2018 44th Euromicro Conference on Software Engineering and Advanced Applications

Safety-Critical Systems and Agile Development: A Mapping Study

Rashidah Kasauli[1,2], Eric Knauss[1], Benjamin Kanagwa[2], Agneta Nilsson[1] and Gul Calikli[1]

[1]Chalmers | University of Gothenburg, Sweden
[1]*rashida@chalmers.se,{eric.knauss, agneta.nilsson, gul.calikli}@cse.gu.se*
[2]Makerere University, Uganda
[2]*bkanagwa@cis.mak.ac.ug*,

***Abstract*—In the last decades, agile methods had a huge impact on how software is developed. In many cases, this has led to significant benefits, such as quality and speed of software deliveries to customers. However, safety-critical systems have widely been dismissed from benefiting from agile methods. Products that include safety critical aspects are therefore faced with a situation in which the development of safety-critical parts can significantly limit the potential speed-up through agile methods, for the full product, but also in the non-safety critical parts. For such products, the ability to develop safety-critical software in an agile way will generate a competitive advantage. In order to enable future research in this important area, we present in this paper a mapping of the current state of practice based on a mixed method approach. Starting from a workshop with experts from six large Swedish product development companies we develop a lens for our analysis. We then present a systematic mapping study on safety-critical systems and agile development through this lens in order to map potential benefits, challenges, and solution candidates for guiding future research.**

***Index Terms*—Safety-critical systems, agile, continuous integration, continuous delivery, continuous deployment, systematic mapping study**

I. Introduction

A system is safety-critical if its failure can cause financial loss, damage to the environment, injury to people and in some cases, loss of lives [8], [3]. The use of software in safety-critical systems has continuously increased to an extent where software failures can impair system safety. Examples in the automotive domain include software in a break system, which in case of failure could result in unacceptable hazards, but also active safety functions that override driver behavior in certain situations to avoid a crash. Similar examples can be found in other domains (e.g. railway [12], avionics [2])

Safety-critical systems (SCS) are heavily regulated and require certification against industry standards by the relevant governing body [13]. For that reason, they have to be designed with safety in mind. However, as opposed to the hardware where general safety design principles have been incorporated into standards, the standards for development of safe software are still developing. These standards, for instance ISO26262 for automotive, IEC61513 for nuclear, IEC62304 for medical and DO-178B for avionics aim to ensure that best engineering practice is followed and are often based on traditional, plan-driven approaches [NHR13][1]. This is related to the common criticism that agile methods neglect upfront analysis with a focus on scenarios for describing requirements [9]. Because the standards are written like that and because people think that agile methods are not suitable, SCS development is most often characterized by highly-structured, plan-driven approaches such as V-model or waterfall [GPM10].

While agile methods have significantly changed the way most software is developed, safety-critical software was widely excluded from this trend [9]. Arguments for this exclusion relate to the need for documenting safety cases and the need to analyze and specify safety requirements upfront, both thought to be in conflict with agile values. Thus, companies have different approaches to develop software, depending on whether it is safety-critical or not (e.g. allowing faster time to market of non-critical software such as infotainment) [7]. Recently, however, the situation has changed through potentially disruptive trends that significantly increase the need for short development cycles and quick time to market. Complex dynamic systems-of-systems require components of different vendors to interact at runtime and the dynamic nature of such environments can make continuous deployment a necessity for product success and for maintaining functional safety, e.g. in the automotive domain [10]. The huge attention given to autonomous driving and intelligent vehicles has lead to a jump in complexity of SCS. Companies like Tesla have demonstrated how the ability to deploy new functions and explore their performance in the field can yield huge advantages. Software companies such as Google, Apple, and Amazon have pushed in the automotive market, increasing the need to develop competencies in continuous software engineering. Slowly, companies developing SCS realize the competitive advantages that agility can provide [CWR10].

Existing reviews on agile development and safety [4], [CWR10] mostly focus on determining whether agile methods can improve embedded systems development [4] in general and on how these agile methods are being adopted in these environments. Kaisti et al.'s study [4] does not particularly focus on safety-critical systems whereas Cawley et al. [CWR10],

[1]In this paper, we use numbered references for regular citations and abbreviated style for primary papers of the mapping study.

978-1-5386-7383-6/18/$31.00 ©2018 IEEE
DOI 10.1109/SEAA.2018.00082

with a focus on regulated environments, found few works indicating a low but growing adoption of agile methods. Thus, it is about time to investigate the state of supporting agile and continuous development of safety-critical software.

In order to enable future research in this important area, we present in this paper a mapping of the current state of practice based on a mixed method approach. Starting from a workshop with experts from six large Swedish product development companies, we develop a lens for our analysis. The aim of this mapping study is to evaluate and present the combination of agile development methods and SCS. Our goal is to provide a map that can enable future research and method development for researchers and practitioners. Specifically, we map suggested benefits, challenges, and solution candidates as well as their relation to up-front analysis, just-in-time activities of agile teams, and infrastructure to support these. We aim to answer the following research questions.

RQ1: *What research exists about agile development of SCS?*
RQ2: *What are the key benefits of applying agile methods and practices in SCS development?*
RQ3: *What challenges exist with agile development of SCS?*
RQ4: *What solution candidates (e.g. principles and practices) promise to address challenges with respect to agile development of SCS?*

Thus, this paper contributes an overview of existing research, potential benefits, challenges, and solution candidates of agile development of SCS. We discuss our findings in three dimensions: The necessity of up-front analysis, the ability to shift effort into just-in-time activities, and the potential of better infrastructure to support these. We expect this contribution to be of value for practitioners aiming to optimize their processes as well as for researchers who may base future research on our synthesis.

II. Methodology

In order to provide the required context while answering our research questions, two complimentary activities have been used; a cross-company workshop and a mapping study. This section provides an elaboration on these two processes.

The Cross-Company Workshop: We started to get interested in the interplay of agile methods and SCS based on ongoing collaborations and research projects on continuous integration and deployment. Through this existing network, we invited experts responsible for adoption of continuous integration, continuous deployment, and agile methods as well as safety experts from large Swedish companies to a workshop on SCS in continuous software engineering. During this workshop, representatives from six companies attended. We presented a very preliminary screening of literature, followed by presentations from industry participants. We then had an open discussion through which we arranged the information presented and agreed on critical aspects to be further investigated. From this activity, we derived a lens for conducting our mapping study to guide our research. We use this lens to provide an overview of research in the field, specifically focusing on identifying *proposed benefits*, *challenges*, and *potential solutions* and to discuss these findings based on their relevance to *upfront analysis* (i.e. activities to be done before agile teams can continuously add value through agile development methods), *just-in-time activities* (i.e. activities done during agile development sprints), and *long lasting infrastructure beyond a single project*.

TABLE I
Number of papers found, excluded and included

Inclusion/exclusion criterion	Number of documents	
	Excluded	**Included**
Search from 2001 to 2017		1986
Limit results to subject *Computer Science* in Scopus	−1420	546
Exclude documents without authors	−81	465
Exclude non-english documents	−6	458
Exclusion based on title and abstract *- Document mentions safety but not in the agile development context.* *- Document discusses agile without a particular focus on SCS.* *- The subject area was not software engineering or development.* *- The document is not peer reviewed.*	−391	69
Criteria based on full text *- Exclude documents where full text could not be obtained.* *- Exclude documents to which an extended version was found and could be included.* *- Exclude documents that do not allow extraction of context information.* *- Include documents presenting an experience report.* *- Include documents presenting a technical solution paper or an experiment.* *- Include documents presenting a case study with students or practitioners.* *- Include other documents (e.g. short papers) that include industry experience.* *- Include papers obtained by snowballing given they meet the above criteria.*	−35	**33 + 1**

The mapping study: The key steps in the mapping study followed guidelines by Petersen et al. [11] while also borrowing some concepts from the established method of systematic literature reviews [5]. Kitchenham and Charters [6] outlined three phases of performing systematic reviews, i.e., planning, conducting and reporting the review. These phases were later simplified by Pertersen et al. [11] into five stages that include: defining research questions (see Section 1), conducting a search for primary studies, screening primary studies based on inclusion/exclusion criteria, classifying the primary studies, and data extraction and aggregation. These are explained in the subsections that follow.

A. Search strategy

Basing on our research questions, we selected the major search terms; "safety-critical systems" and "agile development". We then adapted the PICO (Population, Intervention, Comparison and Outcome) criteria proposed by Kitchenham and Charters [6] to construct the search terms used in our final search. With respect to *population*, we include terms related to safety-critical systems. These are synonyms or terms related to the major term and include safety-critical software, safety, safety critical, regulated, regulation or software intensive. With respect to intervention, we included terms relating to agile development and agile methods. These include agile, agility, continuous delivery, continuous deployment, continuous integration, scrum, xp, agile method, agile process and agile practices. Specifically, we relied on a literature review on agile methods [1] to select most commonly used search terms. We did not use the term 'software development' in our search since

it was too specific and limiting, even though it is the context in which we are working. We do not aim to compare methods and therefore did not use the comparison criteria nor did we use the outcomes criteria since the scope of the outcomes of this study is hard to limit. A pilot literature search was then performed to verify whether omission or addition of one or more search terms could lead to a decrease or increase in the documents returned. Search terms that did not add any new documents were dropped and others kept. After several iterations, the following search string was defined:

TITLE-ABS-KEY(agile OR agility OR "continuous integration" OR "continuous deployment" OR "continuous delivery" OR scrum OR "agile practices" OR xp) AND TITLE-ABS-KEY("safety-critical systems" OR "safety-critical software" OR safety OR "safety critical" OR regulated OR regulation OR "software intensive")

B. Inclusion and Exclusion criteria

Using the search string obtained, we performed an automated search on Elsevier Scopus (www.scopus.com), a database indexing many relevant venues and journals. The search returned 1986 papers which were then limited to the 'Computer Science' subject in Scopus leaving 546 papers. From these, all duplicates, editorials and panel paper were removed leaving 465 papers. The papers that were included were those that presented any kind of empirical study dealing with application of agile methods in a safety-critical environment, which were written in English and which were published from 2001, when the agile manifesto was launched, to 2017.

On the other hand, studies were excluded if they were pure discussion and opinion papers, duplicates (e.g. conference paper extended into journal), or if their main focus was not agile development in SCS. A summary of the selection criteria and as well as the percentage of the paper excluded and included based on our criteria are shown in Table I.

After removing the 6 non-english papers, the first author went through the titles and abstracts of all remaining 458 papers to determine their relevance for the study. The titles and abstracts were taken together since from the preliminary search, we realised that some studies' titles are not that explicit and could lead to elimination of relevant studies. In this step 391 papers were excluded, since their title and abstract were clearly not about agile software development and SCS. In order to increase reliability of our study, we had a random sample of 20 papers rated by three of the other authors and computed an inter-rater agreement (Fleiss Kappa $\kappa = 0.595$). All disagreements were discussed and resolved among the researchers, leading to an agreed result of 69 papers. Then, the first two authors went through the full texts of these remaining 69 papers collaboratively and selected 33 papers based on the inclusion and exclusion criteria in Table I, to which we added one more paper identified through snowballing [14] for consideration (indicated with $+1$ in Table I), resulting in a total of 34 papers.

C. Data extraction and Synthesis

With respect to our research questions, we then extracted metadata, practices used, principles followed, challenges faced, and (potential) benefits from the 34 selected papers based on a predefined template for data extraction presented in Table II. This template allowed us to extract all the information we need for analysis into a MS Excel spreadsheet. The data obtained was analyzed both quantitatively and qualitatively. We used the themes agreed during the workshop with company experts, while basing a thematic approach, to group the results obtained into the upfront, just-in-time and infrastructure. For synthesis, the metadata was summarised and proposed benefits, challenges and potential solutions were collated, analysed, and categorized among the researchers.

TABLE II
TABLE: DATA EXTRACTION TEMPLATE

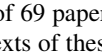
Data item	Description
Title	The title of the paper.
Authors	The full names of the authors.
Year	Year of publication.
Venue	Name of publication venue.
Publ. type	Journal, conference or workshop.
Summary	The main content of the paper.
Benefits	The benefits of agile development in SCS.
Challenges	Problems faced with agile development in SCS.
Practices	The practices used in that safety + agile environment.
Principles	The principles they follow if any.

D. Limitations and Threats to Validity

Our literature search was limited to the Elsevier Scopus database. Even though Elsevier claims that Scopus is the most comprehensive database of peer-reviewed research abstracts[2], additional databases could have provided more relevant studies. However, this was mitigated by snowballing, where the citations of the included papers were reviewed for consideration, leading to one additional paper. In order to mitigate potential misinterpretations of papers, we relied on more than one researcher in each key step of our research. Where appropriate, we calculate inter-rater agreement. Despite these efforts, we may have overlooked relevant papers or misinterpreted them in our synthesis. We aimed for a transparent description of our method to allow for recovery if this should be the case.

III. FINDINGS

A. RQ1: Existing research about agile development of SCS

Based on our search terms in combination with the defined inclusion and exclusion criteria, we selected 34 papers[3]. As shown in Table III, we did not select any paper before 2006 and there has been a *significant increase of papers* () in 2016 and 2017, supporting our assumption that agile for SCS becomes increasingly important. While we did find a number of articles in journals and magazines (5) as well as conference papers (22), most of them are quite short. Together with a considerate number of workshop papers (7) this suggests that the research field is still young. In Table III, we also give a rough characterization of the main concern of the selected

[2]http://www.elsevier.com/elsevier-products/scopus

[3]All papers: https://docs.google.com/spreadsheets/d/19s4aTpB0Yy38mnYUh1F1TiQqPZEdHD1ZYP8VdC2QQdE/edit?usp=sharing

papers of each year, based on concepts highlighted in the abstracts and keywords. While the first papers in 2006-2010 were very positive about agile methods, consecutive papers discuss more and more difficulties. From 2011 on, papers aim to systematically analyse and overcome such obstacles. We observe proposals to enhance (plan-driven) methods for developing SCS with agile practices, but also suggestions to enhance agile methods with concepts from safety standards.

B. RQ2: Key benefits of applying agile methods to SCS

With respect to RQ2, we derived key benefits from the papers and grouped them into the following categories:

Efficient use of available information: Agile methods advocate responding to change over following a plan and encourage developers to use available information to start or continue their work [SM16b]. Even early sprints aim to deliver working software which helps to identify and address technical risks, e.g. with respect to safety cases, early [SW13]. This promises to remove the need for heavy upfront design [WW16].

Improved stakeholder involvement: In SCS development, coordination and understanding between the relevant stakeholders is of utmost importance. Agile methods enable continuous communication between different stakeholders [MSL16], thus allowing them to contribute to development [SM16b]. This keeps the team better aligned with stakeholders and ensures that user needs and intended use are well understood [SW13], increases customer orientation [HWS17], and strengthens the trust between stakeholders [Hee14].

Improved safety culture: In an agile setting, the available information can be used to get an early start on the safety analysis [SM16b]. This helps in identifying possible barriers to safety which also makes it easier to discover and correct faulty system requirements [MSL16]. The practice of frequent integration of software and hardware elements supports early identification of potential issues [RHJS09]. The early focus on safety improves the level of safety awareness within the team and among stakeholders, thus improving the safety culture.

Improved management of changing requirements: Agile methods tend to focus on features during development. Together with the iterative nature of agile methods, this significantly reduces requirements churn [RHJS09] and provides teams with the ability to manage new and/or changing requirements [HWS17]. While most safety requirements may not change that often, it is important to understand the impact a change, for example in functional requirements, would have on safety. This enables teams to re-plan their sprints basing on the most recent understanding of the system under development and its requirements [MSL16].

Improved prioritisation: Agile methods generally give the advantage of better prioritisation during the handling or managing of changing requirements [SHMH14]. With prioritisation, focus is put on the highest value feature first, which leads to more thorough definition and validation of the important requirements [SW13]. This allows to put safety related features into the center of attention when developing SCS.

Mapping of functional and safety requirements: Studies propose maintaining specific backlogs for safety and functional requirements such that the development takes safety into context from the start. This practice supports a mapping between the functional and safety requirements [MSL16] and seamless integration of safety and software engineering [VW17].

Reduced costs: Agile methods encourage simple designs which in turn produce simpler software with reduced development and maintenance cost [MSL16]. The high response to requirements' changes that agile methods advocate helps to reduce the amount of rework [MMC14] and shortens overall time spent on development as well as the cost that would be spent on redoing the work [RHJS09], [HWS17], [GŁ12]. High development costs are a big challenge in developing SCS and the ability to reduce cost and lead time for bringing new safety-critical functions on the market will allow to increase the speed at which organizations learn on how to make such functions even safer.

Better test cases: The iterative development allows the team to only prepare and maintain documents that are needed either for development or certification [MSL16]. This also produces a lower number of tests, but with better focus on their intent, making it easier to understand and debug [RHJS09].

Improved quality: SCS development is a quality matter which is usually checked through rigour in testing. In agile methods, the continuous availability of working software and early response to changes facilitates ongoing testing, which reduces risk to product quality [RHJS09], [SW13], [GŁ12]. Rottier and Rodrigues report that the use of iterative development together with Test Driven Development (TDD) produced fairly consistent high quality code for their product [RR08].

Improved opportunities for reuse: Good adaption of agile methods helps building new frameworks that can be reused in later phases or projects, which will improve efficiency [MSL16], [RR08]. Such frameworks can also enable the reuse of test cases and reduce the overall workload [Wol12].

C. RQ3: Challenges with agile development of SCS

Several of our selected papers identified challenges relating to use of agile methods in SCS development, which we group and summarize into the following themes:

Difficult to manage knowledge flow between many stakeholders: There are many stakeholders in SCS development, including other engineering disciplines, sub-system suppliers, users, and external bodies such as certification authorities [PGC$^+$11], [GŁ12]. Managing communication among these stakeholders is challenging. While agile methods may improve stakeholder involvement (see above), stakeholders of SCS are used to communication through documents. The lack of focus on documentation in agile methods may lead to unclear safety documentation, thus hindering effective communication between the different stakeholders [WRW17], [WBW17].

Safety standards facilitate a waterfall mindset: Safety standards are prescriptive and often described with a waterfall-based process in mind [NHR13]. With this mindset, regulated domains often value effectiveness of waterfall processes over

TABLE III
OVERVIEW OF PAPERS

Year	Count	Journal	Conference/WS	Topics
2017	5		EuroSPI [DKS+17], ICSE [VW17], Profes [WRW17], XP [HWS17], XP WS [WBW17]	Hazard analysis, Safety Assurance, Fitting Safety into Scrum, S-Scrum, Documentation, Safety Story/Epic, Continuous Safety-Builds, Continuous Delivery
2016	9	Crosstalk [McM16]	FedCSIS [ŁG16], ICSSP [TMLB16], RAMS [MSL16], SAFECOMP [SM16a], XP [HHS+16], SE WS [WW16], XP WS [DK16], [SM16b]	Agile and CMMI, Agile Principles in Safety Frameworks, Safety Principles in Scrum, Safe Scrum, Early Safety Analysis, Generic Failure Modes, Domain Specific Fault Trees, Compliance, Incremental Design, Safety Case, safety guided design, safety validation
2015	1		ICCNEEE [AETO15]	systematic iterative approach, safety modular decomposition, safety argument
2014	4		eChallenges [KKK14], SAFECOMP [SHMH14], SPICE [MMC14], XP [Hee14]	agile (in-process) change impact analysis, hybrid: plan-driven and agile, documentation in agile, quality assurance in agile, compliance by design
2013	4	BI&T [SW13]	ICSE [FSOO13], Profes [NHR13], SERENE WS [GŁ13]	human factors, good documentation, certification cost, component reuse, iterative processes, agile and regulation, conformance
2012	4	(CSCI [GŁ12])[1]	SPICE [MMC12], FormSERA WS [Wol12], ISSREW WS [JLP12]	risks of agile, documentation, traceability, regulatory compliance, up front planning, managing multiple releases, formal methods, requirements analysis, change management
2011	2	IJCCBS [PGC+11], SPE [GEI+11]		can agile processes be applicable to SCS? empirical process control, right amount of ceremony
2010	2		AGILE [GPM10], LESS [CWR10]	minimal up front design, iterative development, Lean/Agile adoption for SCS, hybrid processes
2009	1		AGILE [RHJS09]	agile is best suitable for FDA regulated medical devices
2008	1		AGILE [RR08]	test automation, requirements validation, reporting, performance metrics
2006	1		XP [WBHV06]	agile helps with changing requirements of SCS, effectiveness of agile practices changes over time

[1]) Paper added through snowballing since not included in scopus results.

the flexibility of agile methods [FSOO13]. For example, many safety standards demand that one person must not create and review the same artifact [NHR13], [JLP12], [KKK14], while agile methods favor cross-functional teams. This mismatch of the mindset causes risks to potential compliance [SW13].

The waterfall mindset also shows a *strong focus on documentation*. In SCS, documentation is required for showing regulation compliance [MMC12] and is the preferred way of communication with certification bodies [PGC+11]. Documentation is also the primary evidence for traceability [FSOO13]. Agile methods are often reported to suffer from this focus on documentation [WBW17], [KKK14], [FSOO13], [PGC+11], [TMLB16], [Hee14] as extensive documentation will diminish advantages of agility while lack of documentation will lead to insufficient traceability.

Lack of trust in agile methods: Since agile methods do not give clear guidelines for project monitoring and control, it is hard to determine the sufficient level of evidence to present [DK16] for certification. Thus, it appears difficult to estimate effort and through the "unstructured nature" of agile methods, the amount of effort for delivering a feature seems to be underestimated [RR08]. The maturity of agile methods is hard to compare to CMM and ISO standards [PGC+11] and many managers are not convinced about business benefits of agile methods [FSOO13]. There is also doubt about whether agile methods can provide sufficiently rigorous testing [PGC+11].

Upfront planning: Safety standards suggest upfront design, hazard identification, and analysis. With agile methods, it is hard to determine how much time is enough for upfront planning [MMC12], [WRW17] and consequently difficult to identify all of the ways in which software can contribute to system level hazards up front [DK16]. Thus, there is the perception that planning in agile methods is insufficient for SCS development [WBW17], [FSOO13]. The aim for short upfront design and for analysing requirements just in time during iterations puts time pressure for determining the safety requirements and makes it difficult to evaluate the quality of the safety arguments [GPM10], [WRW17] which could impede certification [GŁ12]. Even more, attempts to include safety in agile can shift focus from customer value towards verification and validation efforts [SM16a].

Flexibility vs. Safety: Agile methods do not provide practical guidelines for change impact analysis [SHMH14], yet they provide flexibility in managing requirements. Every update therefore calls for relentless testing and strict configuration management [HWS17] and un-coordinated software changes can lead to increased complexity as the project progresses [WBHV06]. In addition, it is difficult to manage requirements that change iteratively [Hee14].

D. RQ4: Solution candidates (e.g. principles and practices) for challenges with respect to agile development of SCS

We present solutions collected from the selected papers in Table IV. Some of these solutions relate to enhancing agile, others to complementing safety, which emphasizes the aim of these works to bring both worlds closer together. We organized the proposed solutions in different categories, starting with *principles* that were suggested to provide guidance for agile development of SCS. These are rather abstract guidelines or goals to strive for when implementing an agile way of working with SCS. In addition, we identify proposed solutions with respect to *process and release planning* and *roles*. We then continue with more concrete practices that were proposed in literature, which we grouped in relation to *testing and continuous development*, *regular meetings*, and broader *safety engineering practices*.

TABLE IV
SOLUTION CANDIDATES IN LITERATURE

Solution cand.	Description
	Principles
Customer involvement at all levels	**On-site customer** [JLP12] should be part of hazard analysis, safety analysis, SSRS requirements phase, sprint reviews [DK16], [SM16a] at all levels of product development [SM16b], [DKS+17], [SW13]. **Product owner** can act as the on-site customer [FSOO13]. **Frequent contact with the customer** [Hee14] hand **regular communication** [WBHV06] allows team to react on the specific needs of the customers.
Risk management	Identify high level risks in initial phase [RR08] and consider risks during development [KKK14], [RR08]. Ensure that software is truly safe by employing a risk-based approach to planning [PGC+11], testing [KKK14], verification [PGC+11] and validation [FSOO13].
	Process and Release planning
Iterative / incremental development	Develop components iteratively [GPM10], [PGC+11], [MMC12], [HHS+16] in fixed and short iterations delivering functional software [RHJS09], [WBHV06]. Do iterative safety analysis [VW17], [AETO15], have an incremental safety validation plan [SM16a], [RR08] and safety case [SM16a].
Backlog management	Team should maintain a groomed, refined, and prioritised backlog [GŁ13], [McM16], [GEI+11], [SM16a] with two parts: one for functional requirements and one for safety requirements [AETO15], [DK16], [MSL16], [WW16], [RR08], [Wol12], [HHS+16]
User stories	Used for product definition [RR08], [KKK14], high level requirements [HWS17], upfront planning [MMC12], and as basis for safety analysis [MSL16]. Refine with use case diagrams or textual use cases [SM16b].
Essential upfront plan	Do architectural design and hazard/safety analysis up front [AETO15], [GPM10], [DKS+17]. Use hazard lists or checklists from literature [SM16b]; perform analysis of previous failure reports for process hazard analysis on the user stories [SM16b]. Use FMEA [NHR13] for decision support [SHMH14].
Formal change control and prioritisation	Implement formal change control process [RR08] with upfront lightweight CIA [SHMH14] to assign quality scores to all requirements [DKS+17] and to prioritize requirements [FSOO13], [SW13]. Maintain CIA report (issues raised and resolved) [SM16b], [SM16a]. Formalise design reviews for validation and verification [RHJS09]
	Roles
Self-organising teams	Teams are self-organized and empowered to manage daily tasks of producing software on their own [SW13], [MMC14]. Rely on collective code ownership [WBHV06] to allow the whole team to help and keep track of who did what [NHR13].
Expert knowledge	Include expert knowledge in team [GŁ13], [GŁ12]: a QA expert [DKS+17], safety experts [SM16a], [WW16], [WRW17] or nominated safety team member [DK16].
	Testing and Continuous Development
Reviews	A team of peers with assigned roles performs code reviews [GEI+11], [FSOO13] or more formal technical reviews [AETO15].
Acceptance and unit testing	Do unit tests [WBHV06], [RHJS09] and acceptance tests [WBHV06], [MMC14], [PGC+11], [GŁ13] once a number of iterations have been completed [MMC12]. Apply Test driven development (TDD) to establish high test coverage [JLP12], [WBHV06], [MMC14], [HWS17], [WW16], [Wol12], [HHS+16], [CWR10], [RR08] and increased regression testing [McM16]. Tests should be automated [McM16], [RR08], risk-based [KKK14], continuous and extensive [GEI+11], [GŁ12], and should include safety tests [VW17].
Continuous Build	Apply continuous integration [JLP12], [WBHV06], [KKK14], [HWS17], [PGC+11], [FSOO13], [HHS+16], [SW13], [GEI+11] supported by automation [RHJS09] and include continuous safety builds [VW17]. Strive for continuous compliance [FSOO13] in order to facilitate continuous delivery [HWS17], [FSOO13].
Coding standard	Keep high coding standards that all developers should adhere to [JLP12], [HWS17], [GŁ13], supported by pair programming [CWR10], [JLP12], [WBHV06], [MMC14], [PGC+11], [Wol12] and refactoring [HHS+16], [JLP12], [PGC+11], [Hee14], e.g. with help of refactoring stories [FSOO13]
	Regular meetings
Daily meetings	**Daily meetings** [FSOO13], [HHS+16] to provide *visibility of safety requirements satisfaction status* [DK16] and *promote collective ownership* [RHJS09]. **Daily stand-ups** provide *feedback, communication, and coordination* to manage technical and organizational *dependencies* [WBHV06]. Establish **weekly goals** and review them at the end of each week during the daily meeting to *focus on team objectives* [RHJS09].
Sprint review	*Demonstrate to product owner* what has been done by the team [SHMH14], [Wol12], [FSOO13], [HHS+16], [McM16], [WBW17] and to *create awareness* of activities among team members [Wol12]. **Involve all relevant stakeholders** [DK16]. **Include hazard analysis** (ibid) and acceptance criteria in review [WBW17], [SW13], [MMC14] and implement it as *independent process* for good quality assurance [KKK14].
Planning meetings and retrospectives	Create *software development plan* [KKK14] based on features [RHJS09] at the **planning meeting** [SHMH14], [HHS+16]. Document chosen *modelling approach*, *implementation language*, *development environment* and *assessment tools* [DK16] and *determine safety requirements* [WRW17]. Combine sprint planning with **retrospective** [SM16b], [RHJS09], [McM16] to empirically *improve estimations* [FSOO13].
	Safety Engineering Practices
Traceability support	Use Traceability to support certification as well as claims that requirements have been met [MMC12]. Use appropriate tools for automation [FSOO13]. Use trace information to determine which tests need to be rerun [SM16b].
Standard Operating Procedures (SOPs)	Rely on SOPs to improve quality management with clear guidelines to developers [KKK14] and use them as management records for achieving some compliance objectives in agile [HWS17]. The records should include safety standards the team should be familiar with [KKK14] and short written policies for documentation, review and testing, developed by the team [KKK14].

E. Synthesis of Findings

In Table V we give an overview of key benefits, challenges, principles, and practices discussed in literature and in Fig. 1 we show how the selected papers are distributed over these categories. The majority of work relates to just-in-time activities, such as analysis and management of functional or system safety. Fewer selected papers relate to upfront analysis, which in many publications is taken for granted. The smallest amount of papers relates to long-term aspects and infrastructure that lasts beyond an individual project.

It is noteworthy that solution candidates dominate for upfront and just-in-time aspects. In contrast, we found relatively fewer papers relating to solutions with respect to long-term and infrastructure aspects, especially in relation to the amount of challenges found in this category. We found only few papers that discuss a trade-off between upfront analysis and just-in-time analysis or attempt to push more activities from the upfront phases into continuous development.

This result resonates well with the discussions during our initial workshop. Our industry participants highlighted the following challenges that we did not yet found sufficiently supported in the selected papers.

Upfront: Our industry participants saw *benefits* in upfront analysis for agile development of SCS, e.g. with respect to reducing dependencies and confining safety-critical or secret software (e.g. due to IP constraints). *Challenges* relate to a

TABLE V
SUMMARY OF LITERATURE

Benefits	Total Pri. Papers
Improved stakeholder involvement	5
Reduced costs	5
Improved quality	4
Efficient Use of available information	3
Improved safety culture	3
Improved opportunities for reuse	3
Improved management of changing requirements	3
Improved prioritisation	2
Mapping of functional and safety requirements	2
Better test cases	2
Challenges	
Strong focus on documentation	9
Upfront planning	9
Safety standards to facilitate a waterfall mindset	6
Lack of trust in agile methods	5
Flexibility vs Safety	4
Difficult to manage knowledge flow between stakeholders	4
Practices and principles	
Acceptance and unit testing	18
Coding standard	14
Continuous build	11
Backlog management	11
Iterative development	10
Sprint review	10
Planning meetings and retrospectives	9
Formal change control and prioritisation	8
Expert knowledge	7
Customer involvement at all levels	7
User stories	6
Essential upfront plan	6
daily meetings	5
Self-organising teams	4
Traceability support	3
Risk management	3
Reviews	2
Standard operating procedures	2

lack of experience with continuous development of existing products in contrast to developing new products.

Just-in-time: Industry participants saw *benefits* with respect to managing continuously changing dependencies and requirements. The ability to do continuous assessment and certification was seen as a real enabler. Also, agile documentation, i.e. focusing on the essential, product-related information was seen as a potential benefit. *Challenges* relate to constructing the big picture from local information, quality assurance and assessors with waterfall mindset, and the question whether product or process based evidence for safety would be the better choice. With respect to *solutions*, they showed big hopes in applying

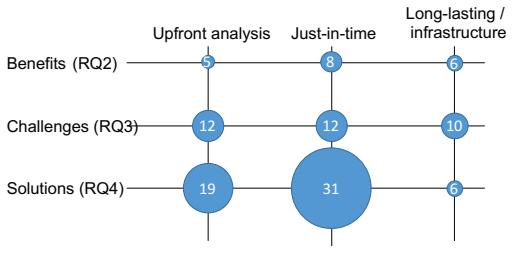


Fig. 1. Bubble chart with numbers of papers per category.

specification by example and hardening sprints.

Long-term/infrastructure: In this category, our participants saw the biggest need for new concepts. Potential *benefits* could include investment in infrastructure with long-term benefits, organization-level assessment, tools to check dependencies, and semi-automation for traceability. The biggest *challenges* were seen in a pay-per-product mentality, lack of traceability, too many layers of requirements (making it impossible to keep safety requirements up-to-date in continuous development), certification of tools (which will slow down their evolution), old assessment frameworks, and lack of trust in agile maturity.

IV. Discussion and Conclusion

In this paper, we provide a systematic mapping of agile development of SCS. We contribute an overview of research papers discussing experience from industry from 2001 to 2017 in the hope that this will enable future research in this area. Our synthesis is based on iterative analysis of selected papers and discussions in a workshop with experts from six major Swedish product companies. We found that potential benefits, challenges, and proposed solutions relate to upfront, just-in-time, or long-term and infrastructural aspects. While companies continue to struggle in all three areas, we see a need for future research specifically in two areas: a) Investigation of trade-offs between effort done upfront and just-in-time together with guidelines on how to shift more effort into just-in-time analysis; b) Investigation on how to establish beneficial infrastructure for long-term support.

Acknowledgements: We thank all participants in our workshop for their great support, deep discussions, and clarifications throughout this work whenever needed. This work was supported by Software Center (www.software-center.se) and the SIDA BRIGHT project.

Primary Papers for the Study

[AETO15] A.A. Abdelaziz, Y. El-Tahir, and R Osman. Adaptive software development for developing safety critical software. In *Proc. of Int. Conf. on Computing, Control, Networking, Electronics and Embedded Systems Eng. (ICCNEEE)*, pages 41–46, Khartoum, Sudan, 2015.

[CWR10] Oisín Cawley, Xiaofeng Wang, and Ita Richardson. Lean/agile software development methodologies in regulated environments–state of the art. In *Proc. of Conf. on Lean Enterprise Software and Systems (LESS)*, pages 31–36, Helsinki, Finland, 2010.

[DK16] Osama Doss and Tim Kelly. Addressing the 4+1 software assurance processes within scrum. In *Proc. of XP Scientific Workshops*, Edinburgh, Scotland, UK, 2016. 5 pages.

[DKS+17] Osama Doss, Tim Kelly, Tor Stålhane, Børge Haugset, and Mark Dixon. Integration of the 4+1 software safety assurance principles with scrum. In *Proc. of European Conf. on Software Process Improvement (EuroSPI)*, pages 72–82, Ostrava, Czech Republic, 2017.

[FSOO13] Brian Fitzgerald, Klaas-Jan Stol, Ryan O'Sullivan, and Donal O'Brien. Scaling agile methods to regulated environments: An industry case study. In *Proc. of 35th Int. Conf. on Software Engineering (ICSE)*, pages 863–872, 2013.

[GEI+11] Kevin Gary, Andinet Enquobahrie, Luis Ibanez, Patrick Cheng, Ziv Yaniv, Kevin Cleary, Shylaja Kokoori, Benjamin Muffih, and John Heidenreich. Agile methods for open source safety-critical software. *Journal of Software: Practice and Experience (SPE)*, 41(9):945–962, 2011.

[GŁ12] Janusz Górski and Katarzyna Łukasiewicz. Assessment of risks introduced to safety critical software by agile practices-a software engineer's perspective. *Computer Science (CSCI)*, 13(4):165–182, 2012.

[GŁ13] J. Górski and K. Łukasiewicz. Towards agile development of critical software. In *In Proc. of Int. Workshop on Software Engineering for Resilient Systems (SERENE)*, pages 48–55, 2013.

[GPM10] Xiaocheng Ge, Richard F Paige, and John A McDermid. An iterative approach for development of safety-critical software and safety arguments. In *Proc. of AGILE Conf.*, pages 35–43, Nashville, TN, USA, 2010. IEEE.

[Hee14] Lise Tordrup Heeager. How can agile and documentation-driven methods be meshed in practice? In *(XP)*, pages 62–77, Rome, Italy, 2014.

[HHS+16] Geir K Hanssen, Børge Haugset, Tor Stålhane, Thor Myklebust, and Ingar Kulbrandstad. Quality assurance in scrum applied to safety critical software. In *Proc. of Int. Conf. on Agile Software Development (XP)*, pages 92–103, Edinburgh, Scotland, UK, 2016.

[HWS17] Geir K. Hanssen, Gosse Wedzinga, and Martijn Stuip. An assessment of avionics software development practice: Justifications for an agile development process. In *Proc. of Int. Conf. on Agile Software Dev. (XP)*, pages 217–231, Cologne, Germany, 2017.

[JLP12] H. Jonsson, S. Larsson, and S. Punnekkat. Agile practices in regulated railway software development. In *Proc. of 23rd Int. Symp. on Software Reliability Engineering, Workshops (ISSREW WS)*, pages 355–360, Dallas, TX, USA, 2012.

[KKK14] Wolfgang Kuchinke, Christian Krauth, and Töresin Karakoyun. Agile software development requires an agile approach for computer system validation of clinical trials software products. In *Proc. of eChallenges Conf.*, pages 1–8, Belfast, UK, 2014.

[ŁG16] K Łukasiewicz and J. Górski. Agilesafe - a method of introducing agile practices into safety-critical software development processes. In *Proc. of Federated Conf. on Computer Science and Information Systems (FedCSIS)*, pages 1549–1552, 2016.

[McM16] Paul E. McMahon. Cmmi the agile way in constrained and regulated environments. *Journal of Defense Software Engineering (Crosstalk)*, JulAug:10–15, 2016.

[MMC12] M. McHugh, F. McCaffery, and V. Casey. Barriers to adopting agile practices when developing medical device software. In *Proc. of Int. Conf. on SW Process Impr. and Capability Determ. (SPICE)*, pages 141–147, Palma de Mallorca, Spain, 2012.

[MMC14] Martin McHugh, Fergal McCaffery, and Garret Coady. An agile implementation within a medical device software organisation. In *(SPICE)*, pages 190–201, 2014.

[MSL16] Thor Myklebust, Tor Stålhane, and Narve Lyngby. An agile development process for petrochemical safety conformant software. In *Proc. of Annual Reliability and Maintainability Symposium (RAMS)*, Tucson, AZ, USA, 2016.

[NHR13] Jesper Pedersen Notander, Martin Höst, and Per Runeson. Challenges in flexible safety-critical software development – an industrial qualitative survey. In *Proc. of Int. Conf. on Product Focused Software Process Improvement (PROFES)*, pages 283–297, 2013.

[PGC+11] Richard F. Paige, Andy Galloway, Ramon Charalambous, Xiaocheng Ge, and Phillip J. Brooke. High-integrity agile processes for the development of safety critical software. *Int. Journal of Critical Computer-Based Systems (IJCCBS)*, 2(2):181–216, 2011.

[RHJS09] Rod Rasmussen, Tim Hughes, J.R. Jenks, and John Skach. Adopting agile in an fda regulated environment. In *Proc. of AGILE Conf.*, pages 151–155, Chicago, IL, USA, 2009.

[RR08] Pieter Adriaan Rottier and Victor Rodrigues. Agile development in a medical device company. In *Proc. of AGILE Conf.*, pages 218–223, Toronto, ON, Canada, 2008.

[SHMH14] Tor Stålhane, Geir Kjetil Hanssen, Thor Myklebust, and Børge Haugset. Agile change impact analysis of safety critical software. In *Proc. of Int. Conf. on Computer Safety, Reliability, and Security (SAFECOMP)*, pages 444–454, Firenze, Italy, 2014.

[SM16a] Tor Stålhane and Thor Myklebust. The agile safety case. In *Proc. of Int. Conf. on Computer Safety, Reliability, and Security (SAFECOMP)*, pages 5–16, 2016.

[SM16b] Tor Stålhane and Thor Myklebust. Early safety analysis. In *Proc. of XP Scientific Workshops (XP WS)*, Edinburgh, Scotland, UK, 2016. 6 pages.

[SW13] John Schmidt and Kelly Weyrauch. Getting agile with medical device development. *Biomedical Instrumentation & Technology*, 47(3):221–223, 2013.

[TMLB16] Kitija Trektere, Fergal McCaffery, Marion Lepmets, and Grainne Barry. Tailoring mdevspice ® for mobile medical apps. In *Software and System Processes (ICSSP)*, pages 106–110, Austin (TX), USA, 2016.

[VW17] S. Vost and S. Wagner. Keeping continuous deliveries safe. In *Proc. of 39th Int. Conf. on SW Eng. (ICSE)*, Buenos Aires, Argentina, 2017.

[WBHV06] Andrew Wils, Stefan Van Baelen, Tom Holvoet, and Karel De Vlaminck. Agility in the avionics software world. In *Proc. of (XP)*, pages 123–132, 2006.

[WBW17] Yang Wang, Ivan Bogicevic, and Stefan Wagner. A study of safety documentation in a scrum development process. In *Proc. of XP Scientific Workshops (XP WS)*, 2017.

[Wol12] Sune Wolff. Scrum goes formal: Agile methods for safety-critical systems. In *Formal Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA)*, pages 23–29, Zurich, Switzerland, 2012.

[WRW17] Yang Wang, Jasmin Ramadani, and Stefan Wagner. An exploratory study on applying a scrum development process for safety-critical systems. In *Proc. of Int. Conf. on Product-Focused Software Process Improvement (PROFES)*, pages 324–340, 2017.

[WW16] Yang Wang and Stefan Wagner. Toward integrating a system theoretic safety analysis in an agile development process. In *Proc. of Software Engineering (Workshops)*, pages 156–159, Wien, Austria, 2016.

REFERENCES

[1] P. Diebold and M. Dahlem. Agile practices in practice: a mapping study. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, page 30. ACM, 2014.

[2] E. S. Grant. Requirements engineering for safety critical systems: An approach for avionic systems. In *2nd Int. Conf. on Computer and Communications (ICCC), 2016*, pages 991–995. IEEE, 2016.

[3] J. Hatcliff, A. Wassyng, T. Kelly, C. Comar, and P. Jones. Certifiably safe software-dependent systems: challenges and directions. In *Proceedings of the on Future of Software Engineering*, pages 182–200. ACM, 2014.

[4] M. Kaisti, V. Rantala, T. Mujunen, S. Hyrynsalmi, K. Könnölä, T. Mäkilä, and T. Lehtonen. Agile methods for embedded systems development-a literature review and a mapping study. *EURASIP Journal on Embedded Systems*, 2013(1):15, 2013.

[5] B. Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004.

[6] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. In *Technical report, Ver. 2.3. EBSE*. sn, 2007.

[7] M. Kuhrmann, P. Diebold, and et al. Hybrid software and system development in practice: Waterfall, scrum, and beyond. In *ICSSP*, pages 30–39, 2017.

[8] P. A. Laplante and J. F. DeFranco. Software engineering of safety-critical systems: Themes from practitioners. *IEEE Transactions on Reliability*, 66(3):825–836, 2017.

[9] B. Meyer. *Agile! The Good, the Hype and the Ugly*. Springer, 2014.

[10] P. Pelliccione, E. Knauss, and et al. Automotive architecture framework: the experience of volvo cars. *Journal of systems architecture*, 2017.

[11] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *EASE*, volume 8, pages 68–77, 2008.

[12] L. Provenzano and K. Hänninen. Specifying software requirements for safety-critical railway systems: An experience report. In *Int. Working Conf. on Requirements Engineering: Foundation for Software Quality*, pages 363–369. Springer, 2017.

[13] M. Vuori. Agile development of safety-critical software. *Tampere University of Technology*, 14, 2011.

[14] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, page 38. ACM, 2014.