Towards Collective Intelligence System Architectures for Supporting Multi-Disciplinary Engineering of Cyber-Physical Production Systems

Angelika Musil, Juergen Musil and Stefan Biffl
CDL-Flex, Institute of Software Technology and Interactive Systems
TU Wien, Vienna, Austria
Email: {angelika, jmusil}@computer.org, stefan.biffl@tuwien.ac.at

***Abstract*—The engineering process of Cyber-Physical Production Systems (CPPS) involves collaboration of multiple engineering disciplines. Major obstacles arising from these multi-disciplinary engineering processes are heterogeneous representations, weak accumulation and integration of dispersed, local engineering knowledge, and required effective coordination between multi-disciplinary engineering teams across the organization. Further, heterogeneous communication channels lead to increased information sharing effort for individual team members, ill-structured knowledge representation and management, and poor discoverability of business-critical know-how. These challenges can be addressed by Collective Intelligence Systems (CIS) that enhance engineering methods and tools in large, multi-disciplinary projects. CIS help to identify important implicit, hard-to-access dispersed information and engineering knowledge, make it explicit, and promote the awareness and efficient management of this business-critical knowledge. Therefore, this paper presents a research agenda focusing on the systematic and empirically-grounded investigation of needs, basic concepts, principles, and models of CIS software architectures in particular application domains, and outlines expected results.**

***Index Terms*—Awareness, collective intelligence, collective intelligence system, coordination, cyber-physical production systems, industrie 4.0, industry 4.0, knowledge management, software architecture, systems engineering.**

I. Introduction

Cyber-Physical Production Systems (CPPS), such as industrial production plants, are typically the result of complex, multi-disciplinary engineering processes, where several engineering disciplines, such as mechanical, electrical, and software engineering, are involved and need to collaborate [1]. In these software and systems engineering environments, the work of engineers depends on inputs and results from other engineering disciplines, e.g., requirements incorporated in a range of engineering models. Therefore, it is critical to systematically support these complex knowledge-intensive tasks in multi-disciplinary engineering processes of CPPS to ensure a high-quality output, early defect detection and prevention, and to reduce operational risks of CPPS.

Major obstacles for improving multi-disciplinary engineering processes are heterogeneous representations (engineering methods, models and terminologies), weak accumulation and integration of the dispersed, local engineering knowledge that is instrumental for the development and validation of CPPS as well as required effective coordination and sharing between multi-disciplinary engineering teams in the organization [1]. The use of heterogeneous communication channels between individuals leads to an increased information sharing effort for the individual the more people are involved. There is also a lack of structured knowledge representation and management as well as effective discoverability of business-critical know-how.

These challenges with engineering CPPS at organization level can be addressed by Collective Intelligence Systems (CIS). Well-known examples of CIS comprise *Facebook*, *Wikipedia*, the collaborative Git repository hosting platform *GitHub*, and the programming Q&A platform *Stack Overflow*. As a socio-technical system [2], a CIS acts as a mediator of interactions among a user community and relies substantially on the continuous flow of contributions of the connected users within the provided services [3]. Thereby CIS provide effective, bottom-up coordination and communication capabilities thus facilitating the aggregation, management and distribution of knowledge in a coordinated way [3]. "By contributing new content individually to these systems, the users build collectively a continuously growing repository of valuable information, knowledge and data, and thus generate collective intelligence of a user community" [3] which would not be possible if each individual acted in isolated ways. Today, CIS haven been adopted in a growing number of various application contexts and domains, in particular as organization-level systems, since they provide macro- and micro-level benefits to online communities of various scales [3]. One domain where organization-level CIS have recently increased in relevance is Industrie 4.0 and CPPS [4]. In large, multi-disciplinary projects such as the engineering of CPPS, CIS have the potential to enhance (software) engineering methods and tools to overcome the existing accumulation, integration, collaboration, and communication complexities of hard-to-access dispersed information and engineering knowledge. Furthermore, CIS help to identify important implicit engineering knowledge and make it explicit as well as to provide the awareness and efficient management of this business-critical knowledge.

Nevertheless, each target application context and domain has characteristic particularities that need to be addressed by a well-suited CIS architecture design to engineer a most effective system. Despite CIS are very popular and widespread

in use, they have not been sufficiently investigated so far with focus on software architecture design. Software architects still face considerable obstacles in understanding and choosing the best fitting CIS design with features that are demanded to satisfy the organization's business needs and goals, existing workflow limitations as well as the specific application context. Feature modifications highly affect a CIS's significant capabilities and consequently its success. Such modifications have a strong impact on the system conceptually (e.g. applied rules of the system) as well as technically (e.g. used technologies). Therefore, the software architect needs a complete understanding about (1) the set of features all kinds of CIS have in common and thus are fixed core features to deal with in the architecture design, and (2) existing system variants and their effects to make better design decisions. Without this knowledge, the design decision making of software architects leaves uncertainty about the effects on the system's capabilities and behavior. A consolidated systematic knowledge base of the architectural principles would extremely improve the understanding of concepts significant for a CIS's success. Furthermore, a methodological support for architecting CIS tailored for complex application contexts and specific domains like CPPS engineering would provide useful assistance and guidelines for software architects to model these systems. These nontrivial research challenges provided the impetus for this paper which presents a research agenda focusing on the systematic and empirically-grounded investigation of needs, basic concepts, principles, and models of CIS software architectures in particular application domains, and outlines expected results.

The remainder of this paper is structured as follows: Section II provides background information about CIS. In section III we present a research agenda describing research challenges, goals and expected results. Section IV summarizes relations to existing work. Finally, section V draws conclusions.

II. Collective Intelligence Systems

Over the last decades Collective Intelligence Systems (CIS) have been established as new forms of web-based, user-contribution-driven information systems in our modern knowledge-driven society. CIS harness the collective knowledge and work of connected people by providing a web-based environment with capabilities to effectively share topic-specific information with the participating community and to discover and reuse relevant knowledge more efficiently [3]. Furthermore, CIS enable self-organizational knowledge transfer and coordination of distributed work [3] as well as ongoing maintenance of the knowledge base, thus supporting individuals to communicate and work together on complex tasks like engineering CPPS.

A central concept behind CIS is the successful emergence of Collective Intelligence (CI) by applying certain mechanisms within the system environment. CI is a well-established phenomenon researched in several fields like sociology, biology, political science, economics for decades [5]. Pierre Lévy's often cited definition describes CI as *"a form of universally distributed intelligence, constantly enhanced, coordinated in real time, and resulting in the effective mobilization of skills. [...] No one knows everything, everyone knows something."* [6]. Recently, the emergence of new information communication technologies has enabled new forms of how collective intelligence occurs [7]. Today, the new possibilities provided by the Internet and the Web 2.0 era [8] have significantly contributed to the research efforts in IT-enabled collective intelligence and the development of an increasing number of CIS in various application contexts and domains. In particular, CIS have experienced a strong growth and acceptance on organizational level like enterprise social networks and wikis, also due to support by the *Enterprise 2.0* [9] movement, which advocates the use of modern social software solutions in and between companies and their customers. A recent discussion of CI in the organizational context (CIorg) can be found in Grasso et al. [10].

III. Research Agenda for Collective Intelligence System Architecture Design Support

To provide the usefulness and general benefits of CIS in order to improve existing workflows, adequate tailoring and business-critical fit of a CIS for the respective application context, domain and organization are key success factors.

Software architects design and build CIS which are tailored towards specific needs of the stakeholders and the context of a domain or organization. But there are several ways to incorporate CIS capabilities into information systems. The definition, design and selecting of a sufficiently complete and well-fitting set of software features and functions that realize a CIS's main regulatory capabilities and address the required business-critical functionality as well as envisioned future growth paths is an enormous challenge for software architects. So far there is a lack of adequate architectural design guidelines for CIS which hinders software architects to systematically conceptualize and manage CIS-specific concerns in architecture descriptions. This circumstance increases the risk that software architects make inappropriate architectural design decisions about system features for the respective context. Current approaches, predominantly comprising trial and error or clone and own from similar successful CIS, often are responsible for missing and ill-prioritized features. Consequently, these strategies lead to ineffective systems and, in worst case scenarios, to project failures due to limited knowledge and understanding of the main CIS capabilities and a systematic design process. Therefore creating a solid understanding about CIS commonalities and variabilities is prerequisite to invent future CIS for multi-disciplinary engineering scenarios in CPPS.

Hence, one key challenge that need to be addressed by research is the *lack of architecture-relevant knowledge about commonalities and significant variabilities* among identified key elements of CIS. In addition, *missing architectural design approaches to describe significant CIS capabilities and manage the variabilities in a systematic way* as well as to *guide the design decision making for supporting context-aware*

selection of adequate feature variants makes it a hard time for a software architect to build the best fitting kind of CIS for the intended purpose and respective application context, domain and organization.

Research in this area should address these challenges by conducting a systematic empirically-grounded investigation of basic concepts, principles, variabilities and models of software architectures for CIS. In particular, research requires to provide a rigorous understanding of the architectural constructs and principles that shape a common CIS as a basis for focusing then on an analysis of *architecture variations of CIS*. A catalogue of identified commonalities and significant variabilities among CIS key features as well as a better understanding of their effects would support a more effective tailoring of a CIS for the respective application domain to better address current business process limitations, needs and concerns. Research areas, that should provide significant contributions to the design and engineering of CIS, are:

1. How, to what extent, and in which system elements do architecture-relevant variations in CIS occur?

Based on the identified challenges, the research issues here relate to specific activities that might explore certain architectural basis patterns. These patterns represent the essence of all these systems in a minimal system description which needs to be differentiated to more specific ones. An investigation of real-world CIS in a wide scope would allow identifying existing CIS variations with respect to the described architectural basis pattern, how these variations are affected by underlying system elements and design decisions, and what are strengths and weaknesses of each variation in order to better assess its potential application areas.

With respect to the CPPS domain, we need to investigate existing CIS, their design and application in more detail in the context of CPPS. In addition, CIS requirements and domain-specific concerns need to be collected in the context of engineering CPPS. These findings would allow a comparison of CPPS-specific CIS and CIS in general as well as support a better understanding of the domain-specific aspects that are important to consider during architecting.

2. How can the identified CIS families be adequately classified?

To map the characteristic features of each identified CIS variation, which branches away from the base system, groups of CIS need to be defined on the basis of shared characteristics. As a consequence, a system taxonomy can be built that supports a comprehensive classification and comparison of CIS architectural sub-styles. First observations of different kinds of CIS revealed that these systems share certain features which were altered to some extent for a single one based on made system design decisions. For example if a wiki and a microblogging platform are compared, different concerns like the time duration between contribution cycles can be noted.

Based on the findings of existing CIS in the application domain of CPPS as well as the collected system requirements and domain-specific concerns, a dominant feature set would emerge that is characteristic for CPPS-specific CIS and allow to assign this kind of systems to an identified CIS variation as well as to integrate it into the CIS taxonomy.

3. How can software architects be supported with a systematic architectural design approach that focuses on identified feature variations in CIS?

To provide systematic support for software architects in choosing a well-suited CIS variant for the respective application context and business goals as well as in the design process of the system with the required set of features, the gained architectural knowledge about the identified variants needs to be consolidated and formalized using established and practical useful methodologies and tools. A variability modeling of CIS should support software architects with variant analysis to evaluate trade-offs before making design decisions and with additional effective design methods.

In particular, to better support software architects in engineering CIS in the CPPS domain, aspects and concerns specific in the context of CPPS need to be analyzed that are important to be addressed in a CIS architecture design for process improvement. This architectural knowledge can be incorporated in a systematic architecture approach specific for this application domain.

Expected results of the research contributions should (1) provide a deeper understanding of commonalities and variabilities of CIS families and subgroups from a software architecture perspective, (2) offer better support to explore novel kinds of CIS and application areas like CPPS, and (3) offer improved design and decision making approaches with regards to the various system families for software architects who work with CIS for particular domains like multi-disciplinary systems engineering.

IV. Related Work

This section presents an overview of related work on software architecture, key concepts and collective intelligence systems.

Recent activities in the area of software architecture research have failed to provide sufficient understanding and methodological support for architecting in CIS domain. Today, research often focuses on the engineering of technical aspects. Although research fields of human computation, crowdsourcing and social computing investigate networked human groups and computing systems [11], they do not address aspects that arise from a software architecture point of view, which is although critical to design well-tailored systems. Therefore, a systematic investigation of underlying models and mechanisms for computational support of mediated social interaction and human cognitive processes is highly relevant [2] to provide a consolidated systematic knowledge base of the architectural principles of a successful CIS.

In recent years there have been efforts to understand characteristic features of CIS. Malone et al. [7] were the first who investigated foundations of CIS and identified staffing, goal, incentives and structure/process as four key elements. Lykourentzou et al. [12] introduced a CI categorization framework

which distinguishes between passive and active CIS, whereby active CIS are differentiated in collaborative, competitive and hybrid ones. Smart et al. [13] created a taxonomy of *Social Machines*, a family of socio-technical systems which subsumes CIS, that consists of eleven characteristic features. From a crowdsourcing perspective, Bernstein et al. [5] categorized them in directed, self-organizational and passive systems. A limitation that is shared among the previous systematizations is that they are broad and abstract, thus cause numerous interpretative gaps, when it comes to the conceptual and technical architecting of CIS. Current compilations of CIS research are provided by Miorandi et al. [14] and Malone et al. [5].

For emerging domains like Industrie 4.0 and CPPS, CIS are of particular interest in their role as enabling technologies. Bauernhansl et al. [4] highlighted on numerous occasions the potential of social media platforms in future cyber-physical production systems, but also the lack of their current application. Two aspects should be highlighted in particular: (1) the efficient collection and maintenance of project-independent knowledge [15], and (2) the increased computerization of physical systems and the resulting need for solution providers to organize and coordinate their internal software base. Developers of software-intensive systems have an own industrial internal software ecosystem (ISECO) [16], which has evolved internally around their platforms. Guiding platform evolution and reducing software architectural erosion as well as architectural technical debt are challenges that increase in relevance with ongoing multi-disciplinary and global distribution of development teams and the increased interdependence of created systems.

V. Conclusion

In this paper we presented a research agenda for investigating systematic architecture design guidelines for CIS that are well-tailored for a respective CPPS application context. We highlighted the importance of better understanding architecture-relevant principles and significant variabilities of CIS as well as an exploration of systematic architectural design approaches as essential research challenges that need to be addressed in future research work. The described expected results should provide contributions to the design and engineering of CIS and to a consolidated knowledge base for the CIS domain as a new promising research field. For the CPPS domain, CIS are of particular relevance to provide new approaches for efficient knowledge exchange and coordination support in multi-disciplinary engineering teams and processes in the design time. The application of CIS in CPPS should improve the aggregation, maintenance and awareness of distributed knowledge and artifacts contributed by various involved disciplines, as well as enable effective coordination and sharing of business-critical know-how. A subsequent application of the researched CIS architectural design approaches in the context of engineering CPPS by considering identified domain-specific aspects and concerns enables software architects to adequately tailor a CIS with regards to current engineering methods and tools and enhance them with required capabilities and thus overcome current challenges and improve the engineering process.

Acknowledgment

This work was supported by the Christian Doppler Forschungsgesellschaft, the Federal Ministry of Economy and Science, the Austrian National Foundation for Research, Technology and Development, and TU Wien research funds.

References

[1] R. Mordinyi and S. Biffl, "Versioning in Cyber-physical Production System Engineering – Best-Practice and Research Agenda," in *Proc. of the IEEE/ACM 1st International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS '15)*. IEEE, 2015, pp. 44–47.

[2] A. Omicini and P. Contucci, "Complexity and Interaction: Blurring Borders between Physical, Computational, and Social Systems. Preliminary Notes," in *Proc. of the 5th International Conf. on Computational Collective Intelligence Technologies and Applications (ICCCI '13)*, ser. LNCS, C. Bdic, N. T. Nguyen, and M. Brezovan, Eds., vol. 8083. Springer Berlin Heidelberg, 2013, pp. 1–10.

[3] J. Musil, A. Musil, and S. Biffl, "Introduction and Challenges of Environment Architectures for Collective Intelligence Systems," in *Agent Environments for Multi-Agent Systems IV*, ser. LNCS. Springer International Publishing, 2015, vol. 9068, pp. 76–94.

[4] T. Bauernhansl, M. ten Hompel, and B. Vogel-Heuser, Eds., *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Springer Vieweg, 2014.

[5] T. W. Malone and M. S. Bernstein, Eds., *Handbook of Collective Intelligence*. MIT Press, 2015.

[6] P. Lévy, *Collective Intelligence: Mankind's Emerging World in Cyberspace*. Perseus Books, 1997.

[7] T. W. Malone, R. Laubacher, and C. Dellarocas, "Harnessing Crowds: Mapping the Genome of Collective Intelligence," MIT Center for Collective Intelligence, Working Paper No. 2009-001, Feb. 2009, available at: http://cci.mit.edu/publications/CCIwp2009-01.pdf (last visited 01/21/2016).

[8] T. O'Reilly and J. Battelle, "Web Squared : Web 2.0 Five Years On," in *Proc. of the 6th Annual Web 2.0 Summit*. O'Reilly Media Inc., 2009.

[9] A. McAfee, *Enterprise 2.0: New Collaborative Tools for Your Organization's Toughest Challenges*. Harvard Business Press, 2009.

[10] A. Grasso and G. Convertino, "Collective Intelligence in Organizations: Tools and Studies," *Computer Supported Cooperative Work (CSCW)*, vol. 21, no. 4-5, pp. 357–369, 2012.

[11] A. J. Quinn and B. B. Bederson, "Human Computation: A Survey and Taxonomy of a Growing Field," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '11)*. ACM, 2011, pp. 1403–1412.

[12] I. Lykourentzou, D. J. Vergados, E. Kapetanios, and V. Loumos, "Collective Intelligence Systems: Classification and Modeling," *Journal of Emerging Technologies in Web Intelligence*, vol. 3, no. 3, pp. 217–226, 2011.

[13] P. Smart, E. Simperl, and N. Shadbolt, "A Taxonomic Framework for Social Machines," in *Social Collective Intelligece*, D. Miorandi, V. Maltese, M. Rovatsos, A. Nijholt, and J. Stewart, Eds. Springer International Publishing, 2014, pp. 51–85.

[14] D. Miorandi, V. Maltese, M. Rovatsos, A. Nijholt, and J. Stewart, Eds., *Social Collective Intelligence: Combining the Powers of Humans and Machines to Build a Smarter Society*. Springer International Publishing, 2014.

[15] J. Nasser, C. Maga, P. Göhner, T. Ehben, T. Tetzner, and U. Löwen, "Improved Systematisation in Plant Engineering and Industrial Solution Business - Increased Efficiency through Domain Engineering," *at-Automatisierungstechnik*, vol. 58, no. 9, pp. 524 – 532, 2010.

[16] K.-B. Schultis, C. Elsner, and D. Lohmann, "Architecture Challenges for Internal Software Ecosystems: A Large-scale Industry Case Study," in *Proc. of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE '14)*. ACM, 2014, pp. 542–552.