# Cloud Based Collaborative Software Development: A Review, Gap Analysis and Future Directions

Stanley Ewenike, Elhadj Benkhelifa and Claude Chibelushi
School of Computing and Digital Technologies,
Cloud Computing and Applications Research Lab
Staffordshire University, Stoke of Trent, UK.
Stanley.Ewenike@research.staffs.ac.uk, (e.benkhelifa, cc.chibelushi)@staffs.ac.uk.

***Abstract*—Organizations who have transitioned their development environments to the Cloud have started realizing benefits such as: cost reduction in hardware; relatively accelerated development process via reduction of time and effort to set up development and testing environments; unified management; service and functionality expansion; on-demand provisioning and access to resources and development environments. These benefits represent only a fraction of the full potential that could be achieved via leveraging Cloud Computing for the collaborative software development process. Related efforts in this area have been mainly in the areas of: asynchronous collaboration; collaboration in isolated aspects of the Software Development process, such as coding activities; use of open-source tools for contributing, improving, and managing code, etcetera. Although these efforts represent valid contributions and important enablers, they are still missing important aspects which enable a more holistic process, with solid theoretical foundation. This paper reviews this research area, in order to better assess factors and gaps creating the need to enhance the collaborative software development process in the Cloud, to better meet the pressure to collaboratively create better cloud-agnostic applications.**

***Keywords—Collaborative software development, Cloud, gap analysis, collaboration***

## I. INTRODUCTION (*HEADING 1*)

A review of related literature reveals a variety of problems and factors which act as barriers to collaboration in the collaborative software development process [17], [18], [22]. These have been broadly grouped into the following categories geographical factors, sociocultural and linguistic factors, temporal factors, management and process factors, infrastructural/technological factors, organizational factors and trust. These problems reveal a need for better and more cohesive collaboration within the collaborative software development process, with some solutions been suggested [21], [23]–[25]. Among some of the suggested ways of addressing the barriers to collaboration in software development is leveraging benefits and opportunities offered by emerging paradigms, namely, the Cloud Computing paradigm.

The prospect of leveraging the Cloud computing paradigm within the structured collaborative software development process, presents a research area of possible synergies yet to be fully exploited[1]. The real-time collaboration and efficiency opportunities offered by the Cloud promises close-knit collaboration for Cloud-based processes[2], [3]. Increased adoption of Cloud applications and servicesintroduces a noticeable shift in the way computing resources and applications are provisioned, accessed, utilized, stored and managed; and necessitates the need to explore and adapt present the collaborative software development process for the Cloud[4]–[6]. Accessing and housing software applications in the Cloud, implies a need for change in the way these applications are engineered[7], [8]. The inherent capabilities of the Cloud hold a lot of promises in the quest to address collaboration in the wider spectrum. Potentials exist for leveraging Cloud capabilities to adapt software development stages for better collaboration and efficiency[1].

This research paper reviews existing literature in the research area to better assess gaps and challenges currently existing. Literature review reveals notable increase in activity from industry in Cloud-based collaboration with a lot of emphasis in content management, sharing and storage, but less in Cloud-based collaborative software development[9].The review was based on the adapted systematic process, where a search for relevant articles in the research domain was carried out. Query strings were used in searching and retrieving literature for review usingMendeley, a reference management tool, with a large, interconnected academic database[10]. This was very useful for finding, storing, managing and correlating academic research materials and libraries. Mendeley was chosen because of its reasonably fair approximation of research databases, such as Scopus, and appears to have one of the largest databases in terms of research articles and journal coverage, and traffic [11].

## II. WHAT ISCOLLABORATION?

According to the Oxford dictionary, Collaboration is “the action of working with someone to produce or create something”[12]. Collaboration is a conceptspanning different context and disciplines, but is commonly used to refer to the act ofworking together towards a common goal[13]–[15]. Collaboration may be in either of two forms – synchronous or asynchronous; and may be based on a variety of factors – model-based collaboration, process-based collaboration, infrastructure-based collaboration, activity-based collaboration, distance-based collaboration and inter-discipline/multi-discipline based collaboration [16]–[19]. Despite the numerous definitions of collaboration as a

concept, it has often been misconstrued, and used quite interchangeably with other concepts or terms like: cooperation, communication, and coordination, depending on context [20]. Hence, for the purpose of this research, Collaboration is used to refer to - the set of activities involving: jointly working together to solve common problems, carrying out complementary activities to solve diverse problems, and all other activities geared towards achieving or accomplishing a common goal[21]. These activities could involve: building and sharing knowledge; accessing shared knowledge; working together in a shared space or distributed space, towards common goals.

## III. Overview of Cloud Computing

Many definitions for Cloud computing exists, but one of the most adapted definitions is that offered by the NIST -*"….a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*"[26]. This definition tries to capture what the Cloud is all about in unambiguous terms. It captures the five main characteristics of the Cloud, which constitute the most attractive features of the Cloud. These features: rapid elasticity, measured service, on-demand self-service, broad network access and resource pooling; represent the strengths from whence, most of the benefits attributed to the Cloud come from. This definition also captures one key point that is sometimes overlooked - the minimal effort it entails to provision services or resources.

The advent of Cloud computing has brought about an increase in the servicification of IT resources such as: Infrastructure as a Service (IaaS), Software as a Service (SaaS) and Platform as a Service (PaaS); resulting in the consumption of these resources as services on a pay-per-use basis which greatly favors organizations and companies with limited resources[27], [28]. These services are deployed either publicly, privately, in hybrid form, or as a community model. The area of software development is not left out too. Effect of these changes can be seen in the paradigm shift from use of desktop IDEs to Cloud APIs, in building software projects. Various Cloud services providers, for example, Amazon, Google, Microsoft, IBM, and a host of others, all have their own API offerings, often built on top of their IaaS offerings[29], [30].

Prior to the advent of Cloud computing, traditional data centers and IT setups often relied on architectures that could at best be described as similar to silos, making it difficult for fluid and easily scalable interactions between infrastructure, applications and data. Situations synonymous with these include: waste of resources, complex administrative and management functions, less agility and response to changing business and user needs, high costs associated with scaling, staffing, maintenance, development, operations, maintenance, and even capital for expansion[31],[32]. However, the emergence of Cloud computing introduce a lot of benefits, as well as open doors for countless opportunities and models of computing and business[33]. Cloud computing has become an enabler of various platforms capable of: relatively higher degrees of flexibility; faster and much larger scale of computation, processing and sharing; wider accessibility and greater availability[34]. Other benefits of Cloud computing include: cost flexibility and efficiency; scalable resources for storage, backup and recovery; relatively easier setting up of customized environments and quicker deployments; and a myriad of service provisioning options[35]. Since the emergence of Cloud computing, more efforts are directed towards exploiting and leveraging cloud computing for the range of benefits and advantages it offers, mostly as services; and this is now evident in a range of services springing up e.g. Big Data-as-a-Service, analytics-as-a-service, and a host of other service offerings in the industry[36].

Cloud computing is a technology trend that is changing the IT landscape and changing collaboration[3]. One of its most notable advantage lies in its adaptability to varying contexts of use, its extensibility, as well as, the numerous possibilities and opportunities it presents for all stakeholders to collaborate [37]. However, not unlike most emerging paradigms, mixed feelings trail adoption of the Cloud[4], [5], [38]. For collaborative software development, the benefits include, but are not limited to, cost savings, scalability, agility for business and development peak period needs, motivation for innovation and increased R&D [29]. On the other hand, there are fears about: security issues; vendor lock-in and interoperability issues, portability issues; automation, performance issues; availability issues; handling uncertainty about: heterogeneity and content type, location of client, bandwidth unpredictability, dynamic workload variations, workflow schedules, architecture and resource optimization; availability and integrity of relevant information within participating teams and systems; context awareness and reproducibility within contexts; amongst others [27], [37], [39]. Some of these challenges and issues listed here, are partly inherited due to the fact that Cloud Computing itself, is a paradigm that leverages a couple of other technologies [40].

## IV. Overview of Collaborative Software Development Process

The importance of software in business and in daily activities cannot be overemphasized. This is partly due to the important role software has come to occupy in daily scenarios, resulting in a lot of attention and attempts been directed towards standardizing and improving the software development process[41]. Further fueling these attempts at improving the process is, the increase in size, complexity, and distribution involved in software development projects[21], [42]. This far exceeds what any one individual, or component can handle, and hence necessitates some sort of standardized collaboration approach between diverse set of people, skills, activities, processes, locations, tools and environments, configurations and specifications. Software developmentis a collaborative activity, involving divergent and convergent activities carried out by people or teams, in an environment, towards achieving a set of objectives or outcome[43][44].

The Software development process refers to the entire process of developing software, encompassing: a team, framework of activities, set of practices providing guidelines for designing, developing, testing, deploying, maintaining and managing software[45]. This includes the interactions too. The entire process involves different parts working together towards a goal. This process spans the entire development lifecycle and is usually embodied in a defined high-level abstraction usually referred to as a software development model[46]. Software development models facilitate and guide a set of tasks or activities to transform problem definitions and requirements into software[23], [45], [46]. Various types of software development models, adapted as different methodologies, are essentially efforts aimed at standardizing and improving the process of developing software[1]. The table presents a cross-section summary of some common software development models and related characteristics, advantages and challenges. This is provided to serve as a baseline review template for relative comparisons, considerations, and reconciliations or possible combinations[23]. Some inherent similarities amongst the models include: reliance on collaborative development process and the team; accountability of the team and process along the lines of responsibility, roles, and functions; iteration within the process geared towards management of change, risks and performance; design, development and testing activities geared towards achieving a common overall outcome[47].

Despite these differentiating aspects, themodels possess some inherent similarities. These include: reliance on the collaborative development process and the team; accountability of the team and process along the lines of responsibility, roles, and functions; iteration within the process geared towards management of change, risks and performance, etcetera. Nonetheless, when properly implemented within the various stages of the software development process, any of the models has the capacity to deliver quality solutions. The stages of the software development process are not always set in stone, neither are the boundaries of the stages always clearly delineated or differentiated[21], [23], [47]–[50].

The stages in the development process are usually carried out via different activities, grouped into development models, with some overlapping, and mixing towards achieving the outcome[47]. A typical software development project usually comprises a team made up of people of diverse cultures, skillset, technical expertise, and technological/non-technological viewpoints, either, working together on different tasks, or separately on complementary tasks at each stage of the process towards a common goal, all the while ensuring communication via a variety of tools or medium[21]. This calls for efficient collaboration and management in the software development process [44]. The important role of software in the society, and other factors such as: increase in size, complexity, and distribution involved in software development projects have generated a lot of attention, leading to attempts been directed towards standardizing and improving the development process[21], [41].

At the beginning of any software development project, determining the scope of collaboration is one of the most challenging aspects of the project. Understanding and defining what presents as core aspects and what is not, is necessary. This definition could sometimes come from stated business values and requirements; as well as from architectures, ontologies, taxonomies, dictionaries, standards[51]. To date, the focus of majority of R&D efforts in the area of Cloud-based software development is at best imbalanced. Most concentrate on specific aspects of the development process, resulting in insufficient attention being paid to other aspects equally undermining collaboration. A review of related literature reveals that efforts devoted towards Cloud-based Collaborative Software Development have been mainly in the areas of: trust and privacy; asynchronous collaboration; isolated collaboration in specific aspects of the process, such as coding activities; use of open-source tools for contributing, improving, and managing code, and some have leveraged social networking as an enabler too [9], [52]–[55]. Although these efforts represent valid contributions and important enablers, they are still missing important aspects that enable a more holistic process, with solid theoretical foundation in the Cloud[52], [56].

The concept of leveraging the Cloud to create or enhance collaboration in different activities is shaping up and gaining solid ground in a lot of areas and field. The table in this section plot the landscape of current updated or reported knowledge on cases of successful leveraging of Cloud Computing capabilities for collaborative software development from a cross section of industry. However, it is important to state here, that this table is not an exhaustive list. This is attributed to reasons such as: work may be unrecorded or unpublished; work may be closely guarded or undocumented industrial intellectual property (IP); or work may be experimental projects yet to be verified or validated [54], [57], [58]. Furthermore, highlighting and reviewing these works would also help in highlighting gaps and emphasizing the need for more research efforts. This table shows that leveraging the Cloud for collaborative software development is a viable area that is gaining traction and being explored by industry, as well as, academia. This is in a bid to: address the inefficiencies and inconsistencies of the traditional process and environment for software development; align software development with current trends and changing business requirements; leverage new concepts and methods for optimal development process, economies of scale and efficient use of resources, tighter collaboration, efficient management from automation and context-aware linking and sharing of information [1], [21]. From the table, it can be seen that most current solutions offered in Industry as 'Cloud-based solutions', offer more support for the coding and deployment stages of the software process, and less for other stages such as the requirements gathering stage, the testing stage and the design stage.Some of the solutions attempt to integrate social communication by featuring some social communication tools [59]–[62]. In the same way that merely developing applications compatible with the Cloud does not make the application Cloud-agnostic, merely integrating social communication tools or features with a Cloud-based IDE does not necessarily make the development environment a collaborative Cloud-based development platform. Arguably, integrating social networks in the enterprise with Cloud

development environments would be an approach towards enabling or enhancing collaboration in Cloud development environments, but leveraging the Cloud for a fully collaborative development environment in the Cloud is more than that[63]. Table 1below, presents a survey of a cross-section of notable open source tools in industry, representing efforts towards Collaborative Software development process in the Cloud. These have been categorized according to various emerging themes of differentiation from surveyed literature.

**Table 1:A Survey of cross-section of notable open source industry tools/platforms towards Cloud-based SDLC process**

| Differentiation themes | GitHub | CloudTeams | Sonarqube | Atlassian Confluence/Jira | IBM jazz/CLM | CollabNet/ TeamForge | Heroku |
|---|---|---|---|---|---|---|---|
| Cloud-based/Cloud-hosted/Non-Cloud | Cloud-hosted | Cloud-based | Cloud-hosted | Partially Cloud-based | Cloud-hosted | Cloud-based | Cloud-based |
| Explicit Collaboration activity-themed, Theoretical Basis for architecture for Cloud-based collaborative software development process | None/Indeterminate | None/Indeterminate | None/Indeterminate | None/Indeterminate | None/Indeterminate | None/Indeterminate | None/Indeterminate |
| Implicitly associated theories | Social Network Graph | None / Indeterminate | Cognitive Complexity | None / Indeterminate | None / Indeterminate | None / Indeterminate | None / Indeterminate |
| Cloud-agnostic, contextualised artefact format for artefacts from all stages | partial | partial | partial | partial | partial | None/Indeterminate | partial |
| Collaboration in all SDLC stages | partial | partial | partial | Yes | Yes | Yes | partial |
| Formal testing across all stages (Validation & verification) | partial | partial | partial | partial | Yes | partial | partial |
| Metrics/benchmarks | Yes | partial | partial | Yes | Yes | Yes | Yes |
| Metrics/benchmarks for analyzing/measuring collaboration within the lifecycle process | partial | None / Indeterminate | None / Indeterminate | None / Indeterminate | partial | None / Indeterminate | None / Indeterminate |
| Traceability | Yes | None / Indeterminate | partial | Yes | Yes | Yes | partial |
| Awareness | partial | partial | partial | partial | partial | partial | partial |
| Co-ordination | Yes | | partial | Yes | Yes | Yes | Yes |
| Communication | Yes | | partial | Yes | Yes | Yes | Yes |
| Shared Workspace | Yes | | Yes | Yes | Yes | Yes | Yes |
| Shared memory | Yes | | Yes | Yes | Yes | Yes | Yes |
| Context-awareness | partial | | partial | partial | Yes | partial | partial |
| Main features | Code repository, Developer profiles, dedicated project pages, code-related actions (Commits, forks, pull requests), subscription actions, version control, documentation | Customizable platform, allowsmashable endpoint connection of development tools, interface to allow anonymous end-user engagement with development teams in earlystages | Java-code analysis engine, metrics & issue detectors, GUI Dashboard with drill-down features, Plug-in extension capabilities | Cloud Platform, UI Modules, Webhooks, Rest API, device drivers, plugin managers, network abstractions and generic services | Integrated set of tools developed on IBM Jazz platform, web-based interface, extension capabilities; | Integrated toolchain combining open source tools for end-to-end application development & testing. Include: Eclipse, Git, Subversion, Jenkins, Visual Studio, Atlassian Jira, JFrog | Managed Containers, Heroku Pipelines, built-in monitoring tools, extension capabilities, GitHub integration, single point dashboard for managing teams& processes, API |

As can be seen from Table 1, most of the surveyed notable open source development tools are cloud-based, and are more collaborative in some stages of the development process than others. A good proportion of the surveyed tools are collaborative in all the stages, with little or no defined metrics for specifically benchmarking collaboration in the process. The main areas of focus for most of the tools include: continuous code quality management via inspection, analysis and reporting on issues, bugs or errors in code; providing interface to mashable collection of popular development tools. For example, in the case of GitHub, collaboration in the process exists in the form of team members working together via pull requests and commit actions. It is sometimes difficult to figure out which projects are live and which are abandoned. Only way of doing so is through history of commit actions, because not all pull requests are guaranteed to be accepted and merged. Another way of considering collaboration in development processes using Github, is by considering projects in light of partial contexts such as: actions on code; who executed the actions; manual linking of related commits, comments and issues, in order to make inferences and reasoning. This platform used to be collaborative only in some stages of the SDLC, and makes provisions for using various methodologies. However, recent updates extended this collaboration across all stages. The end to end traceability offered by artefacts is a good feature, but there is still need to have a full Cloud-agnostic, contextualized artefact format for artefacts from all stages to allow for easy automations and implementation of automations, as well as synchronized understanding.GitHub focuses mainly on developers and. Collaboration exists but mostly centered on the development, testing and deployment of applications Collaboration is mostly asynchronous. This is applicable to most of the surveyed tools. Collaboration is not a focal point, neither does it extend to other stages in the life cycle development process not involving code. Less focus is placed on the activity. Table 1 above clearly shows the absence of explicit activity-themed, theoretical basis for the cloud-based software development process. Some of the tools seek to promote collaboration between end-users and teams via participation and incentives. They do not address underlying issues undermining collaboration such as: complexity or unified formats for output to ensure synchronized understanding. Addressing the latter could lead to developing a formal empirical way of validating that end product meets user requirements, or the proposal of metrics for benchmarking the collaboration in the Cloud-based development process.

## V. Challenges and Future Directions for Cloud based Collaborative Software Development Life Cycle

*Need for Cloud-based Collaborative software development architectures with explicit theoretical foundation.*

Emerging technologies and software engineering trends change the way software is accessed, utilized, stored and maintained. These introduce consideration points such as: more distribution, greater complexity and increase in contexts. The result of this is a constant need to develop safe, secure and reliable software that will continuously evolve and adapt to changing requirements, and a constantly evolving development process. Current innovative solutions rely on results from a mix of successful and failed implementations and glitches [9], [50]–[53], [62]–[68].

**Impact**

i. randomness in the science of the development process
ii. Undermined collaboration in the software development lifecycle process
iii. Increase in emphasis on need for better and sustainable frameworks, architectures, methods, tools, practices and strategies, with explicit theoretical foundations to embrace and adapt to changing trends in technology, process, requirements, and related complexity, whilst still facilitating effective collaboration across the entire development process
iv. need for sustainable change management and self-learning methods in Cloud-based collaborative software development lifecycle

**Proposed recommendation**

Provision of explicit theoretical framework with activity underpinnings to:

i. facilitate sustainable and reproducible blueprint for Cloud-based context-aware, collaborative software development lifecycle process
ii. aid understanding and conceptualization of ways to enhance collaboration in Cloud-based software development lifecycle
iii. lay a foundation for defining processes, activities, and aligning them with goals and deliverables,
iv. synthesize empirical knowledge to facilitate future research, development and adaptation of collaborative models for development and testing of Cloud applications
v. Flag up irregularities, inconsistencies, and other factors which might impact an activity.
vi. reduce or eliminate randomization and reliance on results from failed implementations and glitches

*Need for effective methods for capturing and representing contexts and other related data in a Cloud-agnostic formatfor generation of actionable insights*

Requirements, artefacts, action plans, feedback, and other important related information, necessary to achieve the defined goalare sometimes not clearly and accurately defined within the Cloud-based development process. One factor contributing to this is the poor collection, unsynchronized understanding, ignorance and poor application, of contexts and other related metadata [44], [55]–[59], [62]-[64], [67]–[79].

**Impact**

i. negative impact on balancing and optimization of information flow within development environments and teams;
ii. late detection and resolution of issues and bugs that could have been otherwise avoided via appropriate

collection, consideration and application of sufficient context data within development activities;
iii. inadequate tracking of project progress;
iv. conflicting perspectives, understanding, interpretation and execution of activities, often resulting in defective software, or software needing more rework

**Proposed recommendation**

Design and implementation of a common representational format for: context information, requirements, outputs from each stage of the lifecycle development process, logs, feedback, ideas, instructions, concerns, and other related data. Also recommended is the design and implementation of knowledge management mechanisms and modules for data processing, analytics, and visualization and reporting functions. This would require scalable data storage. Benefits include:

i. Effective traceability, change management, better visibility and synchronised understanding and awareness
ii. Generation of actionable insights from: logs, feedback from tasks, activities, interactions, executions and transformations. This would aid and facilitate:self-learning from historical data, process improvement in management, technical, and coordinating aspects
iii. Building up of domain knowledge for the process, troubleshooting purposes, creation of libraries and templates, as well as improving the adaptability of the process
iv. Automation of information flow frees up valuable resources; reduces unnecessary noise (assumptions and discussions), and makes it easier to monitor and manage - conversations, alerts, notifications, changing parameters, exchanges, design progress, status, changing mission parameters, directives and instructions.

*Need for effective ways of managing complexity across Cloud-based collaborative software development lifecycle*

Certain disciplines such as the engineering disciplines, are usually guided, constrained and regulated by physical laws that ensure regularity and a way of keeping complexity in check. Conversely, Software Engineering is not easily regulated or bound by physical laws. This makes it harder to ensure synchronous collaboration and verifiable outputs at the various stages of the process[1], [21], [69], [80]–[82]

**Impact**

i. Growth in complexity of software artefacts and across the Cloud-based development lifecycle process
ii. Differences and difficulty in understanding, developing and testing in the right way, and correctly.
iii. Increased need to challenge and validate results via some form of empirical effort

**Proposed recommendation**

One way to approach and reduce impact of this gap would be to limit complexity via the development of an architecture. An architecture would contribute towards managing complexity through decomposition and abstraction of main components of the Cloud-based collaborative development process. Furthermore, the provision of an activity-themed or collaboration-themed theoretical foundation for the architecture would help to boost confidence in the architecture, and its sustainability. Like in the case of the engineering disciplines, this theoretical foundation can be derived from existing laws, theories, and concepts, that be applied to guide different aspects of both the architecture and the process. Benefitsinclude:

i. Reduction of constraints impacting the ability to understand, design develop, test and maintain software artefacts. This helps to manage complexity and impact.
ii. Promotion of integrity of the process and outcomes
iii. Facilitation of reusability and impact analysis

*Need for standards and adequate metrics for benchmarking Cloud-based collaborative development and testing*

The existing standards commonly used in software development processes are quite generic. They are mostly used for assessing and analyzing how organizations follow their defined processes, as well as, modelling processes to monitor and control the development of software. These standards do not expressly cater for the analysis, assessment and measurement of the collaborative process within the cloud. Presently, the commonly used standards include: ISO 9000, CMMI, ISO 15504[67], [78], [83]–[87].

**Proposed recommendation**

Introduction of suitable methods for benchmarking Cloud-based collaborative software development lifecycle process to ensure monitoring and management of the process, and continuous process improvement

## VI. Conclusion

The collaborative software development process comprises of divergent and convergent activities carried out by people or teams, in an environment, towards achieving a set of objectives or outcome. Analyzing and differentiating various collaborative approaches, contributions and tools in both industry and academia, helps in vertically organizing and aligning all existing fragmented approaches within context. Cloud-based collaborative software development process needs to incorporate holistic collaborative concepts and technologies; theoretical foundations, as well, as integrate a management layer to effectively manage the collaboration and resources within the project in line with identified constraints, and stated or identified business requirements and needs.

## VII. References

[1] G. A. Dafoulas, K. Swigger, R. Brazile, F. N. Alpaslan, V. L. Cabrera, and F. C. Serce, 'Global Teams: Futuristic Models of Collaborative Work for Today's Software Development Industry', in *2009 42nd Hawaii*

*International Conference on System Sciences*, 2009, pp. 1–10.

[2] F. Lanubile, 'Collaboration in distributed software development', *Softw. Eng.*, pp. 174–193, 2009.

[3] J. Noll, S. Beecham, and I. Richardson, 'Global software development and collaboration: barriers and solutions', *ACM Inroads*, vol. 1, no. 3, pp. 66–78, 2010.

[4] A. M. Magdaleno, C. M. L. Werner, and R. M. de Araujo, 'Reconciling software development models: A quasi-systematic review', *J. Syst. Softw.*, vol. 85, no. 2, pp. 351–369, Feb. 2012.

[5] I. Mistrík, J. Grundy, A. Hoek, and J. Whitehead, *Collaborative Software Engineering*. Springer Science & Business Media, 2010.

[6] A. Begel, J. D. Herbsleb, and M.-A. Storey, 'The future of collaborative software development', in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion*, New York, NY, USA, 2012, pp. 17–18.

[7] M. Derntl, D. Renzel, P. Nicolaescu, I. Koren, and R. Klamma, 'Distributed Software Engineering in Collaborative Research Projects', in *2015 IEEE 10th International Conference on Global Software Engineering*, 2015, pp. 105–109.

[8] Z. Mahmood and S. Saeed, *Software Engineering Frameworks for the Cloud Computing Paradigm*. Springer Publishing Company, Incorporated, 2013.

[9] Box, 'The Cloud: Reinventing Enterprise Collaboration', *FierceCIO*, May 2012.

[10] B. Jackson, 'Cloud Collaboration', *Mix*, vol. 35, no. 5, pp. 16–18, May 2011.

[11] B.-Y. Chang, P. H. Hai, D.-W. Seo, J.-H. Lee, and S. H. Yoon, 'The determinant of adoption in cloud computing in Vietnam', in *2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, 2013, pp. 407–409.

[12] K. Ghaffari, M. S. Delgosha, and N. Abdolvand, 'Towards Cloud Computing: A SWOT Analysis on its Adoption in SMEs', *Int. J. Inf. Technol. Converg. Serv.*, vol. 4, no. 2, pp. 13–20, Apr. 2014.

[13] O. M. Yigitbasioglu, 'Cloud Computing Adoption in Australia: Evidence from the Forensic Accounting Industry - DO NOT CITE', 2014.

[14] S. Zardari and R. Bahsoon, 'Cloud Adoption: A Goal-oriented Requirements Engineering Approach', in *Proceedings of the 2Nd International Workshop on Software Engineering for Cloud Computing*, New York, NY, USA, 2011, pp. 29–35.

[15] L. Riungu-Kalliosaari, O. Taipale, and K. Smolander, 'Testing in the Cloud: Exploring the Practice', *IEEE Softw.*, vol. 29, no. 2, pp. 46–51, 2012.

[16] R. Oberhauser, 'Towards Cloud-based Collaborative Software Development: A Developer-Centric Concept for Managing Privacy, Security, and Trust', in *ICSEA 2013, The Eighth International Conference on Software Engineering Advances*, 2013, pp. 533–538.

[17] J. Raubenheimer, *Mendeley: Crowd-sourced Reference and Citation Management in the Infomation Era*. True Insight Publishing, 2014.

[18] B. Cronin and C. R. Sugimoto, *Beyond Bibliometrics: Harnessing Multidimensional Indicators of Scholarly Impact*. MIT Press, 2014.

[19] Oxford Dictionaries, 'collaboration: definition of collaboration in Oxford dictionary (American English)', Jul-2013. [Online]. Available: http://oxforddictionaries.com/definition/american_english/collaboration. [Accessed: 04-Jul-2013].

[20] A. M. Thomson and J. L. Perry, 'Collaboration Processes: Inside the Black Box', *Public Adm. Rev.*, vol. 66, pp. 20–32, 2006.

[21] A. M. Thomson, J. L. Perry, and T. K. Miller, 'Conceptualizing and Measuring Collaboration', *J. Public Adm. Res. Theory*, vol. 19, no. 1, pp. 23–56, Jan. 2009.

[22] E. A. Henneman, J. L. Lee, and J. I. Cohen, 'Collaboration: a concept analysis', *J. Adv. Nurs.*, vol. 21, no. 1, pp. 103–109, 1995.

[23] F. Lanubile, C. Ebert, R. Prikladnicki, and A. Vizcaíno, 'Collaboration tools for global software engineering', *IEEE Softw.*, vol. 27, no. 2, 2010.

[24] J. Whitehead, 'Collaboration in Software Engineering: A Roadmap', in *Future of Software Engineering, 2007. FOSE '07*, 2007, pp. 214–225.

[25] L. M. Camarihna-Matos and H. Afsarmanesh, 'Concept of Collaboration', *Academia.edu*, 2008. [Online]. Available: http://www.academia.edu/248756/Concept_of_Collaboration. [Accessed: 24-Jun-2013].

[26] L. Badger, T. Grance, R. Patt-Corner, and J. Voas, 'Draft cloud computing synopsis and recommendations', *NIST Spec. Publ.*, vol. 800, p. 146, 2011.

[27] M. Armbrust *et al.*, 'A view of cloud computing', *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[28] M. Armbrust *et al.*, 'Above the clouds: A berkeley view of cloud computing', 2009.

[29] E. M. Maximilien and P. Campos, 'Facts, trends and challenges in modern software development', *Int. J. Agile Extreme Softw. Dev.*, vol. 1, no. 1, pp. 1–5, Jan. 2012.

[30] S. K. Doddavula, I. Agrawal, and V. Saxena, 'Cloud Computing Solution Patterns: Infrastructural Solutions', in *Cloud Computing*, Z. Mahmood, Ed. Springer London, 2013, pp. 197–219.

[31] Quest, 'Challenges-Benefits-Cloud-Computing.pdf', pp. 1–10, 2012.

[32] S. Logo, 'Introduction to Cloud Computing'.

[33] F. Durao, J. F. S. Carvalho, A. Fonseka, and V. C. Garcia, 'A systematic review on cloud computing', *J. Supercomput.*, vol. 68, no. 3, pp. 1321–1346, Jun. 2014.

[34] B. Warth, N. Levin, D. Rinehart, J. Teijaro, H. P. Benton, and G. Siuzdak, 'Metabolizing Data in the Cloud', *Trends Biotechnol.*, 2017.

[35] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, 'A survey on vehicular cloud computing', *J. Netw. Comput. Appl.*, vol. 40, pp. 325–344, Apr. 2014.
[36] G. Skourletopoulos *et al.*, 'Big Data and Cloud Computing: A Survey of the State-of-the-Art and Research Challenges', in *Advances in Mobile Cloud Computing and Big Data in the 5G Era*, C. X. Mavromoustakis, G. Mastorakis, and C. Dobre, Eds. Springer International Publishing, 2017, pp. 23–41.
[37] D. Puthal, B. P. S. Sahoo, S. Mishra, and S. Swain, 'Cloud Computing Features, Issues, and Challenges: A Big Picture', 2015, pp. 116–123.
[38] N. Leavitt, 'Is Cloud Computing Really Ready for Prime Time?', *Computer*, vol. 42, no. 1, pp. 15–20, 2009.
[39] Q. Zhang, L. Cheng, and R. Boutaba, 'Cloud computing: state-of-the-art and research challenges', *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010.
[40] S. M. Hashemi and A. K. Bardsiri, 'Cloud Computing Vs. Grid Computing', 2009.
[41] A. MacCormack, C. F. Kemerer, M. Cusumano, and B. Crandall, 'Trade-offs between productivity and quality in selecting software development practices', *IEEE Softw.*, vol. 20, no. 5, pp. 78–85, 2003.
[42] E. Kalliamvakou, D. Damian, K. Blincoe, L. Singer, and D. M. German, 'Open source-style collaborative development practices in commercial projects using github', in *Proceedings of the 37th International Conference on Software Engineering-Volume 1*, 2015, pp. 574–585.
[43] R. Keil-Slawik, 'Artifacts in Software Design', in *Software Development and Reality Construction*, C. Floyd, H. Züllighoven, R. Budde, and R. Keil-Slawik, Eds. Springer Berlin Heidelberg, 1992, pp. 168–188.
[44] T. Zimmermann and C. Bird, 'Collaborative Software Development in Ten Years: Diversity, Tools, and Remix Culture', in *Proceedings of the Workshop on The Future of Collaborative Software Development*, 2012.
[45] A. Fuggetta, 'Software process: a roadmap', in *Proceedings of the Conference on The Future of Software Engineering*, New York, NY, USA, 2000, pp. 25–34.
[46] I. Sommerville, *Software Engineering*, 9 edition. Boston: Addison Wesley, 2010.
[47] M. Lepmets and M. Nael, 'Comparison of Plan-driven and Agile Project Management Approaches: Theoretical Bases for a Case Study in Estonian Software Industry', in *Proceedings of the 2011 Conference on Databases and Information Systems VI: Selected Papers from the Ninth International Baltic Conference, DB&IS 2010*, Amsterdam, The Netherlands, The Netherlands, 2011, pp. 296–308.
[48] T. Dybå and T. Dingsøyr, 'Empirical studies of agile software development: A systematic review', *Inf. Softw. Technol.*, vol. 50, no. 9–10, pp. 833–859, Aug. 2008.
[49] N. M. A. Munassar and A. Govardhan, 'A Comparison Between Five Models Of Software Engineering', *IJCSI Int. J. Comput. Sci. Issues*, vol. 7, no. 5, pp. 94–101, 2010.
[50] J. Feller, B. Fitzgerald, and others, *Understanding open source software development*. Addison-Wesley London, 2002.
[51] I. Skerrett, 'Collaborative Software Development in the Enterprise', *Open Source Bus. Resour.*, no. January 2009, 2009.
[52] R. Oberhauser, 'Cloud-based Collaborative Software Development: A Concept for Managing Transparency and Privacy based on Datasteads', *Int. J. Adv. Softw.*, vol. 7, no. 3 and 4, pp. 435–445, Dec. 2014.
[53] R. Al Mushcab and P. Gladyshev, 'The significance of different backup applications in retrieving social networking forensic artifacts from Android-based mobile devices', in *2015 2nd International Conference on Information Security and Cyber Forensics, InfoSec 2015*, 2016, pp. 66–71.
[54] J. Cito, P. Leitner, T. Fritz, and H. C. Gall, 'The making of cloud applications: An empirical study on software development for the cloud', in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 393–403.
[55] J. Cito, P. Leitner, H. C. Gall, A. Dadashi, A. Keller, and A. Roth, 'Runtime Metric Meets Developer: Building Better Cloud Applications Using Feedback', in *2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)*, New York, NY, USA, 2015, pp. 14–27.
[56] M. Nordio, H.-C. Estler, C. A. Furia, and B. Meyer, 'Collaborative Software Development on the Web', *ArXiv11050768 Cs*, May 2011.
[57] Linux Foundation, 'Collaborative Development Trends Report, 2014', Mar. 2014.
[58] W. Jun and F. Meng, 'Software Testing Based on Cloud Computing', in *2011 International Conference on Internet Computing Information Services (ICICIS)*, 2011, pp. 176–178.
[59] S. Ardaiz, 'Collaborative Communication: Why Methods Matter', *Triple Pundit People Planet Profit*, Dec. 2011.
[60] C. Gadea, B. Solomon, B. Ionescu, and D. Ionescu, 'A Collaborative Cloud-Based Multimedia Sharing Platform for Social Networking Environments', in *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, 2011, pp. 1–6.
[61] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, 'Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository', in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, New York, NY, USA, 2012, pp. 1277–1286.
[62] A. Begel, J. Bosch, and M.-A. Storey, 'Social Networking Meets Software Development: Perspectives

from GitHub, MSDN, Stack Exchange, and TopCoder', *IEEE Softw.*, vol. 30, no. 1, pp. 52–66, Jan. 2013.
[63] A. Bento and A. K. Aggarwal, Eds., *Cloud Computing Service and Deployment Models: Layers and Management*. IGI Global, 2012.
[64] R. Jeffery, 'Theory, models and methods in software engineering research.', in *ICSE'2000 Workshop on" Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research"(2000)*, 2000, pp. 2–7.
[65] P. Ralph, 'Possible Core Theories for Software Engineering'.
[66] P. Ralph, 'Software Engineering Process Theory: A Multi-Method Comparison of Sensemaking-CoevoIution-Implementation Theory and Function-Behavior-Structure Theory', *ArXiv13071019 Cs*, Jul. 2013.
[67] I. Gorton, A. B. Bener, and A. Mockus, 'Software Engineering for Big Data Systems', *IEEE Softw.*, vol. 33, no. 2, pp. 32–35, Mar. 2016.
[68] M. DEVLIN and S. DRUMMOND, 'Software Engineering Students' Cross-Site Collaboration: An Experience Report.', 2007.
[69] M. Mohtashami, V. Kirova, T. Marlowe, and F. Deek, 'A Comparison of Three Modes of Collaboration for Software Development', *AMCIS 2009 Proc.*, Jan. 2009.
[70] N. Chanda and X. F. Liu, 'Intelligent analysis of software architecture rationale for collaborative software design', in *2015 International Conference on Collaboration Technologies and Systems (CTS)*, 2015, pp. 287–294.
[71] T. Marlowe, 'Addressing Change in Collaborative Software Development: Process and Product Agility and Automated Traceability'.
[72] N. Jastroch, 'Advancing Adaptivity in Enterprise Collaboration', Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 1907348, Nov. 2009.
[73] G. Mark, 'Extreme Collaboration', *Commun ACM*, vol. 45, no. 6, pp. 89–93, Jun. 2002.
[74] T. Hildenbrand, F. Rothlauf, M. Geisser, A. Heinzl, and T. Kude, 'Approaches to Collaborative Software Development', in *International Conference on Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008*, 2008, pp. 523–528.
[75] A. Kyriakidou-Zacharoudiou, 'Distributed development of large-scale distributed systems: the case of the particle physics grid', phd, The London School of Economics and Political Science (LSE), 2011.
[76] J. Münch and K. Schmid, *Perspectives on the Future of Software Engineering: Essays in Honor of Dieter Rombach*. Springer Science & Business Media, 2013.
[77] V. Pankratius, *Emerging Research Directions in Computer Science: Contributions from the Young Informatics Faculty in Karlsruhe*. KIT Scientific Publishing, 2010.
[78] M. Richards, *Software architecture patterns*, 1st ed. O'Reilly Media, Inc., 20015.
[79] B. Boehm, 'Some future trends and implications for systems and software engineering processes', *Syst. Eng.*, vol. 9, no. 1, pp. 1–19, Mar. 2006.
[80] B. W. Boehm, 'Some Future Software Engineering Opportunities and Challenges', in *ResearchGate*, 2010, pp. 1–32.
[81] B. Boehm, 'A View of 20th and 21st Century Software Engineering', in *Proceedings of the 28th International Conference on Software Engineering*, New York, NY, USA, 2006, pp. 12–29.
[82] M. B. Chrissis, M. Konrad, and S. Shrum, *CMMI for Development: Guidelines for Process Integration and Product Improvement*, 3 edition. Upper Saddle River, NJ: Addison Wesley, 2011.
[83] M. Mohtashami, T. J. Marlowe, and C. S. Ku, 'Metrics Are Needed for Collaborative Software Development', *J. Syst. Cybern. Inform.*, vol. 9, no. 5, pp. 41–47, 2011.
[84] E. M. Bouwers, 'Metric-based Evaluation of Implemented Software Architectures', 2013.