Collective Intelligence-Based Quality Assurance: Combining Inspection and Risk Assessment to Support Process Improvement in Multi-Disciplinary Engineering

Dietmar Winkler[1,2] Juergen Musil[2] Angelika Musil[2] Stefan Biffl[2]

[1] SBA Research gGmbH, Favoritenstrasse 16, 1040 Vienna, Austria
dwinkler@sba-research.org
[2] Institute of Software Technology and Interactive Systems, CDL-Flex,
Vienna University of Technology, Favoritenstrasse 9/188, 1040 Vienna, Austria
{firstname.lastname}@tuwien.ac.at

Abstract. In *Multi-Disciplinary Engineering* (MDE) environments, engineers coming from different disciplines have to collaborate. Typically, individual engineers apply isolated tools with heterogeneous data models and strong limitations for collaboration and data exchange. Thus, projects become more error-prone and risky. Although *Quality Assurance* (QA) methods help to improve individual engineering artifacts, results and experiences from previous activities remain unused. This paper describes a *Collective Intelligence-Based Quality Assurance* (CI-Based QA) approach that combines two established QA approaches, i.e., (Software) Inspection and the *Failure Mode and Effect Analysis* (FMEA), supported by a *Collective Intelligence System* (CIS) to improve engineering artifacts and processes based on reusable experience. CIS can help to bridge the gap between inspection and FMEA by collecting and exchanging previously isolated knowledge and experience. The conceptual evaluation with industry partners showed promising results of reusing experience and improving quality assurance performance as foundation for engineering process improvement.

Keywords: Collective Intelligence System, Defect Detection, Engineering Process, Improvement, FMEA, Inspection, Review, Risk.

1 Introduction

In *Multi-Disciplinary Engineering* (MDE) environments, different stakeholders have to collaborate along the project course [1]. Examples for MDE environments include the engineering of automation systems, such as production automation systems, steel mills, or hydro power plants. For instance, plant planners are responsible for the basic configuration of a plant, mechanical engineers design the physical setting of the planned plant, electrical engineers provide electrical and wiring plans, and software engineers design the control software for operation [2].

In MDE projects, engineers typically follow a sequential process approach with parallel discipline-specific engineering tasks and isolated *Quality Assurance* (QA) activi-

ties [1]. Thus, engineering projects typically suffer from limited data exchange capabilities and become more error-prone and risky [3]. Fig. 1 illustrates an example of a sequential engineering process observed at an industry partner, key deliverables, related stakeholders, and isolated QA activities. Examples of isolated QA activities are reviews/inspections [4, 5], Software and System Testing [6, 7], *Failure Mode and Effect Analysis* (FMEA) [8, 9], the *Fault Tree Analysis* (FTA) [10], or the *Defect Causal Analysis* (DCA) method [1][11]. While reviews/inspections and testing focus on defect detection, FMEA, FTA, and DCA focus on assessing risks and on identifying root causes of defects. Typically, these QA approaches focus on individual engineering artifacts with limited data exchange and knowledge and experience reuse. However, knowledge and experience from method applications can provide a valuable input for improving QA methods. For example, defect lists (a key outcome of inspection) can drive the FMEA to assess related risks; candidate risks (a key outcome of the FMEA) represent knowledge that can be reused to improve inspection processes.

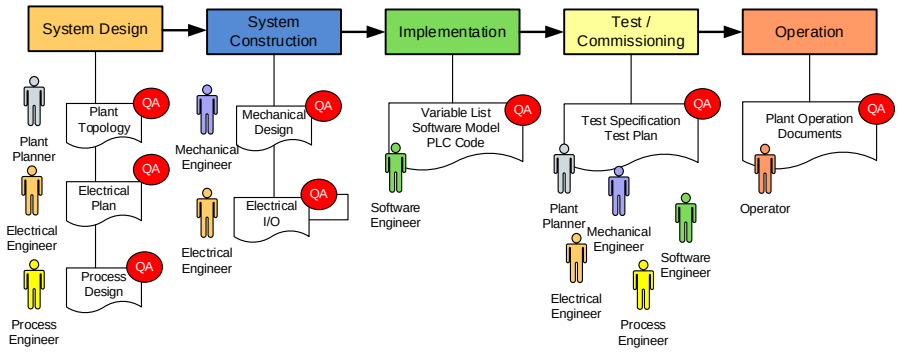

Fig. 1. Sequential Engineering Process in Multi-Disciplinary Engineering Projects [1].

However, individual QA methods are applicable for different types of engineering artifacts. For instance, reviews/inspections are well established in Software Engineering [4] and focus on early defect detection in various types of engineering artifacts, e.g., text documents, images, engineering plans, or software code. In Systems Engineering, e.g., in the Automotive Systems Domain, the FMEA is an established approach for risk assessment with focus on design, product, and process requirements [8]. Although reviews are used in Systems Engineering, more structured approaches, such as inspections, are rarely applied. Further, to the best of our knowledge, there are only limited attempts to combine different QA approaches, such as reviews/inspections and the FMEA, to gain additional benefits derived from applied methods. The combination of reviews/inspections and the FMEA can enable engineering process improvement in terms of using defect lists (derived from review/inspection approaches) as input for the FMEA; identified risks and countermeasures can be re-used in an inspection approach to improve review and inspection processes. However, main challenges include how to combine review/inspection processes and the FMEA in terms of improving QA mechanisms. More specifically, (a) how to reuse results from review/inspection in the FMEA (and vice versa) and (b) how to collect, aggregate, disseminate, and reuse engineering

knowledge coming from method applications. The reuse of engineering knowledge can improve individual methods and increase method application performance. A type of software system that could address these capabilities are *Collective Intelligence Systems* (CIS), which are a particular kind of collaborative, social platform that focus on aggregation and dissemination feedback loops of user-generated content [13]. In software engineering, CIS have been sustainably integrated as tool support in best-practice software development processes, such as bug tracking (*Jira*[1]), code reviews (*Gerrit*[2]) or wide-scale software repository reuse (*GitHub*[3]). Therefore, CIS seems to be a promising starting point to bridge the gap between the aforementioned, so far isolated approaches. This paper addresses the challenges of combining inspections and FMEA on a conceptual level to provide a mechanism for reusing knowledge in engineering projects to (a) improve the engineering product and (b) to improve methods for defect detection (i.e., inspection) and risk management (i.e., the FMEA) processes by using a *collective intelligence system*.

The remainder of this paper is structured as follows: Section 2 presents related work on reviews and (software) inspection, the FMEA, and collective intelligence systems. Section 3 presents the research issues. Section 4 introduces to the concept of collective-intelligence driven defect detection and risk management based on required capabilities. Section 5 presents an initial concept evaluation. Finally, Section 6 discusses strength and limitations of the approach and concludes the paper.

2 Related Work

This section summarizes related work on (software) inspections for early defect detection (Section 2.1), the FMEA for systematic risk assessment (2.2), and *Collective Intelligence Systems* (2.3) to capture, manage, and reuse engineering knowledge for better supporting both inspection and FMEA processes.

2.1 Reviews and (Software) Inspections

Software reviews and inspections are well-established formal defect detection approaches in Software Engineering [4] to identify defects early and efficiently. Reviews and inspections follow a defined process executed by defined stakeholders. Fig. 2 presents a common inspection process [14] including related roles.

The traditional inspection process includes six steps [14]: *Inspection Planning* (1) is based on project/quality plans or driven by a decision to conduct an inspection for a specific engineering artifact. A moderator is responsible for planning tasks, i.e., assessing inspection artifacts (e.g., based on inspection entry criteria), providing method support (e.g., reading techniques), and organizing team members and inspection activities. Depending on the experience of the team members and the complexity and novelty

[1] Software tool *Jira*: https://www.atlassian.com/software/jira
[2] *Gerrit*: https://www.gerritcodereview.com
[3] *GitHub*: https://github.com

of the engineering artifacts, an *Optional Overview* (2) meeting helps inspection team members to get familiar with the provided inspection package. Note that inspections can also facilitate knowledge exchange and learning [15]. *Individual preparation* (3) takes as input the inspection package and delivers a set of individual defect lists provided by inspectors of the inspection team. The main goal of an *Inspection Meeting* (4) is to derive an agreed team defect list from the discussion of individual defect lists. In the *Rework* (5) phase responsible authors fix reported defects in their engineering artifacts and provide updated engineering artifacts. Finally, during the *Follow-Up* (6) phase the moderator checks these modifications and decides on (a) releasing the engineering artifact or (b) scheduling another inspection cycle (i.e., re-inspection), if quality criteria are not acceptable. Reasons for a re-inspection are based on too many reported defects or critical issues that might cause risks or have an impact on other engineering artifacts.

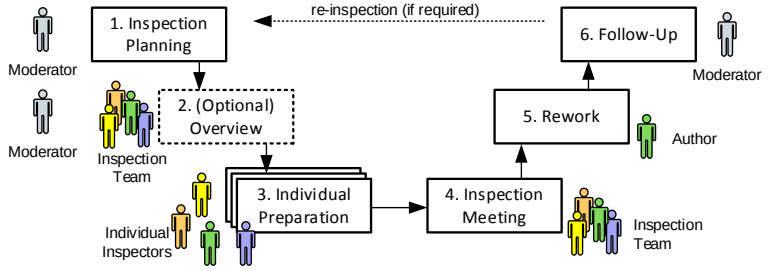

Fig. 2. (Software) Inspection process steps with related stakeholders [14].

Reading techniques are supporting guidelines that guide a team of inspectors through the reading process and support them in detecting defects in various engineering artifacts, e.g., specification documents, models, diagrams, or software code. A reading technique is a structured approach on how to review/inspect a specific engineering artifact [16, 17] and thus, represent engineering knowledge for inspection application. Several studies investigated different reading techniques and reported on strength and weaknesses in different study contexts [4, 16, 18]. A *Checklist-Based Reading* (CBR) technique approach consists of a set of sequential and domain-specific tasks that enables inspectors stepping through the inspection artifacts and report candidate defects. *Usage-Based Reading* (UBR) focuses on prioritized use cases [14] and apply business-critical scenarios for defect detection. The application of *Perspective-Based Reading* (PBR) enables defect detection from various perspectives, e.g., developer, tester, or system architect [19]. Although reading techniques are popular in software engineering, they are not widely used in MDE. In the context of MDE, perspectives seem to be a promising approach, because engineers from different disciplines can take their individual viewpoints on engineering artifacts and report candidate defects. Winkler and Biffl [20] proposed the focused inspection approach with tool support that enables the application of perspective-based reading in MDE contexts.

Beyond improving products (artifacts) in Software Engineering, inspections, results of inspection process steps represent *explicit knowledge* on engineering artifacts (e.g., entities or relationships) or the application domain (e.g., architecture best practices).

However, this explicit knowledge typically is lost after inspection and is only rarely used for improving processes or for supporting related engineering or QA activities.

2.2 Failure Mode and Effect Analysis

The main goal of the *Failure Mode and Effect Analysis* (FMEA) focuses on (a) the assessment of product reliability and (b) the early identification of risks, which can have a critical impact on the customer and use of the product [8]. FMEA team members identify and assess risks, candidate defects, and countermeasures based on system requirements, required features, and proposed solutions. During risk assessment, the FMEA team estimates probability (P), severity (S), and detectability (D) of candidate defects on a linear scale from 0 to 10[4] for every single requirement/feature and derives the *Risk Priority Number (RPN)*: RPN = P*S*D. In the context of an FMEA, probability refers to the likelihood of defect occurrence in the final product in the field; severity describes the consequences and the impact of a defect in the field; and detectability refers to complexity factors of identifying and locating the defect in the final product or artifact. The related RPN threshold values are defined in the project context to focus on the most critical issues. Based on this defined threshold value, RPNs above this value require countermeasures, RPN values below this threshold are accepted as risks and no actions are defined.

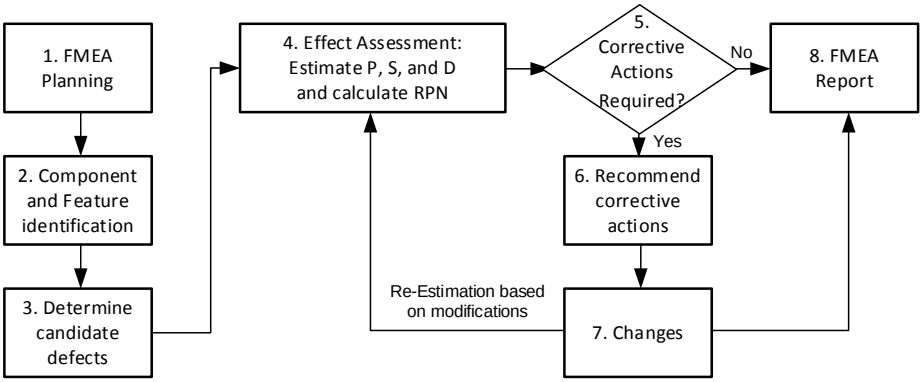

Fig. 3. Failure Mode and Effect Analysis (FMEA) process steps based on [9].

Fig. 3 illustrates the basic FMEA process based on [9]: (1) *FMEA planning* is executed by the quality or project manager based on the application domain, requirements, or expected risks including scope definition, team composition and scheduling. In FMEA workshops, the team (often key stakeholders in MDE environments) identifies

[4] Linear scale for probability, severity, and detectability: 0 stands for very low probability, severity, and detectability; 10 indicates critical probability, severity, and detectability. For example, the rating 10/10/10 means that candidate defects will definitely be in the final product (high probability) with a very critical impact (high severity) and it is very hard to identify the defect early (high detectability).

(2) *Key Components and Features* and determine (3) *Candidate Defects*. Candidate defects typically represent a list of risks and possible issues in context of the artifact or project. Experiences from previous projects often help to identify typical issues in the domain. However, this step often applies implicit experience and knowledge of experts in the FMEA team. (4) *Effect Assessment*. For every candidate defect the FMEA team determines defect probability, severity, and detectability and calculates the RPN based on expected defect effects. (5) The *Decision on Required Corrective Action* is based on the RPN, i.e., whether or not the RPN exceeds defined threshold values. (6) *Corrective Actions* (countermeasures) need to be identified and recommended by the FMEA team. (7) Implemented *Changes* require a re-assessment of the RPN. Note that this cycle could be repeated several times. Finally, the FMEA results in a (8) *FMEA Report*.

The FMEA has been successfully applied in systems engineering domains, such as automation systems, for early risk and defect assessment and prevention. However, existing experiences (from previous projects) or knowledge are typically embodied within FMEA team members but rarely made explicit for reuse in other projects or improvement of FMEA methods. CIS can help to make this implicit knowledge explicit.

2.3 Collective Intelligence Systems

Collective Intelligence Systems (CIS) are socio-technical platforms, which provide the efficient aggregation and dissemination of user-generated content and knowledge [13]. Representative examples of popular CIS are *Wikipedia*, *Twitter*, *YouTube* and the *Eclipse Marketplace*[5]. In addition, CIS possess effective self-organization capabilities, which are enabled by a characteristic feedback loop [12]. This loop coordinates the overall information flow between the platform users, thus enabling the division of labor and increased awareness of mission-relevant information.

Both inspection and the FMEA strongly rely on the interaction of human experts, who contribute to projects by applying best-practice methods. Often experiences and knowledge are implicitly embodied by experts but not explicitly expressed. Implicit knowledge hinder reuse of experiences and knowledge (a) in projects with similar method application and (b) across methods, here: inspection and the FMEA. Reusing experience and knowledge is an important foundation for engineering improvement. In context of information artifacts improvement efforts often focus on usability aspects (such as the cognitive dimensions framework [21]) to lower barriers for users to share and retrieve information. Another approach would be to address these issues on a systemic level. Thus, CIS capabilities could support human-centric activities, such as inspection and FMEA. In context of this paper, we consider a CIS as a “black box” and focus on the application of a CIS on a conceptual level for combining inspections and FMEA on process level to gain benefits from method application towards product and process improvement.

[5] *Eclipse Marketplace*: https://marketplace.eclipse.org

3 Research Issues

The isolated application of inspection for early defect detection in Software Engineering and the FMEA for early risk assessment in Systems Engineering helps to improve related artifacts in isolated phases. The combination of inspection and FMEA, powered by a CIS, aims at supporting project stakeholders in (a) improving methods based on the previous method application results, (b) making implicit process knowledge explicit for reuse purposes, and (c) gain additional benefits and synergies for method and process improvement. Therefore, we derive two research issues:

RI-1. How can a collective intelligence-based quality assurance (CI-based QA) approach support engineering process improvement in the MDE domain? The combination of inspection and the FMEA on process level can help to link results and experiences between both approaches. For instance, team defect lists (derived from the inspection process) can support guiding the FMEA process as foundation for determining candidate defects; results from FMEA application (candidate risks and defects) can support improving defect detection techniques to address these risks.

RI-2. What capabilities are required to enable CI-based QA for Inspection and the FMEA? Inspection, FMEA, and the combination of both approaches are typically human-centric application including high expert effort for inspection and FMEA processes. However, tool support aims at decreasing effort and cost, and increasing project, product, and method quality. Thus, the main question is which capabilities are needed for a tool solution, based on a CIS, to mediate the integrated quality assurance approach.

4 Collective Intelligence-Based Quality Assurance

The combination and integration of Inspection [18] and the FMEA [9] – as part of a *Collective Intelligence-based Quality Assurance* (CI-based QA) process – aims at including advantages of inspection and the FMEA, supported by a CIS. Advantages of software inspection include (a) early defect detection in various types of engineering artifacts by applying a structured process approach, (b) guidance of inspectors and teams of inspectors through the reading process by using reading techniques, and (c) making implicit knowledge explicit based on process steps, reading techniques (e.g., checklists or scenarios), and inspection outcomes (e.g., defect lists). Advantages of the FMEA include (a) early risk assessment of engineering artifacts, (b) list of components/features, related candidate risk/defect lists, RPN measures (see Section 2.2), corrective actions, and (c) process knowledge based on FMEA application process steps.

Fig. 4 presents the combined inspection and FMEA process approach with a CIS including (A) inspection process steps (left hand side); (B) FMEA process steps (right hand side); and (C) *Collective Intelligence System* (middle part of Fig. 4). The *inspection process* (A in Fig. 4) takes as input the inspection object, e.g., an architecture design document and a set of requirements (A1). Note that typical inspections focus on the analysis of inspection objects in context of a stable reference document, i.e. requirements in this case. Reading techniques (A2) provide concrete guidelines for traversing

the document under inspection, the perspective-based reading (PBR) approach in this example. Outcome of the inspection process is the agreed team defect list (A3) and engineering knowledge, represented by identified entities, relationships, typical defects, and explicit inspection knowledge (A4), such as dependencies between the architecture design and requirements or defects related to design elements. See section 2.1 for details on the inspection process. The *FMEA process* (B in Fig. 4) takes as input requirements (common for inspection and the FMEA) and the team defect list (B1), which was the output of the inspection process, for FMEA execution. Furthermore, defined guidelines for risk assessment (B2), e.g., the *fault tree analysis* (FTA) approach, are used to identify additional candidate defects, root causes, and corrective actions. Outcome of the FMEA include RPN, risks, and corrective actions (B3). Furthermore, engineering knowledge represent explicit knowledge on how to derive risks, estimation values and experiences for RPN calculation, and how to identify corrective actions. See Section 2.2 for details on the FMEA process approach.

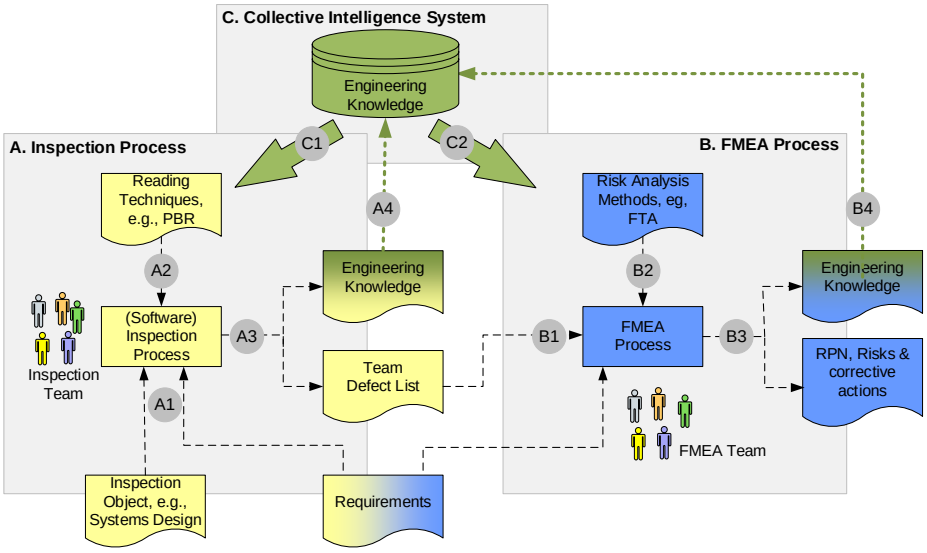

Fig. 4. Combined Inspection and FMEA process bridged with a *Collective Intelligence System (CI-based QA)* with input/output artifacts.

Finally, the *Collective Intelligence System* (C in Fig. 4) aims at collecting experiences and knowledge from inspection processes (A4) and the FMEA process approach (B4). *Semantic Web Technologies*, e.g. based on [22, 23], can integrate and link explicit experience and engineering/process knowledge aspects from collected inspection results (e.g., requirements, defects, inspection process steps, and reading technique aspects) and FMEA results (e.g., risk, candidate defects, RPN estimations, and corrective actions). Combined/integrated engineering and process knowledge is used (a) to improve inspection processes and reading techniques (C1), e.g., to better address identified candidate defects (derived from FMEA processes), and (b) to improve FMEA processes by providing knowledge from inspection processes (e.g., scenarios that can support

FTA). Knowledge and method engineers can use this experience/knowledge to improve the methods in their application domain.

However, method and process improvement (for future application) typically require time and effort to be implemented within an organization. The proposed approach gains immediate benefits that come from reusing results from previous tasks, here inspection results (i.e., team defect lists that represent real defects) in the FMEA approach to assess additional risks and identify corrective actions. Furthermore, inspection is well suited for training and learning purposes. Thus, immediate benefits can arise if new engineers have to be introduced to a project. Following the structured inspection approach, engineers will get familiar with engineering artifacts as "a by-product" of defect detection. Finally, in the MDE context, engineers are typically driven by mechanics and electrics and they are often familiar with the FMEA approach, while software inspection is not well established in this domain. Combining best practices from software engineering and automation systems development is a promising research direction to improve engineering projects and processes in MDE.

5 Tool Capabilities and Conceptual Evaluation

Inspection and the FMEA are important QA approaches in Software Engineering and the Automation Systems Domain. However, discussions with industry experts in the automaton system domain and research collaborators showed the need for process and tool support like a *CI-based QA* approach, e.g., combining inspection and the FMEA. For instance, customers of our industry partners claimed the need for FMEA for risk assessment. However, the application of a FMEA is time-consuming without capabilities for systematically reusing results or experiences from FMEA workshops. Thus, there is a clear need for providing method and tool support for FMEA execution. Software Inspection seems to be a promising approach to complement the FMEA to make implicit experience and knowledge explicit and to provide tool support for method application based on CIS, i.e., *CI-based QA*. Based on the conceptual process description (see Fig. 4), experiences with Software Inspection and the FMEA, and in discussions with our industry partners in the automation systems domain, we derived a set of capabilities needed for *CI-based QA* tool support:

Defect Detection Performance:

- Support for early defect detection and risk assessment to identify defects and risk early in the engineering process.
- Need for *effective and efficient* defect detection / risk assessment.

Risk Assessment:

- Need for *systematic* and *traceable* quality assurance processes, i.e., process steps and related outcomes have to be repeatable and traceable.
- Need for defined *responsibilities* and *roles* for method application, e.g., for planning, execution, and rework.
- Need for *guiding* less-experienced team members during method application.

Reuse of Experience and Tool Support for Engineering Process Improvement:

- Need for *reusing experiences and knowledge* from method application for engineering process improvement. This need includes the collection, aggregation, dissemination, and reuse of engineering knowledge to improve engineering methods.
- Need for *immediate improvements* of artifacts and engineering plans after method application, i.e., immediate effects of method application.
- Need for *tool support* to help inspection/FMEA teams in executing inspections/FMEA processes more effectively and efficiently.

Based on prototype evaluations and discussions with industry experts and research partners, we assessed the traditional inspection approach, the FMEA, and the *CI-based QA* process approach towards expected/needed key capabilities. Table 1 summarizes needs/capability considerations based on the initial evaluation of the process prototype and identified key capabilities towards a tool-support for CI-based QA.

Table 1. Needs/capabilities of Inspection, FMEA, and CI-based QA approach (++ strong support, o neutral or method-specific support, -- weak support).

Needs / Capabilities	Inspection	FMEA	CI-Based QA
Effective & efficient **defect detection**	++	--	++
Effective & efficient **risk assessment**	o	++	++
Systematic quality assurance	o	o	++
Traceable results	o	o	++
Defined **roles and responsibilities**	o	o	o
Guidelines for method application (methodological support)	o	o	++
Reuse of Experiences and Knowledge	--	--	++
Immediate **artifact improvements**	o	o	++
Tool support	o	o	-- [6]
Implementation/Application Effort	o	o	o

Main results of the initial evaluation showed promising results for the *CI-based QA* approach because the approach combines/integrates best practices from both application methods, i.e., method support and processes from inspection and FMEA and process interfaces in between. However, typical applications focus on isolated improvements of artifacts rather than on a comprehensive view on linked methods. Therefore, in this paper we identified key needs and required capabilities for a comprehensive tool solution. A tool solution for the *CI-based QA* approach is currently under development.

6 Discussion, Limitations, and Future Work

In this paper, we introduced a *collective intelligence-based quality assurance* (CI-based QA) concept, an integrated approach for combining best-practice Inspection and the *Failure Mode and Effect Analysis* (FMEA). On a process level, both approaches can

[6] Tool support is currently under development.

complement each other by reusing QA process artifacts, e.g., the team defect list (an important output of inspection) can help to drive the FMEA process based on identified defects in inspection objects. Both approaches, inspection and FMEA, come with comprehensive method support and guidelines, but there are limited exchange opportunities to benefit from each other. Based on guidelines and method best-practices, engineering knowledge is embodied in human experts and are not available explicitly. A CIS can help to make this knowledge explicit and available for (a) improving individual artifacts, (b) supporting individual methods for improvement, and (c) gain additional benefits from cross-method applications.

Research issue *RI-1* focuses on how a collective intelligence-based quality assurance *(CI-based QA)* approach can support engineering process improvement in the MDE domain. Figure 4 presents an integrated process approach for bundling benefits of inspection and the FMEA and support knowledge generation, aggregation, dissemination, and reuse based on a CIS. Discussions with industry experts and research collaborators found the approach promising for real-world application to improve defect detection and risk assessment based on experiences provided by CIS. However, tool support is needed to support CI-based QA. In addition, evaluations in industry contexts remain for future work.

Research issue *RI-2* focuses on required capabilities for a *CI-based QA* tool support. Table 1 summarizes the main capabilities derived from observations and discussions with industry partners and domain experts. This list of expected capabilities represents the foundation for implementing and evaluating a *CI-based QA* tool.

Limitations. Most important limitations of the approach focus (a) on limited tool support of the *CI-based QA* (currently under development) and (b) limited industrial evidence on benefits/limitations of the application of an integrated CI-based QA tool. First results in lab environments and during pilot tests at industry partners showed promising results. However, in-depth empirical studies are needed to investigate the impact of CI-based QA on defect detection and risk assessment.

Future work will include (a) definition of a *collective intelligence system* capable of supporting key capabilities, (b) implementation of the *CI-based QA* approach, (c) related pilot studies at industry partners, and (d) empirical evaluations in larger industrial contexts.

Acknowledgements. Parts of this work were supported by the Christian Doppler Forschungsgesellschaft, the Federal Ministry of Economy, Family and Youth, the Austrian National Foundation for Research, Technology and Development, and the TU Wien Doctoral College on Cyber-Physical Production Systems.

References

1. Kovalenko, O., Winkler, D., Kalinowski, M., Serral, E., Biffl, S.: Engineering Process Improvement in Heterogeneous Multi-Disciplinary Environments with the Defect Causal Analysis. In: 21st EuroSPI Conference, pp. 73-85, Springer (2014).

2. Biffl, S., Lüder, A., Winkler, D.: Multi-Disciplinary Engineering for Industrie 4.0: Semantic Challenges, Needs, and Capabilities. In: Biffl, S., Sabou, M. (eds.) Semantic Web for Intelligent Engineering Applications, Chapter 2, Springer (2016) (to appear).
3. Biffl, S., Moser, T., Winkler, D.: Risk Assessment In Multi-Disciplinary (Software+) Engineering Projects. IJSEKE, SI on SW Risk Assessment, 21(2), pp. 211-236 (2011).
4. Aurum, A., Petersson, H., Wohlin, C.: State-of-the-Art: Software Inspection after 25 years. J. of Software, Testing, Verification and Reliability, 12(3), pp. 133-154 (2002).
5. Wiegers, K.: Peer Reviews in Software: A Practical Guide. Addison-Wesley (2001).
6. Broekman, B., Notenboom, E.: Testing Embedded Software, Addison Wesley (2002).
7. Myers, GJ., Sandler, C., Badgett, T.: The Art of Software Testing, Wiley & Sons (2011).
8. Stamatis, DH.: Failure mode and effect analysis: FMEA from theory to execution, ASQ Quality Press (2003).
9. Teng, S-H., Shin-Yann, H.: Failure mode and effects analysis: an integrated approach for product design and process control. J of Quality & Reliability Mgmt 13(5), pp.8-26 (1996).
10. Ericson, CA.: Fault Tree Analysis Primer. CreateSpace Independent Publishing, (2011).
11. Kalinowski, M., Card, DN., Travassos, GH.: Evidence-based guidelines to defect causal analysis. IEEE Software, 29(4), pp. 16-18 (2012).
12. Musil, J., Musil, A., Weyns, D., Biffl, S.: An Architecture Framework for Collective Intelligence Systems. In: 12th Working IEEE / IFIP Conference on Software Architecture (WICSA), pp. 21-30, IEEE (2015).
13. Musil, J., Musil, A., Biffl, S.: Introduction and Challenges of Environment Architectures for Collective Intelligence Systems. In Weyns, D., Michel, F. (ed). Agent Environments for Multi-Agent Systems IV, Springer International Publishing (2015).
14. Laitenberger, O., DeBaud, J-M.: An encompassing life cycle centric survey of software inspection. J. of Systems and Software, 50(1), pp. 5-31 (2000).
15. Carver, J., Shull, F., Basili, V.: Can observational techniques help novices overcome the software inspection learning curve? An empirical investigation. ESE J, 11(4) (2006).
16. Kollanus, S., Koskinen, J.: Survey of Software Inspection Research: 1991-2005, Working Papers WP-40, University of Jyväskylä (2007).
17. Travassos, G., Shull, F., Fredericks, M., Basili, VR.: Detecting defects in object-oriented designs: using reading techniques to increase software quality. In ACM Sigplan Notices, 34(10), pp. 47-56). ACM (1999).
18. Biffl, S.: Inspection Techniques to support Project and Quality Management. Habilitation, Vienna University of Technology, Shaker (2001).
19. Shull, F., Rus, I., and Basili, VR.: How perspective-based reading can improve requirements inspection. IEEE Computer, 33(7), pp. 73-79 (2002).
20. Winkler, D., Biffl, S.: Focused Inspections to Support Defect Detection in Multi-Disciplinary Engineering Environments. In: 16th International Conference on Product-Focused Software Process Improvement, Research Preview Paper (2015).
21. Blackwell, A., Green, T.: Notational systems – the cognitive dimensions of notations framework, HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science. Morgan Kaufmann (2003).
22. Moser, T., Mordinyi, R., Winkler, D., Melik-Merkumians, M., Biffl, S.: Efficient Automation Systems Engineering Process Support Based on Semantic Integration of Engineering Knowledge. 16th Int. Conf. on Emerging Techn. and Factory Automation (ETFA) (2011).
23. Novak, P., Serral, E, Mordinyi, R., Sindelar, R.: Integrating Heterogeneous Engineering Knowledge and Tools for Efficient Industrial Simulation Model Support. Advanced Engineering Informatics, 29(3), pp. 575 – 590 (2015).