-organisational community can thus attempt to change the consensus on practice by importing realistic alternatives to it. During this
process, those struggling to resist change can point to how a new practice is not codified in a standard, while those struggling to change can point at the acceptance of the practice and that the standards allow for it. In other words, in the negotiation of practice within communities, standards can also be leverage.

Fig. 2 summarises our analytical framework.

3. Research design and methods

This section provides a case description, positions the research as a critical case study, and details the data collection, data analysis and validation of the study.

3.1. Case description

The firm on which the study is based is a multi-national engineering company developing CPS. The firm employs about 50,000 employees in 150 countries developing products for both civil and defence purposes in the aerospace, marine, nuclear and power domains. Engineering is set up in organisationally separate business sectors focused on the different domains. However, certain capability functions and initiatives have an enterprise-wide reach. The CoP concept is actively supported by the firm, with more than 350 communities currently organised in a bottom-up way around shared interests. These interests include design and verification, partitioned across disciplines and domains. Furthermore, in line with the original view of CoP as groups self-organising a mutual learning process, the firm also operates several extensive initiatives for bottom-up knowledge sharing - including an innovation portal, a social media platform and several wikis related to engineering.

This study refers primarily to the civil aerospace part of the business. Being the largest sector in the firm with about 20,000 staff directly and indirectly involved, it has existed in various forms since the inception of the aerospace domain. The software experts within the aerospace sector have an in-depth understanding of applicable safety-relevant standards and have contributed actively to standards such as DO-178C, ARP4754A and ARP4761 throughout the last three decades. They are also active in working groups relevant to many aspects of software engineering, such as those of the Object Management Group, the International Council on Systems Engineering and the United Kingdom Safety-Critical Systems Club. Furthermore, the primary software unit in this business sector includes a group dedicated to supporting software engineering across all business sectors, for instance through knowledge transfer between them. Strategic outlook capabilities thus include both a depth in the aerospace domain and a breadth across several other domains.

Since the software level concept was incorporated into DO-178A (RTCA, 1985), the business sector has returned to it at each major update of product architectures. As the focus of the software development organisation in this business sector is on software components at the highest level of assurance, the approach to diversification into software levels has traditionally been to limit it. However, software is still currently being developed in accordance with several of the levels defined by DO-178C.

This development of highly safety-critical systems is currently changing due to influences from the wider CPS industry, where the use of smart functionality such as Artificial Intelligence (AI) and predictive analytics is increasing quickly (Töngren et al., 2017; Geisberger and Broy, 2015). The technical impact of these trends is straight-forward. Manufacturers make use of smart functionality. However, they do not make use of all functionality made feasible by such technology, as some of it would imply that the associated components would have to be assured at a high software level. The software level of the associated components is thus kept low, allowing manufacturers to avoid the
considerable difficulties of handling smart functionality at higher software levels. This means that components at lower software levels increase in importance and size – firms have even had to diversify into more software levels than they have traditionally considered.

3.2. A critical case study

Case studies have been used extensively in software engineering, especially since this type of research often tries both to increase the knowledge of a phenomenon and to change it (Runeson, 2012). That case studies can be used to explore or describe a phenomenon is generally accepted, while the explanatory power of case studies is a more contentious issue (Runeson and Höst, 2009). This is due to explanatory research in case studies typically relying on a qualitative understanding of a phenomenon in its context to argue for the generalizability of conclusions (Runeson, 2012).

The studied firm supports standards both existing and under development, interacts with organisations gathering inter-organisational CoP, and provides intra-organisational support for the CoP concept. The firms’ CoP thus enjoy the freedom to organise bottom-up and hold significant informal power, while receiving direct access to current, evolving and future best practice. With the business sector in focus comes a large international stable environment which has revisited the concept of software levels over several decades. This case thus forms a critical case (Flyvbjerg, 2011): the CoP can be expected to strongly set the consensus and agenda on practice with little preconception of what constitutes valid practice; if disagreement regarding how to change practice has never existed or led to significant struggle, then one should not expect anything but case-specific conflicts in other firms. Using our case we can thus with strong confidence verify the existence of any underlying generalizable conflict centred on software levels within or between CoP. This case is also ideal for exploring the characteristics of such a conflict, since engineers at the firm are likely to have experienced it over time. However, by being a “most likely” critical case (Flyvbjerg, 2011), it is not well suited to exploring other conflicts that might only manifest under specific circumstances. Therefore, this is not an objective of this study.

3.3. Data collection

Data was collected through semi-structured interviews according to the procedure defined by Brinkmann and Kvale (Brinkmann and Kvale, 2015), which includes thematising and designing the investigation; conducting, transcribing and analysing the interviews; and verifying and reporting the results.

The thematising of the interviews started when the case and research questions were elicited as part of the planning of the case study. The interviews were intended to both verify the existence of conflict and describe it, with the case allowing for generalizable results. To ensure that findings were not simply a reflection of prevalent organisational values, it was decided that these should first be identified through a preparatory round of interviews conducted across the firm. The sampling for this preparatory round was opportunistic, focusing on known opinion leaders across the firm, and ultimately included 13 people. The selection of interviewees for the primary round was by contrast careful, focusing on covering a representative and knowledgeable sample from each of the three primary CoP. The characteristics of the resulting sample of 20 people are reported in Table 1. As part of the data analysis, we identified two groups of designers: signers with a tactical role, i.e. those involved in the day-to-day activities of software development; and designers with a strategic role, i.e. those involved in such things as expert advice and process planning. The last column in Table 1 lists to which group each software designer among the interviewees belonged.

An interview script was designed for each round and community. These scripts ensured that all topics were covered. They also allowed for the interviewers to “push forward” (Brinkmann and Kvale, 2015), as several follow-up questions were explicitly noted in the scripts. This ensured that researchers were reminded to clarify the meaning of ambiguous statements. The threat to internal validity of interviewees providing biased or incorrect information was also considered. This was deemed most likely if a response could somehow affect an interviewee’s career negatively. Therefore, all interviewees were assured that no data would be shared from the interviews that could be used to identify an interviewee.

Interviewers need to listen actively (Brinkmann and Kvale, 2015), otherwise internal and construct validity can be compromised by ambiguous responses or failure to follow up on important leads. To ensure active listening in this study, at least two interviewers, sometimes three, were present at each interview, each taking turns to either ensure that the interview script was followed or focus on the interviewee’s responses. For the primary round a pre-interview questionnaire was also sent out, which supported active listening through the opportunity for the interviewers to discuss the script in more detail prior to the start of each interview. Each interview took about 1 h to conduct and all were conducted face-to-face.

All interviews were recorded and then transcribed by the firm’s transcription service. To ensure the reliability of the data the transcribers were instructed to leave parts that were difficult to transcribe to the interviewers. As the aim was solely to capture the meaning of the interviewees’ comments, they were not transcribed verbatim. However, neither were grammatical errors corrected. This ensured that the data analysis could identify ambiguities and refer back to the recordings in order to handle them.

3.4. Data analysis

The data collection for the preparatory round took 1 month, followed by 3 months of analysis and verification. The data collection for the primary round took 2 months, followed by 6 months of analysis and verification.

During the initial analysis of each round, each interviewer coded the transcriptions with descriptive codes (Saldaña, 2009). All three interviewers then met weekly to discuss the codes and arrive at a common code book. The code book for the preparatory round eventually included 167 codes, and the code book for the primary round 174 descriptive codes. All interviewers agreed that the meaning and application of these codes were consistent. This was required to ensure that internal validity was not affected by unreliability of the coder or coding.

The initial analysis was followed by weekly meetings focusing on recoding the descriptive codes to identify patterns (Saldaña, 2009). The resulting secondary coding aimed at interpreting the meaning of the interviews in light of the analytical framework (Brinkmann and Kvale, 2015). Through this process, patterns were developed iteratively, which resulted in the themes reported in Section 4. This ensured that the interpretation was free of contradictions, even when the findings seemed paradoxical at first glance. It also ensured that the overall interpretation could be tested against its parts, the firm and the literature forming the analytical framework – all important parts of analysing the meaning of interviews (Brinkmann and Kvale, 2015). An example of the development of the categories is given in Fig. 3 to help clarify the process to the reader and to illustrate how initially contradictory statements were reconciled.

3.5. Validation

As outlined in the previous subsections several actions were taken to ensure the internal validity, construct validity and reliability of the study: the preparatory round ensured that results were not simply a reflection of the firm’s organisational values; the design of the interview script, the pre-interview questionnaire and the use of several
Table 1
Profiles of the interviewees from the primary round.

| Interviewee | Experience at Firm (years) | Background at Firm | Community of Practice | Tactical or strategical designer |
|-------------|-----------------------------|--------------------|------------------------|---------------------------------|
| 1           | 20–25                       | Design             | Design                 | Strategic                        |
| 2           | 20–25                       | Design             | Manager                | (Not applicable)                 |
| 3           | 20–25                       | Quality Management | Manager                | (Not applicable)                 |
| 4           | 20–25                       | Design and Verification | Manager             | (Not applicable)                 |
| 5           | 20–25                       | Verification       | Verification           | (Not applicable)                 |
| 6           | 20–25                       | Verification       | Verification           | (Not applicable)                 |
| 7           | 20–25                       | Design and Verification | Manager             | (Not applicable)                 |
| 8           | 20–25                       | Design and Verification | Design              | Strategic                        |
| 9           | 20–25                       | Design             | Design                 | Tactical                         |
| 10          | 20–25                       | Design             | Design                 | Strategic                        |
| 11          | 20–25                       | Design             | Design                 | Strategic                        |
| 12          | 30–35                       | Design and Verification | Design              | Tactical                         |
| 13          | 15–20                       | Design             | Design                 | Strategic                        |
| 14          | 20–25                       | Programme Management | Manager             | (Not applicable)                 |
| 15          | 20–25                       | Design             | Design                 | Tactical                         |
| 16          | 15–20                       | Verification       | Verification           | (Not applicable)                 |
| 17          | 20–25                       | Design             | Design                 | Strategic                        |
| 18          | 20–25                       | Design             | Design                 | Strategic                        |
| 19          | 20–25                       | Verification       | Verification           | (Not applicable)                 |
| 20          | 15–20                       | Design             | Design                 | Strategic                        |

Fig. 3. Pattern codes to descriptive codes.

interviewers removed ambiguity; the sample of interviewees ensured a complete coverage of perspectives across the firm; anonymity minimized the risk of false or biased information; several interviewers decreased the chance of interviewer bias; following up on uncertain transcriptions increased the reliability of data for analysis; analysing and coding together meant coder bias was minimized and coding reliable; and testing interpretations against each other, the firm and the organisational framework ensured consistency.

The cooperation on preparing the interview script, coding and analysis was a major part of ensuring internal and construct validity by minimizing bias, as the researchers all came from different backgrounds. Member checks were also used to ensure the internal and construct validity of the results (Creswell and Miller, 2000). This meant that interpretations and conclusions were continuously checked with employees of the firm, and the complete study eventually presented and verified by two of the interviewees as well as other senior employees at the firm.

The external validity of the study is primarily based on the analytical generalisation presented in Section 3.2 (Brinkmann and Kvale, 2015). Identifying underlying generalizable conflict should reasonably be within the abilities of this study. However, it is difficult for this study to define the magnitude of the effect of such conflict at other firms, since these may exhibit other case-specific conflicts that counter or enhance the effect. This is not unexpected in qualitative studies, as the transferability of results is often a prerogative of the reader (Brinkmann and Kvale, 2015). Further research is thus required to verify any implications of the results in smaller, less international, more hierarchical or less externally facing firms.

4. Results

This section presents the results from the study in the form of themes arising from the interviews. These are grouped into tables based on which CoP shows consensus on the theme, with example quotations to clarify their individual meaning. Summaries are provided after each table to clarify the combined meaning of each group. When deemed appropriate to simplify reading, examples from the tables have also been reproduced together with the summaries. Similarly, when quotes have been obfuscated to hide the interviewees’ identities, some examples are also provided together with the summaries.

4.1. All in agreement

Tables 2, 3 and 4 group those themes on which all in the firm agreed, including all members of the primary CoP pursued.

Interviewees across the board stated that they had a high awareness of the importance of individual employees acting in a professional manner, especially with regard to the possible implications of the firm’s products on safety.

Everyone also stated that the firm’s comprehensive standard-based process descriptions were not the primary factor behind engineers adapting a professional safety-minded engineering approach. According to the interviewees this practice was primarily learnt by engineers working together with other engineers.

The interviewees stated that cost was the largest issue with working at higher assurance levels, and the reason firms would want to develop software components to lower levels. However, according to the
Table 2
Example quotations on core values.

| Professional responsibility | “… they should have a responsibility for doing the best they can and taking, yeah, I don’t know, whatever you want to call it, ethics and environmental and all those other things that say well actually I’ve got the – and obviously responsibility to the company as a whole.” |
| Safety as a Core Value | “So your professional responsibility to [Firm] is to deliver a quality product that’s safe that meets the customer requirements, is efficient, etc., the whole raft of different criteria around it. You’ve got to take professional responsibility for delivering that.” |

4.2. Agreement across CoP

very much ingrained in the engineering workforce and unlikely to interviewees the current practice, associated with higher levels was very much ingrained in the engineering workforce and unlikely to change to accommodate lower costs when moving to lower levels.

Table 5 groups those themes on which software designers and testers agreed, and Table 6 gives the themes on which software designers and managers agreed. This provides additional detail to the themes identified across the firm.

Designers and testers attested clearly to the importance of standards and processes, but maintained that there was little need for the average engineer to frequently refer back to them to ensure their correct use. Indeed, they stated that there were other, larger risks associated with well-structured processes divided into multiple independent steps: the processes could pigeonhole employees, indirectly decreasing the employees’ ability to maintain product safety by decreasing the understanding of how activities ultimately contribute towards this goal. Furthermore, the independence could lure engineers into a sense of security based on later process steps, directly undermining product safety by decreasing the rigour of engineering activities. Designers and testers stated that avoiding these issues was beyond the abilities of the average manager, who relies on engineers to provide the appropriate understanding of practice to get the processes right.

Designers and managers both stated that there was a need to change some parts of the current practice to work cost-efficiently at multiple software levels. Simultaneously, rather than eliminating as much practice as possible at lower levels, they agreed that much of the other parts of best practice should be kept even if this required a large costly effort. Among these other parts of best practice, several of the interviewees mentioned the need to keep performing reviews of all aspects of high-level requirements, and the importance of keeping these reviews independent of those creating the requirements.

4.3. The design community – tactical vs strategic

Engineers appeared to wield informal power through their specialist knowledge, as the perspective of software designers was a common denominator across the primary CoP. This suggested that we take a closer look at this community, which – as previously mentioned – led us to identify two groups: designers with a tactical role, i.e. those involved in the day-to-day activities of software development; and designers with a strategic role, i.e. those involved in such things as expert advice and process planning. We discovered that these groups held partly conflicting views, but more importantly that they each had detailed explanations for the themes described in previous subsections. Table 7 reports on the resulting categories.

On the one hand, tactical designers stated that dropping some parts of existing practice when moving to lower software levels was possible, as long as a proper assessment was performed. At the very least there was ample opportunity to perform some practice less stringently under these circumstances. Examples of such changes to practice involved working in pairs to improve the design process and ensure quicker feedback, while at the same time dropping the independence between the developer and the reviewer of a work artefact. Another example was to increase the number and scope of changes addressed by change management during a specified time period. This would increase the risk that a fault would be overlooked. However, assessment would have indicated that such faults would be identified later by other means, and this change would also allow quality improvements to find their way

Table 3
Example quotations on learning.

| Learning from Others | “I don’t think it’s credible to give a newcomer a thirty page process, expect them to then follow it. Because by the time they get to page five they’ll have forgotten page two. So at which point they’re going to get the gist of the process and ask the guy next to them, what do I do? “Well, don’t just point them towards the process and tell them to go and do it. They need to be with somebody, at least some of the time anyway.” “I think some people will go and rigorously read the document and other people will ask somebody else … I would say the majority are probably people-people, to be honest.” |
|                     | “So you’d read the process and then see how people were using it, but you would follow how people were using it.” |
|                     | “You’ll ask somebody that you’re working with … I think it’s human nature to say, how do I do this and somebody will show you what they do.” |
quicker into the product. Tactical designers stated that this would rely on engineering judgement and informal coaching of newcomers, as they could only identify an implicit link between development activities and product risk. On the other hand, strategic designers stated that they were negative to such attempts, and for several reasons: it could incur large costs later in the life-cycle of components; there were several examples of how dropping some parts of practice had had a negative effect on development; and large reduction in cost would not come by dropping some parts of current practice, but rather with the introduction of new practice to support product functionality currently un-feasible to introduce at the highest level of assurance. The strategic designers gave several examples of such new practice. One example was the use of Commercial-off-the-Shelf components and techniques for ensuring that these complied with written specifications or given performance requirements. Another example was the use of algorithms for machine learning, specifically when one could take advantage of certain use cases not requiring these algorithms to be deterministic. None of these examples were mature solutions, but rather early suggestions based on an appreciation for the challenges of contemporary safety-critical systems development. They all either sought to decrease the reliance on or change the character of the required process evidence at the highest level of assurance. Instead, these suggestions attempted to leverage on (novel) characteristics of the software components and the product environment to ensure confidence commensurate to the contribution of the software to system risk.

5. Analysis

This section addresses the first two research questions by analysing the results. It is thus our interpretation of the interviewees’ responses in Section 4 based on the analytical framework provided in the end of Section 2. However, where appropriate we refer back directly to the framework or the interviewees’ statements as summarised by Table 2–7.

5.1. A conflict on the implications of diversifying

The importance of organisational values and informal learning at the firm, as described in Tables 2 and 3, arguably lines up well with the CoP concept. Superficially, the situation seems straightforward to explain based on the themes elicited across all interviewees. As exemplified in Table 4, the general perspective by the interviewees was that working primarily at the most rigorous software levels will ingrain the practice required at these levels into the organisational culture of large firms. While firms can, to lower costs, try to drop part of this practice by diversifying development into several software levels, they would struggle to reap the associated benefits. Engineers used to working at the highest software levels would not be comfortable with changing their practices even when the implications of a component failure is not severe. If the interviewees are correct the result would rather be a conflict on practice between management and other engineers, with management at a disadvantage due to the specialist knowledge of designers and testers. The effect of this informal power is also seen at the firm: in Table 5 designers and testers describe the importance of their expertise, and in Table 6 managers echo the discussion within the designer community. From this perspective the evolution of product architectures provides opportunities to accommodate smart functionality, but engineering practice inertia acts as an obstacle to realising these opportunities.

We argue that this type of conflict can be described as centred on single-loop learning in line with the internal path to influence consensus outlined by the analytical framework. The different CoP observe an unintended consequence in the too high cost of development, but managers and designers disagree on how to address it. However, due to the implicit nature and technical complexity of the associated safety-relevant standards this disagreement is arguably unlikely to lead to a direct struggle for the right to decide on the future of practice within the firm. As designers are in possession of the skill required to interpret the standards, they have the final say on whether practice will change or not. The situation could of course be altered if safety became less important as an organisational value at the firm, or if managers recruited engineers from outside the safety-critical industry to develop components to lower levels of assurance. Strictly speaking, the former case has implications far beyond that of software levels and is best studied separately – for instance in relation to firms close to economic collapse. The latter case could change the situation for a time, but the designers would still be interpreting the standards. The newly employed designers could be kept separate from the rest of their community, but eventually the common connections to safety management and the importance of safety should lead to interactions. The discussion regarding the future of practice at the firm would then occur inside the designer community.

The example statements in Table 7 from the designer community by contrast suggest how disagreements could lead to conflict, as perspectives differ between tactical and strategic designers. To explain the situation one should note that both of these groups will agree on the lowest possible software level for any given software component and product architecture, as it is decided by the product’s current safety assessment. The divergence arguably lies in their roles leading to different ideas about why and how to change practice to ensure that this development is sustainable. Tactical designers are...
| Table 5 | Example quotations on agreement between designers and testers. |
|---------|-------------------------------------------------------------|
| Infrequent and Partial Reference to Standards and Processes | “I should imagine that for most test engineers in [Department] they have a specific [Department] test process, they read it when they first start, it’s not very complicated, it tells us what to do. Every now and again we look at it, if it’s a bit of clarity, probably once every two or three years, just to re-familiarise yourself.”
| | “What I have done in more recent years is look at specific elements of that, so things like review checklists for example, I’ve not looked through all of the process every time, what I’ve thought is, I understand what we’re doing, I understand the basic process, every now and again I think, I’m not quite sure I’ve remembered exactly all the details of a certain part of it, so I’ve gone to look for those. That tends to be what makes me go to look.”
| | “Very few, because most people have to just do one point in the process many times. So they might have to produce a low level test, do six months at low level test, so they’ll go and ask the person who knows how to do low level tests on that project how to do it and that’s probably adequate. The team leads and people like myself might step back to the plan set, and work out way through from that. If I’m doing a stage of involvement audit I’m actually not looking for process compliance anyway, I’m looking for certification compliance. So I’m looking that the objectives of [Standard X] have been met or [Standard Y] or [Standard Z] have been met, so I don’t need to look at the process reviews to produce the stuff, I only need to look at the stuff. So yeah, we plan that, we write it. Once it’s been socialised, once it’s embedded that that’s how we do stuff we don’t go back and refer to it every day to work out what to do.” |
| Process Pigeonholes People | “I think so because I think we’re kind of – certainly now we’re geared up to very big teams and people just doing a very tiny part of a job … It just seems to have grown up over the years that teams have become bigger and bigger and more complex and people are isolated.”
| | “They work in that bubble and that’s what they do … So it’d probably depend on where you are in your career and what’s gone on. So early in your career, and some people just stay there, you will be pigeon holed into you’re doing [certain activity] and you’re doing it for the next six months. Some people stay there forever.”
| | “I mean if you’re just doing something, which most people do, in isolation … well, it’s not necessarily going to be evident that if you get your bit wrong … I’m sure it will be because some people won’t have an overview of the whole system because they’ve never actually done anything with the whole system; they’ve just done — I wouldn’t say always the same part, but just parts of that they can get to put together to make the system.”
| | “I think it would be useful for people to have a general understanding about how all of the objective interrelate, because that’s one of the big issues that people only see their small part of the process … they make changes that have knock-on effects downstream that they don’t understand, because they don’t understand how the evidence that they’re producing actually contributes to a set of evidence that shows compliance to the standard. So understanding, you know, [Standard] isn’t a discrete set of things you do; it is a set of things that collectively provide a body of evidence that allows you to come up with effectively a safety case or a statement of compliance. And people, they don’t necessarily have to understand the detail of the wording in [Standard], but they should understand how the whole software process then produces that set of evidence for certification purposes.”
| | “Well how can you be personally accountable if a [high assurance] process goes to independent review and then goes through independent tests and … it goes through so many steps that by the end of it you kind of think, well if everybody else has seen the behaviour and seen all the artefacts and they’re all happy with it, it must be right.”
| | “You’ve got a rework cycle in that or a scrap cycle because they don’t get it right first time, and in part that process drives our rework cycle because it introduces a mind set that I can let it go at slightly lower level of quality as a non-zero defects mind set because I know there’s an independent review and that guy will catch it and give me a comment, so I’ll get it on the flip side when I do my rework, and that’s obviously not good.”
| | “I don’t know if it’s fair to blame the process. I think the organisation makes it very difficult to feel accountable, because of the size of the project, you know, there’s a massive feeling I think, you know, that there’s an army of people that’s going to test this somewhere else, and I don’t know even know who they are, so I don’t have to worry too much maybe because if there’s an error, somebody else will find it.”
| Process Makes Me Feel Safe | “They employ people like [Design Specialist A] and then myself and [Design Specialist B] to make these decisions. So if we had to go and justify some budget to do it, yeah. But to actually make a change — there’s a group of us would come to a conclusion.”
| | “I think it comes from the under-confidence of management, who shelter behind processes. They don’t, they’re not in post long enough to really understand what they’re doing, a lot of them are advanced enough early, before they’re ready for it, so it’s a process role, you know, a process should be grey bars, and not everybody’s suited to it.”
| | “I think we cut in at probably [high management] level to say, yeah, I think somebody in [senior management] position could be expected to have enough of an understanding of what processes we’ve been working to and how we’re going about doing that. Enough of an understanding that we need, that he needs somebody at a line below him if you like to be making sure that those are still the right practices. So I wouldn’t expect [senior manager] to be looking at that and saying, should we change something, but I would expect him to have an interest in making sure there is somebody taking account of changes in the world, and keeping up to date with what’s happening. So somebody feeding [senior manager], you know [senior manager] encouraging that, and somebody feeding [senior manager] to say, this is what we’ve done previously, we think there’s a benefit in doing this.”
| | “I think ultimately there’s need for engagement [from management], but I don’t think there’s need for engagement in the nth degree of detail. I think that’s more of specialist role and even then only when strictly necessary. We need to develop our people such that they can do the detail and abstract such that they tell you about why it’s good.”

pressured to minimize the risk of not delivering on time with the available resources. This risk is minimized by keeping to well-known practice, but lowering the constraints on software development when possible. From this perspective, an in-depth understanding of how to eliminate engineering practice decides if it is possible to accommodate smart functionality. This knowledge is then what allows one to enable sustainable development by moving to an efficient, uniform set of practices, applied at several software levels.

In other words, both groups are trying to address risk, but as each role focuses on different risks they advocate elimination and transformation of practice, respectively. Tactical designers advocate elimination of practice: that practice can mostly stay the same, but that – at times and at different software levels – there is room for dropping some parts of existing practice or doing it less stringently. Strategic designers advocate transformation of practice: that realistic alternatives to existing practices should be adopted at all times and across all software levels. There is thus no real obstacle to the diversification of software levels, but there is a conflict on what this diversification should entail. This answers our first
research question by identifying a generalizable conflict within the community of software designers.

5.2. Characterising the conflict

In this subsection we address our second research question, characterising the nature of the conflict.

To explain the situation we note that tactical and strategic designers agree on the need for single-loop learning. As exemplified in Table 5, designers highlighted that the average engineer would not refer to standards frequently, with the risk of processes pigeonholing engineers or providing a false sense of security. In response, both groups work within the framework of guidance provided by standards to shape the practice of the average engineer at the firm. As exemplified in Table 7, tactical designers discuss the lack of an effect when dropping certain practice, while strategic designers highlight how this might mean that life-cycle costs are forgotten. There is also no disagreement that safety should guide this learning, as everyone interviewed at the firm agreed that this organisational value was paramount. The conflict is thus not a matter of organisational single-loop learning.

The conflict is arguably rather a matter of organisational double-loop learning in line with the external path to influence consensus outlined by the analytical framework. The norm in flux is whether to aim for aligning with the well-established practice or the most promising evolving practice in the inter-organisational community of practice. The former would already be explicitly coded into standards, while the latter would be allowed by (but not necessarily explicitly coded into) them.

This means that even if only designers and testers possess the technical expertise to interpret standards, software levels carry broader implications for safety management.

6. Discussion

This section addresses the third research question by discussing our analysis. It is thus our evaluation of the implications of our results for both practice and theory. In regard to practice, we contribute by discussing the implications of the identified conflict for the studied engineering context. Regarding those who work in engineering organisations developing CPS products. Therefore, rather than building directly on the theoretical framework used for the analysis, we contribute to the theory on safety by linking our results to more generic concepts in the discourse on resilience.

6.1. Contribution to practice

This subsection starts by evaluating whether firms might be affected in different ways by the conflict described in the previous section. After identifying how a difference in organisational values should make it more likely that a firm supports the perspective of either tactical or strategic designers, we discuss the implications of this on safety management.

6.1.1. Early adopters and the majority of firms

As seen in Tables 6 and 7, the interviewees provided examples of how practice could – in their opinion – be changed for the better as the importance of lower software levels increase. It should be noted that this study does not focus on identifying the “correct” choices in this regard. Many of the suggestions were based in tacit knowledge, making a persuasive argument for adopting them outside the reach of this study. Regardless, such an aim would have presented at least two other major problems. Firstly, it could be in the interest of a firm to spread news about promising practice later rather than sooner, in case it provides a competitive advantage. Reporting on promising practice could thus be difficult, even if it was identified. Secondly, even if a practice is accepted by an increasing part of a community, it could be failing in ways that are not obvious to practitioners. The associated failures could be rare or difficult to detect, meaning that a significant amount of field data would have to be gathered to make an objective evaluation of a change to practice.

In fact, as the identified conflict is related to double-loop learning, whether practice is eliminated or transformed will most likely depend on the mission statement of a firm (Arad et al., 1997; Koryak, 2018). This statement articulates the firm’s organisational values, including the priorities it assigns to organisational states and outcomes such as innovation and predictability. Organisational values that emphasise innovation should give strategic designers a stronger position in the negotiation of practice within their community in the identified conflict, and vice versa. In fact, even if managers overcome the difficulties in acting independently associated with a lesser in-depth technical understanding of standards as pointed out in Table 5, they would still be likely to act in step with the community of designers: as they share these organisational values with designers, they are likely to assess arguments in a similar way. What our study indicates is thus that software levels will have different implications for practice in regard to the adopters that make early use of new technology to gain disruptive benefits and the majority of firms that do not make it their mission to lead on technology (Törngren et al., 2017; Geisberger and Broy, 2015). The former are more likely to embolden the strategic designers that seek to adopt novel practice. The latter are more likely eliminate practice when diversifying development between several software levels. In fact, the latter might even carry on without changing practice.
Table 7
Example quotations from strategic and tactical designer roles.

**Tactical: Eliminating a Practice – Acceptable**

“We could develop [lower assurance] with the same type of [Practice A] that we use for [higher assurance]. My view is that we don’t need to do that and that therefore, when I think we are in some ways doing that already with the [CPS Product], we are.”

“I think we tend to apply it too early in a product development, certainly when it’s pre-release.”

“Oh no they wouldn’t, they wouldn’t, because they wouldn’t have time. There would be – even without knowing the details of the project, I can guarantee there would be time pressures and so if you told people they don’t have to do a particular part of the process, they will happily not do it.”

“I think there are probably some more that we could take out, it’s a matter of, you know, taking the time and just working out what’s actually required. There’s perhaps not enough benefit to spend the activity time doing that now, or it doesn’t, maybe there is, but it doesn’t seem obvious to do that.”

“Well, if I didn’t do this to any kind of great degree would it make any difference to the end product? And you could probably say, “Well no, it wouldn’t.” … whereas other things, doing a part of a design or something, you might say, “Well, yes. That does have a big impact on that.” But it’s kind of down to experience I suppose and engineering judgment in a way.”

“In terms of the consequences, it depends what you’re looking, sort of the specifics or more general … specifics, it’s probably difficult to pin that down.”

“So you know that, even if we didn’t do the [Technique A], a greater proportion of things would get through, but some things get through anyway, so yeah, I don’t think you can ever, I don’t it’s easy to say there’s a direct link between one activity, but they all, overall add to the – so it’s an incremental thing isn’t it.”

**Tactical: Task Risk Unknown**

“Some are based on experience of – sometimes you can kind of say, “Well, if I didn’t do this to any kind of great degree would it make any difference to the end product?” And you could probably say, “Well no, it wouldn’t.” … whereas other things, doing a part of a design or something, you might say, “Well, yes. That does have a big impact on that.” But it’s kind of down to experience I suppose and engineering judgment in a way.”

**Tactical: Coaching New Generations**

“A: But certainly, the [Low Assurance Development] team – I think it’s just something that we all know where we’ve got to get to and the team works together to get there … it’s very varied … people with 5 years experience; 10 years, 20 years. It’s a very, very diverse team … We have regular team meetings and issues like that will be kind of flagged up at the team meetings and probably discussed and then people will go away and kind of discuss those things outside of the meeting as appropriate.

B: So a lot of these issues are really solved by informal discussions?

A: Yeah.

“… but that as part of their development within the organisation, it’s not because you’re following mandated process that says you have to do it … I’m just saying that because a process says you don’t have to [Technique A], that doesn’t mean that it wouldn’t still be part of training people and starting to deploy them onto projects and stuff like that.”

“So that’s still the senior members of the team going round and talking to more junior members. So it’s much more informal … it’s much more informally going and talking to people a couple of times a week and saying, “If you’re doing this, how are you getting on? What are you working on?” and having that discussion with people. If, through that discussion, you think that somebody’s maybe over-egging a solution, maybe doing something a little bit more complicated, over-considering what they’re doing, then you kind of bring them back down a bit and say, “Well actually, don’t worry about that. Concentrate on this.” Or equally, if somebody’s maybe not documenting everything or not doing what you think they should be doing according to the process, then kind of guide them at that point and say, “Well actually, you need to do more documentation. You need to document that particular aspect or do this sort of test,” … “…. The coaching aspect is there, but it’s informal coaching a couple of times a week.”

**Strategic: Keep Due to Quality and Life Cycle Costs**

“So, my interest is not so much in trying to, if you like, drop the level of software down to something that is more economic for producing bits of software, but it’s to try and raise the level at which we can do software in an economic way, because there’s a lot of evidence, and I think I put a reference to an article in my thing there, that if you do things rigorously, it may cost you more upfront but it will save you a lot on the maintenance cost, right, because you get fewer problems in the field, which are terribly expensive. … So it think we should be very careful to make sure we look at life cycle costs rather than development costs. That’s the bit that worries me at the moment with all this discussion is we’re not looking at the whole life cycle, we’re just looking at the very immediate development cost.”

“… ‘Yes, the business requirements are still high, so you might adhere, you might choose to adhere to everything that is actually down there for [high assurance], even though you’re doing a [lower assurance] job, just because you want that level of confidence in the product”

“Because of course for them the impact of getting it wrong and getting it out into marketplace was – had massive business impact, just the same as it does in many safety critical ones. So the reasons for doing high integrity software are far broader than pure safety.”

“I think the [Practice A] gives a lot of benefit around the quality side of things, and finding the issues still earlier in the life cycle.”

“I suspect that’s one that you could lose very quickly … and I think in most cases it’s failed because actually when you do [Practice B] light what tends to happen is people stop doing [Practice B] and they do get into all the kind of messes that [Practice B] was trying to avoid, sort of the speci… it’s the bit that worries me at the moment with all this discussion is we’re not looking at the whole life cycle; we’re just looking at the very immediate development cost.”

“… “…. The coaching aspect is there, but it’s informal coaching a couple of times a week.”

**Strategic: Eliminating a Practice – Bad Example**

“I think it would be very interesting if the business to reflect on the experience on [project] where they’ve been trying to run outside of [Practice B] for quite a long time. And it certainly – it obviously has its positives, but it certainly seems to have some significant negatives as well. So I think there’s got to be balance somewhere and I’m not sure we’ve got it.”

“Well, I think the evidence is that if you don’t do some form of [Practice B] then you can lose track of what’s going on very rapidly.”

“I think we’ve, well there’s been several projects where the [restructuring project] cost hasn’t, I don’t think, hasn’t met what the business thought it was going to be…”

And there’s a lot of evidence that – I mean [Manager] again collected a lot of evidence in this area, saying that [Practice A] paid for itself.”

“ ‘I think that it would be very interesting for the business to reflect on the experience on [project] where they’ve been trying to run outside of [Practice B] for quite a long time. And it certainly – it obviously has its positives, but it certainly seems to have some significant negatives as well. So I think there’s got to be balance somewhere and I’m not sure we’ve got it.”

“Well, I think the evidence is that if you don’t do some form of [Practice B] then you can lose track of what’s going on very rapidly.”

“I think we’ve, well there’s been several projects where the [restructuring project] cost hasn’t, I don’t think, hasn’t met what the business thought it was going to be…”

“My argument is we should be able to bring in alternate means as soon as we get down to [lower assurance] and below, particularly if we actually don’t want to develop the software ourselves, if we actually want to bring in COTS components or software developed elsewhere or software of uncertain pedigree, we don’t have process evidence for those. Typically we don’t have process evidence for software acquired via a means other than develop it ourselves, but it might be the most cost-effective way of getting that particular

(continued on next page)
product. So having the ability to have some sort of in-service reliability metric, say, for software and use that as a means of saying, “This software is good enough.”

“I think that what we haven’t touched on, I don’t think, is opportunities. So there are – we’ve talked about variation of [assurance] based upon process objectives and whether people can switch between them, etc., etc., etc. There are product features that are enabled by [lower assurance]. So there are things that – this is probably a debate actually. Does [high assurance] mean deterministic? So there are features that we’re talking about putting in our [product] that are things like algorithms that iterate to solutions and the more time you give them to iterate, they will get a more accurate solution, but less time it’ll be a less accurate solution but it might still be valuable, etc., etc. So there are soft real time and processor resource-based functions that we might be able to take advantage of …”

“Yeah, I mean you know, you said that if we take something like the [COTS Component] that we’re not really striving for [high assurance] in that. I don’t see it like that, I think we are. I think when we take something like the [COTS Component], we’re actually saying, I think there’s a lot of, what we would’ve called in the [Standard] World Service History, associated with the [COTS Component], therefore if we use this particular version of it, we’ve got this evidence about how stable that is, and how good that is and what the bugs are in that.”

“And, you know, that sometimes the potential savings of working at a [flow assurance] are exaggerated and the overhead of working to [high assurance] is not necessarily as much as people sort of feel it can be.”

instead waiting until any new practice has been universally accepted or even codified into standards.

6.1.2. Splitting the community

Firms that choose not to lead on technology will see less of an impact at lower software levels and be less willing to move ahead of other firms in the market. They are thus likely to support the perspective of tactical designers. By being more likely to allow a differentiation between software levels, with the software at each level being substantially different in size and importance, it is also more likely that firms that do not lead on technology will physically split the community of software engineers between levels. As mentioned in Section 5.1, engineers with a background from outside the safety-critical industries could even be recruited to lower levels to avoid newcomers insisting on practice enforced at higher levels of assurance. It is unlikely that this isolation would be complete, as both communities would likely be coordinated by the same safety management. However, knowledge will no longer flow as easily between community members.

Engineers working at higher levels in the majority of firms will thus be less exposed to trends such as smart functionality. They will not need to make the associated judgement calls and thereby become less prepared for the day when the technological trend makes it into highly safety-critical functionality. As an example, the optimization of fuel consumption is a key selling argument in the aviation domain. The introduction of AI could be used to lower fuel consumption. One can also envision future functionality where this AI is used for adjacent reasons, such as how to handle fuel during an emergency landing. However, as long as the limitations of this optimization cannot be fully assured, an aeroplane will – to ensure safety – still have to be weighed down by carrying a non-optimized amount of fuel. If the gain is thus limited, the introduction of such functionality can be delayed. This would mean that engineers working at higher levels of assurance would not engage with associated strategic capabilities, such as the new practice and certification arguments needed to develop and evolve features based on AI.

To avoid a situation where the resilience of a firm is left vulnerable by engineers working at higher levels not knowing what to expect and look for, safety management should support the identification, organisation and training of informal opinion leaders working at different software levels. By being aware of differences across the organisation, gatekeepers (Paul and Whittam, 2010) and organisational liaisons (Tushman, 1977) can then eventually act as “knowledge transformers” (Whelan et al., 2009) able to help engineers absorb information from other levels.

6.1.3. Organising support across the industry

The process of standardisation in the safety-critical domain is not without issues – it can exclude relevant experts and has to date failed to make many underlying rationales explicit (Habli, 2017). The current shift towards more explicit safety argumentation in standards and by engineering organisations seems likely to succeed (Habli, 2017; Knight, 2014). In the future it is thus more likely that arguing for alternative means of compliance, as allowed by some standards, will come down to providing sufficient evidence of equivalence between the novel and replaced practice in regard to an explicitly defined goal. The challenge for firms will be to ensure that the evidence is plausibly complete, i.e. that all corner cases have been considered. While current standardisation processes itemize applicable techniques well by utilizing a few experts (Knight, 2014), the challenge of identifying corner cases is more suitably solved by including a broad set of participants. The identified conflict centred on software levels makes this involvement less likely to happen, as a large set of the relevant CoP will rather have to focus on meeting the needs of smart functionality by eliminating practice. This would involve a significant effort in evolving the system safety assessment processes that determine software levels and the product architectures that enable their diversification. This will make it more difficult for early adopters of technology to get the feedback they need to ensure they stay resilient when introducing novel practice.

Another concept to CoP that relates to group activity in organisations is crowdsourcing, which describes the use of information technology to elicit crowds to address an intra- or inter-organisational open call (Zuchowski, 2016). A difference to CoP is that expertise is less emphasised, often meaning that pre-selection or limitation of participants is viewed with scepticism (Zuchowski, 2016; Geiger, 2011; Simula and Ahola, 2014). Originally crowdsourcing was used to tap into the “wisdom of the crowd” (Surowiecki, 2004), but crowds have since then also been used for expert problem-solving (Feller, 2012). This activity includes several phases of which the most known is probably the evaluation and aggregation of results, which frequently involves crowd voting (Zuchowski, 2016). A known risk with crowdsourcing is thus the pressure to conform to the opinions of peers (Stieger, 2012), which may lead to certain users dictating the responses of crowd members. However, crowdsourcing also contains phases where the focus is on developing ideas in groups, and the evaluation and aggregation can be performed by select experts. As a complement, crowds could in this way offer a possibility to analyse and discuss the strengths and weaknesses of novel practice across a larger group. Safety management at a firm that is aiming to align with promising evolving practice in inter-organisational CoP would thus do well to establish at least an internal crowdsourcing capability for cooperation on idea analysis and refinement.

In a similar way, additional benefits could be achieved by extending the crowdsourcing to other firms with the intent to support standardization. Firstly, this would allow certifiers to learn from the discussion on assuring novel practice earlier, lowering the risk of using novel assurance argumentation. Secondly, it would provide more examples of how to handle novel practice in the standardization process, clarifying the interpretation of standards. Thirdly, it would make the standardization process less reliant on individual experts, especially if digital
tools for using the power of large crowds were adopted. Fourthly, if large firms involved many of their employees, it would arguably both pressure and make it possible for standardization organisations to lower the cost of gaining access to standards. The largest issue with such an approach would arguably be how to deal with intellectual property with so many employees from different firms involved (as this is a common problem when large firm-external crowds are used (Zuchowski, 2016; Schenk et al., 2017). However, as safety argumentation to some extent relies on openness to be accepted by certifiers, there is an incentive for firms to allow some risk in this regard. Furthermore, ensuring this openness would also be important to avoid the negative effects alluded to in the previous paragraph, such as peer pressure, rumour spread or use of made-up numbers. Keeping the evaluation of the crowdsourcing separate from the crowd would allow experts to evaluate whether the supplied arguments and evidence are objective or not.

6.2. Contribution to theory

The perspectives of the two different groups of designers are well aligned with two models on how rules primarily support safety that have been discussed for operational contexts in the discourse on resilience. According to Model 1 rules “define and guide behaviour in complex and often conflicting environments and processes” (Hale and Borsy, 2013), while Model 2 sees them as “supports, not strait jackets; as tools to coordinate and structure creativity and innovation, not as controls to limit freedom” (Hale and Borsy, 2013). As seen in Table 7, strategic designers gravitated towards a Model 1 perspective, arguing that there were many bad examples from breaking the “rules” of the standards. The tactical designers instead adopted a Model 2 perspective, arguing that expert engineering judgement could usually lead to acceptable interpretations of these “rules”.

While there is a wealth of research on these models in operational contexts, little is known about how they relate to standardization (Hale and Borsy, 2013). We suggest that the results of this study are helpful in contexts, little is known about how they relate to standardization (Hale and Borsy, 2013). We suggest that the results of this study are helpful in this regard. Perspectives adopting Model 2 are often found where firms have to rely on the ability of operators to steer their organisation away from unsafe situations (Hale and Borsy, 2013; Rasmussen, 1997; Dekker, 2006). These firms come to depend on collegial patterns of authority and well-informed decentralised decision making that allow operators to act autonomously (La Porte, 1996). In complex domains guidance is thus likely to be more usable if specified as process rules rather than action rules (Hale and Borsy, 2013). Arguably, the expectation could be that causality would also work in the opposite direction - that firms operating in a complex development environment would align more with Model 2 if more leeway was given to engineering judgement, for instance by less prescriptive standards. Firms could be expected to develop their resilience (Hollnagel, 2006) by enabling engineers to individually overcome the problems exemplified in Table 5, i.e. the risk of processes pigeonholing or providing them with a false sense of security. In other words, that the individual engineer would be tasked with more responsibility for the organisation’s ability to ensure an understanding of what to expect, what to look for and what to do to ensure safety as circumstances change.

However, our results show that rulemaking does not automatically shift towards Model 2 in these circumstances. The perspective on rulemaking that an engineering firm adopts is more likely to be related to whether it is an early adopter of technology. Only then do standards become important by either allowing novel practice to be imported or not. Less prescriptive standards might thus mean that a firm puts less rather than more stock in their engineers’ ability to decide autonomously what to do to ensure safety.

7. Conclusions

Our results suggest that there is a potential for conflict when the consensus on practice for software engineers used to primarily working at higher levels of assurance is eroded by trends that increase the importance of lower software levels. This conflict materialises between tactical and strategic software designers, as the different risk foci of these groups make them advocate elimination and transformation of practice, respectively. The nature of this conflict can best be described as a matter of organisational double-loop learning, where the norm in flux is whether to aim for aligning with well-established practice or the most promising evolving practice.

As the conflict is centred on double-loop learning, it will have different implications for adopters that make early use of new technology to gain disruptive benefits than for the majority of firms that do not make it their mission to lead on technology. For the majority of the firms the risk is that engineers working at higher levels are caught unprepared. To avoid a loss of resilience, safety management should support the identification, organisation and training of informal opinion leaders working at different software levels. For early adopters the risk is that they do not get the feedback they need to ensure they stay resilient when undergoing change. To compensate, safety management should explore crowdsourcing for analysing and discussing the strengths and weaknesses of novel practice. If crowdsourcing for this purpose could be extended across entire industries, it should provide benefits to the entire engineering standardization process.

The contribution to theory by this study indicates that less prescriptive standardization does not automatically shift rulemaking towards allowing operators to act more autonomously. Organisational values, such as the willingness to lead on technology, are more important in establishing the perspective on rulemaking. Only after these values have established a perspective on rulemaking will standards become important by either allowing novel practice to be imported or not. Less prescriptive standards might thus mean that a firm puts less, rather than more, stock in their engineers’ ability to decide autonomously what to do to ensure safety.

Declaration of Competing Interest

None.

Acknowledgements

The research leading to these results has received funding from Sweden’s Innovation Agency (VINNOVA) under grant agreement no 2015-01524. Special thanks go to Vicki Derbyshire for her help in proofreading.

References

Arad, S., Hanson, M.A., Schneider, R.J., 1997. A framework for the study of relationships between organizational characteristics and organizational innovation. J. Great. Behav. 31 (1), 42–58.
Arzyris, C., Schon, D.A., 1996. What Is An Organization That It May Learn? In: Organizational Learning II, Theory, Method, and Practice. Addison-Wesley Publishing Company; Reading, Massachusetts. p. 3–29.
Beecham, S., et al., 2008. Motivation in software engineering: a systematic literature review. Inf. Softw. Technol. 50 (9–10), 860–878.
Björnsson, F.O., Dingsøyr, T., 2008. Knowledge management in software engineering: a systematic review of studied concepts, findings and research methods used. Inf. Softw. Technol. 50 (11), 1055–1068.
Bolmsjö, E., Scavo, E., 2014. The place of communities of practice in knowledge management studies: a critical review. J. Knowl. Manage. 18 (2), 366–381.
Boreham, N., Morgan, C., 2004. A sociocultural analysis of organisational learning. Oxford Rev. Educ. 30 (3), 307–325.
Brinkmann, S., Kvale, S., 2015. Conducting an Interview, in InterViews, Learning the Craft of Qualitative Research Interviewing. SAGE Publications, Inc. p. 125–147.
Brinkmann, S., Kvale, S., 2015. Interview Analyses Focusing on Meaning, in InterViews, Learning the Craft of Qualitative Research Interviewing. SAGE Publications, Inc., 2015, p. 231–247.
Brinkmann, S., Kvale, S., 2015. Thematizing and Designing an Interview Study, in InterViews, Learning the Craft of Qualitative Research Interviewing. SAGE Publications, Inc. p. 125–147.
Brinkmann, S., Kvale, S., 2015. The Social Construction of Validity, in InterViews,
