Liability for Software in Safety-Critical Mechatronic Systems: An Industrial Questionnaire

Holger M. Kienle, Daniel Sundmark, Kristina Lundqvist, and Andreas Johnsen
Mälardalen University, Västerås, Sweden

***Abstract*—There is very little research on how industry is dealing with the risk of legal liability when constructing safety-critical mechatronic systems that are also software intensive. In this paper we propose a case study approach with the goal to understand how liability concerns in this setting impact software development in industry. The approach takes into account that software development is embedded into a complex socio-technical context involving stakeholders from technical, managerial and legal backgrounds. We present first results of our case study from a questionnaire involving six companies that develop software-intensive, safety-critical systems in the vehicular and avionics domains. The results of the questionnaire shed light on current industrial practices and concerns. The results indicate that liability seems indeed a concern and that a more in-depth analysis of this topic would be desirable to better understand the strategies that are used by industry to address liability risks.**

I. Introduction

There is a rather limited amount of research in software engineering that addresses legal considerations, but they are a reality that industry has to deal with. There are many potential legal issues that may need to be considered, depending on the nature of the software and the environment in which it operates. Examples of legal issues are data protection, licensing, intellectual property, contracts, and consumer protection.

IEEE standard 1420.1b (withdrawn) broadly defines liability as "the state of being responsible or answerable under a legal obligation" [1]. Product liability can be defined as "the legal liability of manufacturers and sellers to compensate buyers, users, and even bystanders for damages or injuries suffered because of defects in goods purchased" [2]. There can be intentional liability (requiring an intentional act that is reasonably foreseeable to cause harm) and strict liability (requiring no intent and no negligent act). The latter typically applies to manufacturers of dangerous goods that can cause physical harm. Generally, the standard of care when determining negligence is that of a ficticious person—the "reasonable man of ordinary prudence" [3].

For a software system (product) liability can mean that the producer of "defective" software could be held responsible under certain circumstances for (physical) harm cause by the "malfunctioning" of the software. According to Dowlatshahi "courts have shown little mercy for manufacturers who neglect safety and who produce products that later prove to be unsafe" [4] and Hoffmann reports that "in May [2011], the European Commission introduced a proposal to hold companies liable for for damages caused by faulty software" [5]. We restrict our research to legal concerns surrounding defective software; we do not address other legal issues that may be also relevant in this domain such as contracting for externally developed code and limiting exposure to intellectual property infringements.

There is anecdotal evidence that litigation involving liability is a concern for industry, but authors rarely elaborate on this issue. For example, Ackermann et al. say that for automotive companies and their suppliers such as Bosch "safety, warranty, recall and liability concerns … require that software be of high quality and dependability" [6]. We believe that a more thorough analysis of the current situation in industry is of interest; if liability is indeed a concern for a company, then this concern should be explicitly reflected in some form in the company's software engineering processes and practices.

In this paper we focus on liability risks of mechatronic systems such as they can be found in the automotive and avionics domains. In these domains liability needs to be part of risk analysis because they are safety-critical and accidents may lead to loss of life and liability claims. Liability for these mechatronic systems is closely related to (safety) standards and standards certification. Åkerhom et al. say that "legislative and standardization authorities around the world currently increase the pressure on vehicle manufacturers to comply with safety standards for their electronic systems" [7]. Nowadays, mechatronic system are often also software-intensive systems and as a consequence concerns for liability need to address software as well. Generally, if software development ignores or violates standards then the risk of liability increases.

Liability for software has been mostly addressed from a strictly legal perspective, discussing/arguing how courts may/should apply liability to defective software and to software practitioners (e.g., [8] [9] [10]). To our knowledge there is very little work in the software engineering community on liability. Specifically, we are not aware of any empirical research that has studied the impact of liability on industry with the help of established research methodologies such as case study, survey, or action research. Our research strives towards closing this gap.

Given that there is little investigation on how liability impacts software development for mechatronic systems, we propose to better understand this relationship with the help of an industrial case study. Case study research is well suited because it is "aimed at investigating contemporary phenomena in their context" [11]. In our study, the phenomena are liability and software development in the context of mechatronic systems manufactured by an industrial organization. This leads to our *main research question*:

> How are concerns for liability reflected in companies' software development and their organizational structure and practices?

We are tackling this question by breaking it down into more detailed ones as discussed in Section II. Our case study is *exploratory* in the sense that it strives to establish whether liability is indeed a concern and whether it is an issue that deserves further research. It is *descriptive* in the sense that it investigates the current situation in industry by studying real companies.

II. Case Study Approach

In this section we describe our approach and the key elements of the case study.

For conducting the case study we decided to pursue it in two stages. The first stage consists of a questionnaire, which was sent to suitable people within the targeted companies. One purpose of the questionnaire is to establish whether liability is indeed a promising research topic. This is not clear from the outset because of missing prior research results. A questionnaire seems ideal in this case because it requires little time and effort for both sides. The second stage consists of a more involved data gathering approach in the form of semi-structured interviews. In this paper we report on the results of the first stage only (i.e., questionnaire).

When conducting a case study, it is important to identify the study's context (cf. Section II-A). Since liability cross-cuts legal, technical and social issues within and beyond an organization we decided to approach the context from the perspective of socio-technical theory. We have also conducted a literature search to identify the key concerns that practitioners (both in the legal and information technology domains) have voiced with respect to liability risks and strategies that companies can use to mitigate these risks (cf. Section II-B).

Based on the context and the literature search, we identify relevant aspects (cf. Section II-C) that the case study needs to address in order to provide a holistic picture. Based on these aspects the instruments for data collection—questionnaire and interview—are designed. Our target for data collection are companies that produce safety-critical mechatronic systems. We are aiming to independently study a number of such companies. Thus, we are conducting an *embedded case study* [11].

A. Socio-Technical Context

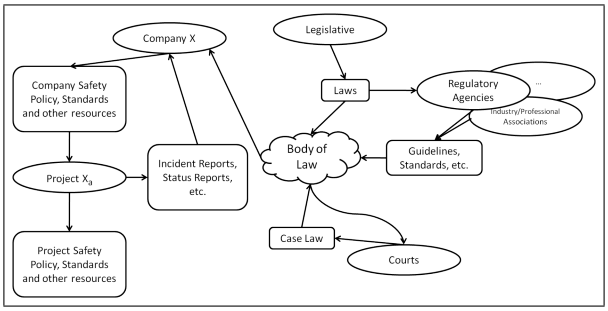


Figure 1. Regulatory structure (case study's context)

Leveson presents a model of socio-technical control that exposes the interactions within an organization and external governmental stakeholders [12]. Figure 1 is based on this model with an emphasis on the legal and liability aspects. The legislature enacts laws (e.g., because they are concerned about the safety of a domain). These laws may be further refined by (regulatory) agencies that provide more concrete guidelines. Law is also shaped by case law (i.e., relevant court decisions that interpret the laws). Individual companies (or industry associations) react to the legal and regulatory environment by defining (company-wide) safety policies and internal guidelines or standards. For individual projects these policies, guidelines, and standards are further refined with concrete practices, including software development practices.

Importantly, there is also feedback in the other direction. Experiences from individual projects may cause changes to company-wide policies, and incidents and accidents involving products in the domain may cause regulatory changes. Socio-technical control is present not only at systems development, but also at systems operation/maintenance.

B. Literature Search on Liability Risks

We have conducted a literature search to identify papers that address liability for software. Specifically, our goal was to identify papers that go beyond a mere description of the legal side by providing (actionable) guidance for software practitioners. Table I summarizes the strategies for reducing liability risk as reported in the literature.

Table I
Issues to address for liability reported in the literature

Issue	Literature
Having a well-defined/complete devpmt. process	[13] [14]
Knowing and following relevant guidelines	[3] [14] [15]
Document process/guidelines and adherence to it	[3] [16] [17] [15]
Establishing safety/reliability procedures	[3] [14] [18]
Establishing a measurement program	[16]
Doing (requirements) traceability	[17] [19]
Doing testing and bug-tracking	[14] [20] [21]
Limiting exposure to liability with contracts	[3] [22]

A key concern in the literature is a discussion of the potential risks involved in legal disputes and how to mitigate these risks. In case of a legal dispute, courts will typically try to determine a baseline by looking at the state-of-the-practice and standard/duty of care [15] [23] [16]. Jones describes the typical course of action as follows: "software expert witnesses are hired to prepare reports and testify about industry norms for topics such as quality control, schedules costs, and the like. ... The expert reports produced for lawsuits attempt to compare the specifics of the case against industry background data for topics such as defect removal efficiency levels, schedules, productivity, costs, and the like" [19].

Consequently, Cusumano says that "companies delivering software that exceeds the bounds of common industry practice are vulnerable to penalties" [23]. In this case, it is important that the company can show that it follows general guidelines (e.g., professional codes of conduct/ethics/practice) as well as applicable (safety) standards. Aiken et al. report the following experiences: "Evidence is mounting that public [codes of conduct] serve as standards for evaluating the performance and determining the responsibilities not only of IEEE members, but IT professionals in general. ... Following a [codes of conduct] is one way to ... insulate contracting parties from potential legal liability" [15].

Generally, legal risk is already reduced if the development process for a system encompasses "all reasonable steps" and if "good engineering" principles are followed [13]. The system developer also has to document the employed process and principles so that it is possible to prove that all activities in the process have been followed and the corresponding work products have been produced; obviously, "liability will not be avoided by instituting procedures which are ignored" [3]. Importantly, design alternatives and tradeoffs need to be documented as well [17]. Generally, "documentary evidence is persistent and not easily dismissed" whereas "total reliance on human testimony ... is a very risky strategy" [17].

Jones emphasizes "requirements traceability" (i.e., backtracking of requirements from code and other deliverables) in the domains of defense applications and embedded software because such systems "often have serious legal and liability issues associated with them" [19, p. 462]. DeMarco and Lister state that "organizations that cannot or do not measure themselves in a fairly systematic way are always at a huge disadvantage in litigation. ... Metrics is one of the ... major subjects on which virtually all litigations turn" [16].

The development process should incorporate state-of-the-practice techniques such as they can be found in the relevant guidelines. For instance, Aiken et al. say that "courts, juries, and arbitration panels are finding that failure to follow generally accepted public standards for design and testing of software are grounds for seeking damages" [15]. Palermo recommends to build such techniques into the process as discrete milestones [3]. Turner and Khosmood suggest that for each step in the process there should be negligence analysis/research as well as dedicated evidence generation, archiving, and traceability [17]. They also propose to address and integrate liability issues into the process via "the creation of a database where important legal constraints are linked to specific development aspects that address them" [17].

C. Liability Aspects

To get a holistic picture, we look at both (1) aspects outside of the software development process that have an impact on it, and (2) aspects within the development process itself.

We consider the following aspects. For each of the aspects we provide sample questions that are derived from our main research question.

Organizational structure: Are concerns for liability reflected by dedicated roles and positions within the company's organization? For example, is there a legal expert within the company and/or has the company identified and established ties with external sources for legal council if the need arises?[1]

Business process: Are there established business processes that explicitly address liability concerns? For example, is there an opportunity for legal experts to provide input and influence decisions? Does the company conduct liability-related risk analysis/assessment? Is there an established workflow for the reporting and analysis of incidence reports? Does the business process address how to "interface" software development with other interdependent artifacts such as hardware and documentation?[2]

Use of external guidelines and codes: Is there awareness and use of guidelines, codes or standards that are potentially related to liability? Examples of generally applicable documents are the ACM Code of Ethics and Professional Conduct and the IEEE Code of Ethics. Examples of more specific documents are various (safety) standards [7].

Use of internal guidelines and codes: Has the company developed internal guidelines, codes or standards that address liability? Such internal documents could be based on external documents and internal expertise and are tailored to the needs of the company.

Development process: How are liability concerns considered or addressed during software/system development? Specifically, what are the activities, roles, and workproducts of the software development process that address or impact liability? For example, is there (dedicated) documentation that identifies liability concerns? Are liability issue traceable across workproducts?

[1]Jones identifies more than 100 different kinds of specialists in large software organizations, among them "litigation support specialist" [19].

[2]Documentation that is not in sync with the software's capabilites may be a liability risk [24].

Tool/method support: Is software development supported by tools or methods that address or impact liability? Examples of methods that could be applied are: formal verification [3]; bug tracking (of safety/liability-critical defects) [14]; fault injection [10]; safety analyses such as hazard analysis, fault tree analysis, and failure modes and effects analysis [18] [25]; and code/test coverage analysis tools (coverage monitor) [20].

III. Questionnaire: Design and Results

The fundamental goal of the questionnaire was to establish whether the issue of liability is indeed a concern for companies in the domain of safety-critical mechatronic systems. Furthermore, the questionnaire gave feedback about the individual person's awareness and involvement regarding liability. The questionnaire is available as PDF at http://bit.ly/liab-q.

The first part of the questionnaire asks about a few selected characteristics of the company and the person in question. The second part asks about aspects of liability as identified in Section II-C. The questionnaire contains 17 items, which are all multiple choice answers and closed questions (i.e., answerable with a short phrase). We chose this approach because it minimizes the effort of the responding person (and also simplifies subsequent analysis).

A. Characteristics of Respondents and Companies

Answers to our questionnaire cover 9 respondents from 6 different companies. To protect confidentiality, we do not give details about the companies, but all of them are of significant size (both personnel and budget), operate and sell their products internationally, and develop safety-critical software for mechatronic systems. Respondents described the products that they are concerned with in their work as follows: earth moving vehicles; embedded systems for automotive; vehicular; train, locomotive or metro; safety critical flight controls; and control equipment for aircraft. According to respondents the products of all but one company require certification, where certification is performed in 3 cases by units external to the product development (e.g., FAA/EASA, TÜV, or another independent part of the same company) and in 2 cases internally via self-certification.

We also asked *(Q13) What safety-related regulations and standards affect software development in your company?* Only one respondent said that there are no relevant standards. Table II summarizes the respondents' answers by domains. The responses reveal that there are domain-specific standards such as EN 50128 for railway applications and ISO 15998 for earth-moving machinery. Respondents in the vehicular domain (4 people from 3 different companies) cited IEC 61508, which is a general safety-related standard; specific interpretations of this standard are EN 50128 (railway) and ISO 26262 (automotive). Even though these standards focus on safety concerns, they are potentially applied in liability litigation for determining the expected state-of-the-practice in industry.

Table II
Standards cited by questionnaire's respondents

Domain	Standards
vehicles	IEC 61508, 61511, 62061; IS0 5010, 13849, 15998, 26262; EN 474, 954; ATEX, EMC, Low Voltage, Machinery directives;
aviation	FAR Title 14; EASA CS-25, Part-21; SAE ARP4754, ARP4761; RTCA DO-178B, DO-254; applicable FAA CAST position papers; applicable EASA Certification Review Items
train	EN 50126, 50127, 50128

Originally, our intent was for each company to question persons that cover all of the the following roles: legal council, business manager, safety expert, project leader for software development, and software developer. These roles are covering the relevant stakeholders of the case study's context (cf. Section II-A). However, it turned out that employees are very cautious and thus obtaining information was more difficult and work-intensive than envisioned. Indeed, an established relationship of trust was essential for getting answers. In the end, we were able to cover the following roles (self-assessment of respondents, multiple roles permitted): safety expert (6), business manager (1), technical manager (3), and developer (2). All respondents of the survey function at least in a role that is interfacing between software development (technical) and other non-technical areas (managerial/legal). Thus, we are confident that the socio-technical environment in which companies operate is sufficiently reflected by the responses. A potential bias is introduced by the fact that all respondents are located in Sweden.

We asked respondents about the number of years they have been working in their current role in industry. Responses ranged from 1 to 19 years with an average of 8.7 years. We also asked the following question: *(Q14) How would you rate your overall knowledge of software liability?* On a five-point scale the respondents selected: expert knowledge (none), advanced knowledge (2 people, 22%), some knowledge (6 people, 66%), very little knowledge (1 person, 11%), no knowledge (none). Thus, all respondents are aware of the concept of liability and the average rating is slightly better than some knowledge. Since none of the respondents is in the legal profession, it is not surprising that nobody claimed expert knowledge.

B. Responses Concerning Liability

We asked a number of questions to assess how companies address liability risks. All questions in this section have a *Don't know* option. The reported percentages disregard *Don't know*s and blank/unchecked entries.

One approach for companies is "offloading" of legal risks via an insurance that covers the cost incurred by legal fees and penalties. However, from personal communication

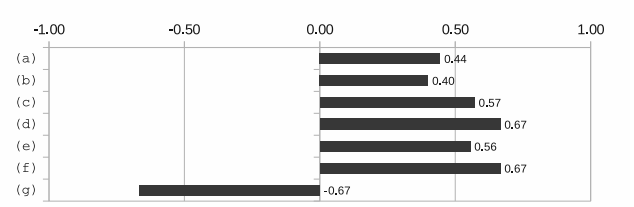
with a software expert at a large German subcontractor to the automotive industry we learned that insurance is often not practical; this is in line with Heckman's observation that "liability insurers are reluctant to write policies [for software], and the few available contain extensive limitations and come with prohibitively high premiums" [9]. When asking *(Q6) Does your company have insurance that covers liability claims?* only 1 respondent (33%) answered with yes, 2 (66%) with no, and the other 6 did not know. It is interesting that several people with a non-developer role and with many years of experience did not know whether their company actually has insurance.

A basic approach to risk mitigation is to learn from previous incidents. When asking *(Q12) Are incidents where the company's products have caused injury, or potentially could have caused injury, reported and analyzed?* 6 respondents (100%) affirmed and the other 3 did not know. One respondent substantiated that "our customer[s] have procedures to record and analyze every incident [and] we have an agreement that they involve us whenever needed."

Legal risk is reduced if a company can demonstrate that it follows certain guidelines—internal and/or external ones. When asking *(Q11) Are there company-internal codes of conduct/practice that relate to liability?* 5 respondents (71%) answered with yes, 2 (29%) with no, and the other 2 did not know. One "no" respondent pointed out that the company has internal codes, but they do not explicitly address liability. When asking *(Q10) Does the company request of you to follow external codes of conduct/practice that relate to liability?* 2 respondents (33%) answered with yes, 4 (67%) with no, and the other 3 did not know. Thus, it seems companies are more relying on internal guidelines rather than external ones. Interestingly, no respondent did refer to ethical guidelines even though these are arguably relevant in liability cases.

Another strategy to minimize risk is to seek guidance from legal experts for software development practices that minimize legal exposure. When asking *(Q7) Does the company obtain legal advice that impacts or provides input to the way software is developed?* 3 respondents (37%) said that in-house legal services are involved while 4 (50%) said that there is no legal input. Only 1 respondent (13%) reported that both in-house and external legal advice is sought. Given that appropriate technical documentation is crucial as legal defense (cf. Section II-B) we were surprised that for at least half of the companies there is no input from the legal into the technical area.

Besides the technical area, we did ask whether the business area addresses liability risks. When asking *(Q8) Are there elements in the company's business processes that explicitly address liability concerns?* 6 respondents (86%) said yes, 1 said no (14%), and the other 2 did not know. When asking *(Q9) Are legal experts involved in the decision-making in the company's business processes?* 7 respondents (100%) said yes, and the other 2 did not know. These responses clearly indicate that legal concerns are reflected in the way businesses are operating.



X-Axis scale: -2 = strongly disagree; -1 = disagree; 0 = neutral; 1 = agree; 2 = strongly agree

Figure 2. Averages of responses for Q15 (a)–(g)

To better understand how respondents assess their company's performance with respect to liability, we did ask them *(Q15) How strongly do you disagree/agree with the following statements?*:

(a) *My company has a safety culture.*
(b) *My company learns from previous liability lawsuits.*
(c) *Liability issues* related to software *are an important concern within my company.*
(d) *In my company software is developed according to the applicable safety-related rules and regulations.*
(e) *In my company software is developed following (internal) rules that minimize liability risks.*
(f) *In my company I receive training/education that addresses safety issues.*
(g) *In my company I receive training/education that addresses liability issues.*

Results are summarized in Figure 2. For statements (a)–(f) the respondents clearly tend towards agreement, indicating that liability and safety concerns are indeed recognized by companies. However, agreement is rather weak, indicating that companies could increase effort in addressing these concerns. Statement (g) shows that there are deficiencies in educating personnel about liability issues (compared to safety issues). Further research could explore what level of legal expertise (for each role) is desirable from a company's perspective.

Next we asked for the "things" given in Table III: *(Q16) In your opinion, how important is each of the following things in your company's ability to deal with liability concerns?* In the table, the averaged responses are represented with black bars to give a visual impression of the variations among the "things" that we asked for. Also in the table, depending on the average ratings we subjectively identified three groups of "things" with highest, middle and least importance. Of highest importance is safety culture (b), a concept which is already recognized by Leveson's model (cf. Section II-A). Internal guidelines (d) are seen as more important than external ones (c). Somewhat surprisingly, legal council (a) ranks rather low; same for IT support (g) and tool support

Table III
GROUPING OF AVERAGES OF RESPONSES FOR Q16:
In your opinion, how important is each of the following things in your company's ability to deal with liability concerns?

Importance	"Thing"	Averaged Responses (scale: 1–5) †
highest	(b) Safety culture	4.67
	(e) Risk analysis/assessment with respect to liability	4.29
	(d) Internal guidelines, codes, standards and other documents	4.11
middle	(c) External guidelines, codes, and standards	4.00
	(f) Elements in the software development process that address liability concerns	4.00
least	(a) Legal council	3.56
	(h) Tool support for software development that assist liability-related tasks	3.17
	(g) IT support for managerial tasks that relate to liability	2.40

† Average based on 5-point Liker scale: 1 = not important at all; 2 = slightly important; 3 = moderately important; 4 = important; 5 = very important

Table IV
GROUPING OF AVERAGES OF RESPONSES FOR Q17:
In your opinion, how important or unimportant are the following techniques in software development to mitigate liability issues?

Impo-rtance	Software developing technique	Averaged Responses (scale: 1–5) †	Used?	
			yes	no
highest	(f) Safety analyses (e.g., hazard, fault tree, and effect analysis)	4.88	100%	0%
	(b) Bug tracking (of safety/liability-critical defects)	4.63	100%	0%
	(a) Formal verification	4.50	43%	57%
	(g) Software testing independent from software development	4.50	100%	0%
middle	(i) Traceability of liability-related requirements	4.40	80%	20%
	(c) Architectural/design reviews	4.25	100%	0%
	(d) Code reviews	4.25	100%	0%
least	(e) Fault injection (software testing)	4.00	100%	0%
	(h) Code/test coverage analysis	3.88	100%	0%

† Average based on 5-point Liker scale: 1 = not important at all; 2 = slightly important; 3 = moderately important; 4 = important; 5 = very important

(h), which is in contrast to German et al.'s "believe that there is a strong need for techniques and tools that support developers in coping with legal issues from a technical point of view" [26].

Software engineering offers a range of techniques for which there is evidence that they have a positive impact on the software's quality. For safety-critical systems it is expected that such techniques are used. To get an assessment of how practitioners judge the effectiveness of certain techniques we asked *(Q17) In your opinion, how important or unimportant are the following techniques in software development to mitigate liability issues?* Table IV shows the techniques under question—which have been identified based on the relevant literature and the authors' domain knowledge—along with the respondents' average ratings. Analogous to Q16, we subjectively group the responses into three categories depending on their relative importance.

All techniques are rated quite high: all but one (h) are rated at least as being "important." Thus, all techniques are potential candidates that should be considered. In contrast to the other three techniques with highest importance (f/a/g), bug tracking (b) is rarely mentioned in the literature in the context of safety or liability. Interestingly testing techniques (e/h) rank lowest, and formal verification (a) ranks high, but is not seen as a panacea. We also did ask whether a technique is applied by the respondents' companies (Table IV, column "Used?"). Besides traceability (i) and formal verification (a) all techniques are used within all companies. Formal verification sticks out as the least used with 43%, indicating that there are still significant adoption hurdles for this technique even in safety-critical domains.

C. Limitations and Future Work

Our results have the typical threats to validity that are often associated with questionnaires, in particular, potential selection bias, and questions that are vague, ambiguous or suggestive in wording. Due to the limited sample size there may be low (statistical) significance.

Since we decided to have a questionnaire with closed questions, responses may lack desirable details. For instance, we can report that the majority of companies has internal guidelines that address liability (Q11), but we are lacking details of the guideline's structure and content and how they are leveraged in the business and software development processes. In such cases, the questionnaire is a useful tool to confirm that further probing is indeed worthwhile.

It is a concern that the number of questionnaire respondents is rather low, but it is still sufficient to obtain first results and valuable feedback for future research. Rather than increasing the sample size of the questionnaire, in future work we want to conduct *semi-structured interviews* to obtain further details. The questionnaire results will be useful for selecting interview candidates. For example, one respondent has a high number of *Don't know* answers even though he is a mature safety export with 8 years of experience due to the fact that he only recently started

working for his current company. Interviews also provide the opportunity for clarifications of (seemingly) incongruous responses. For example, one respondent said that internal legal advice is used (Q7), while another respondent for the same company answered this is not the case. For these respondents there is also disagreement about the presence of internal documents (Q11). Lastly, interviews provide an opportunity to explore questionnaire results in more depth. For instance, it would be interesting to know why some companies employ formal verification while others do not.

Future work should deepen the understanding of the relationship between liability and other quality attributes. While liability should be part of a portfolio of risk-mitigation strategies to ensure the long-term survival of a company that operates in the safety-critical domain, it should not be (ab)used as a strategy to neglect quality attributes. One respondent sees a "rigorous safety culture" as the foundation for his organization: "We really don't worry about liability after contract signature. We worry about safety, reliability and availability, in that order. Focusing on liability ... [is] an insufficient mission to make an organization work."

IV. Concluding Observations

In this paper we have explored the issues surrounding liability for software in the domain of safety-critical mechatronic systems with the help of a questionnaire involving 9 respondents from 6 companies. The questionnaire's primary goal was to establish whether liability concerns do merit further study. This is affirmed by the following evidence: responses indicate that liability is a known concept that influences both software and business processes. Especially, companies tend to observe external guidelines and augment them with internal guidelines to minimize liability risks.

Thus, for researchers that are often primarily concerned with the technical and engineering side it is important to understand that legal issues such as liability have an impact on them in the sense that legal issues impose constraints and requirements on the software process and the software itself. In other words, technical advances cannot be implemented in a legal vacuum. We believe it would be highly desirable to conduct more research on how (software development) processes, techniques and tools can be augmented to address liability risks in a more systematic manner.

References

[1] *IEEE Trial-Use Supplement to IEEE Standard for Information Technology—Software Reuse—Data Model for Reuse Library Interoperability: Intellectual Property Rights Framework*. IEEE Computer Society, Jun. 1999, withdrawn.

[2] K. Mykytyn, P. P. Mykytyn, and C. W. Slinkman, "Expert systems: A question of liability?" *MIS Quarterly*, vol. 14, no. 1, pp. 27–42, Mar. 1990.

[3] C. J. Palermo, "Software engineering malpractice and its avoidance," *3rd IEEE International Symposium on Software Reliability Engineering*, pp. 41–50, Oct. 1992.

[4] S. Dowlatshahi, "The role of product safety and liability in concurrent engineering," *Computers & Industrial Engineering*, vol. 41, no. 2, pp. 187–209, 2001.

[5] L. Hoffmann, "Risky business," *Communications of the ACM*, vol. 54, no. 11, pp. 20–22, Nov. 2011.

[6] C. Ackermann, R. Cleaveland, S. Huang, A. Ray, C. Shelton, and E. Latronico, *1st International Conference on Runtime Verification (RV 2010)*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2010, vol. 6418, ch. Automatic Requirements Extraction from Test Cases, pp. 1–15.

[7] M. Åkerholm, R. Land, and C. Strzyz, "Can you afford not to certify your control system?" *iVTinternational*, Nov. 2009, http://www.ivtinternational.com/legislative_focus_nov.php.

[8] F. E. Zollers, A. McMullin, S. N. Hurd, and P. Shears, "No more soft landings for software: Liability for defects in and industry that has come of age," *Santa Clara Computer & High Technology Law Journal*, vol. 21, no. 4, pp. 745–782, 2005, http://www.chtlj.org/sites/default/files/media/articles/v021/v021.i4.Zollers.pdf.

[9] C. Heckman, "Two views on security software liability: Using the right legal tools," *IEEE Security & Privacy*, vol. 1, no. 1, pp. 73–75, Jan./Feb. 2003.

[10] J. Voas, G. McGraw, L. Kassab, and L. Voas, "A 'crystal ball' for software liability," *IEEE Computer*, vol. 30, no. 6, pp. 29–36, Jun. 1997.

[11] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Journal on Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, Apr. 2009.

[12] N. G. Leveson, *Engineering a Safer World*, Jul. 2009, http://sunnyday.mit.edu/book2.pdf.

[13] J. Cosgrove, "Software engineering and the law," *IEEE Software*, vol. 18, no. 3, pp. 14–16, May/Jun. 2001.

[14] C. Kaner, "Software liability," 1997, http://www.kaner.com/pdfs/theories.pdf.

[15] P. Aiken, R. M. Stanley, J. Billings, and L. Anderson, "Using codes of conduct to resolve legal disputes," *IEEE Computer*, vol. 43, no. 4, pp. 29–34, Apr. 2010.

[16] T. DeMarco and T. Lister, "Both sides always lose: Litigation of software-intensive contracts," *CrossTalk*, vol. 13, no. 2, pp. 4–6, Feb. 2000, http://www.stsc.hill.af.mil/crosstalk/2000/02/demarco.html.

[17] C. S. Tuner and F. Khosmood, "Rethinking software process: the key to negligence liability," *5th IASTED International Conference on Software Engineering and Applications*, Aug. 2001, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.4208.

[18] M. Hecht, "The role of safety analyses in reducing products liability exposure in "smart" consumer products containing software and firmware," *IEEE Reliability and Maintainability Symposium (RAMS 2003)*, pp. 153–158, Jan. 2003.

[19] C. Jones, *Software Engineering Best Practices*. McGraw-Hill, Oct. 2009.

[20] C. Kaner, "Software negligence and testing coverage," 1996, http://www.badsoftware.com/coverage.htm.

[21] C. S. Tuner, D. J. Richardson, and J. L. King, "Legal sufficiency of testing processes," *SAFECOMP'96*, 1996, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.1464.

[22] J. Armour and W. S. Humphrey, "Software product liability," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-93-TR-13, Aug. 1993, http://www.sei.cmu.edu/reports/93tr013.pdf.

[23] M. A. Cusumano, "Who is liable for bugs and security flaws in software?" *Communications of the ACM*, vol. 47, no. 3, pp. 25–27, Mar. 2004.

[24] C. Kaner, "Liability for defective content," *22nd ACM International Conference on Design of Communication (SIGDOC '04)*, pp. 145–151, Oct. 2004.

[25] F. Bott, A. Coleman, J. Eaton, and D. Rowland, *Professional Issues in Software Engineering*, 3rd ed. Taylor & Francis, Aug. 2000.

[26] D. M. German, J. H. Weber, and M. D. Penta, "Lawful software engineering," *ACM Workshop on Future of Software Engineering Research (FoSER 2010)*, pp. 129–132, Oct. 2010.