Classifying collaborative approaches for Cloud Based Collaborative Software Development

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*— Software development is an activity involving a remix set of different people, tools, practice culture, etcetera, and working towards an end goal. Achieving the goal necessitates that all these aspects work together towards the end goal. Furthermore, the size, complexity, longevity and tight delivery timelines of software projects, form part of the rationale for collaboration in software development processes. With the advent of Cloud computing, these factors have become more pronounced. Other factors such as increased distribution, have also become part of the rationale increasing the need for better collaborative approaches. Collaboration can take numerous forms and dimensions, but that does not necessarily mean that any form of collaborative approach is good for every scenario or context. There is no "one size fits all" approach. Different development contexts may require different collaborative approaches for greater effectiveness. So, which collaborative approach is right, and which is wrong, for Cloud-based software development lifecycle? This paper reviews literature with an aim of presenting a classification for collaborative approaches towards context-aware Cloud-based software development.**

Keywords—cloud; collaborative software development; collaborative approach; classification

I. INTRODUCTION

The broad nature of collaboration in activities, in terms of coverage, composition and process, makes classification a way of simplification. This could be via identification of salient or implicit features within a given scenario or context. A good classification would make it easier to identify and assemble necessary building blocks for an adequate collaborative approach for a given scenario, context, or even activity. Software development process is a collective activity by nature, requiring joint efforts to work towards achieving a common goal[1], [2]. This implies a need for collaboration. This is further buttressed by the decentralization and concurrency introduced by emerging technologies and paradigms such as cloud computing. Although these, bring about efficiency in certain aspects of the process, they introduce complexities and complications in other areas of the Software development process. Some of these areas include co-ordination, communication and awareness, amongst others [3].

The focus in this research is on collaborative approaches to Cloud-based software development. Software development departments and organizations in the cloud are getting more decentralized with increasing distribution in location, processes, cultural practice and team makeup [4], [5]. Accessing and housing software applications in the cloud, implies a need for a change in how software is developed [6] These changes give rise to novel challenges, as well as, increase demand for different kinds of software, highlighting the increasing importance of adopting collaborative approaches and tools [1].This warrants a review and analysis of existing collaborative approaches. Identification and classification of existing collaborative approaches to software development, is of great importance [3]. This would contribute towards better understanding, analysis, characterization and evaluation of these approaches and any gaps therein. Sequel to this is, investigation of better ways, paradigms and technologies to leverage, towards addressing challenges in collaborative software development in the Cloud. This would facilitate better insight and understanding when seeking to develop frameworks and architectures to enhance the collaborative development process[7].

Prior to classifying collaborative approaches for Cloud-based Software development, it is necessary to review literature to identify trends and gaps which emphasize this need. This helps with alignment of classification with this research domain. The primary aim of this research is: a review of existing collaborative approaches, in alignment with gaps identified from a synthesis of related literature, for the proposal of an adequate classification of collaborative approaches for Cloud-based collaborative software development. This would facilitate design and development of better architectures, tools, techniques, methodologies and processes, for supporting collaborative software development in the Cloud.

This paper is structured around literature findings and evidence. The structure of this paper is laid out as follows. Section I introduces this research, the aim of the paper as well as, contextual background for the research. Section II provides related work. Section III explores collaborative approaches and proposes a classification of collaborative approaches for Cloud-based collaborative software development. Section IV concludes the paper with suggestions for future direction.

II. RELATED WORK

Software development is 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[8][9]. 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[10]. 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[11]. Software development models facilitate and guide a set of tasks or activities to transform problem definitions and requirements into software[10]–[12]. Various types of software development models, adapted as different methodologies, are essentially efforts aimed at standardizing and improving the process of developing software[7]. A typical software development project usually comprises a team made up of people of diverse cultures, skillset, technical expertise, technological and 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[4]. This in itself, calls for efficient collaboration and management in the software development process [9].

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[4]. Analyzing various collaborative approaches, methods and tools in the different phases of the software development process helps in vertically organizing all existing fragmented approaches, and aligning them with the value chain [3]. This contributes towards development of a framework to act as a process model for implementing a vertically collaborative and continuous software development process. At the time of this research, most collaborative approaches are horizontally collaborative, mostly asynchronous or partially synchronous, rarely context-aware and mostly not developed with distributed environments in mind [3], [5], [7], [13].

Collaborative Software development in the Cloud, introduces complexities and contexts, amidst other factors, that were hitherto, either non-existent or less pronounced [2]. These are sometimes underestimated, ignored, or sometimes not given enough consideration and planning. This undermines the collaboration in the process, randomizes the process, and impacts the ability to facilitate a reproducible, sustainable, context-aware collaborative software development process in the Cloud [10]. This has led to need for: identification of reliable ways of managing and measuring collaboration and other success factors within the process; new methodologies and ways of enhancing effective collaboration within the lifecycle development process; effective ways of managing complexity and ensuring synchronous regularity, as well as, verifiable outputs and outcomes at the various stages of the collaborative development lifecycle process [9], [14], [15]. Furthermore, development of key dimensions for analyzing and benchmarking the collaborative process is necessary step towards continuous process improvement and sustainability. This would translate into the ability to consistently reproduce the enhanced process, and ultimately, standardization in the form of frameworks, architectures and standards.

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 very little R&D efforts have been devoted towards enhancing collaboration in software development in general, as well as, in the Cloud [9], [14]–[20]. A few efforts have been devoted towards 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 [4], [5], [14], [21]. There has been very few efforts devoted towards developing classifications for collaborative approaches for software development, and none for Cloud-based software development [3], [4], [22]–[24]. Table II below summarizes some of the areas of R&D efforts from the last seven decades, as well as, effects from these efforts that have further contributed towards undermining real-time collaboration and reducing the overall effectiveness of the development process. Although these efforts represent valid contributions and important enablers, they are still missing important aspects that enable a more effective collaborative cloud-based development process[15], [25].

Table 1: Summary of a review of seven decades of R&D efforts and effects on the development process

Decade	Summary of trends and R&D efforts
1950s	Emphasis was adoption of other branches of science to aid in the evolution of software processes and methodologies. Introduced increasingly unpredictable and diverse range of fast-growing problems.
1960s	Application of low-risk, fast, out-of-the-box processes for approaching and solving software development problems. This method often introduced defects, needing patches or rework.
1970s	Use of structured and formal methods in a bottom-up approach towards goal-oriented and purposeful software projects, with priority given to early elimination of errors. Often required pre-determination of system's purpose and domain understanding. Resulted in issues with compliance, scalability, reusability, and process overheads.
1980s	Adoption of various approaches towards increasing productivity in software projects. Mainly directed towards aspects such as: staffing, architecture, compliance, component reuse, process improvement and maturity. Productivity increased, but skepticism flourished, whilst the rate and scale of innovation reduced.
1990s	More emphasis on reducing the time to market software, improving usability and usefulness of software, as well as maximizing returns on investment. Popular methods of this decade include agile methods, product reuse, concurrent processes and rapid composition methods. This sometimes resulted in overambitious and unrealistic milestones and deadlines; incomplete specifications, incompatibility, and lots of time spent on rework and integration
2000s	Use of adaptable methods e.g. model and plan-driven methods, hybrid agile methods, as well as service-oriented architectures. These were adapted towards addressing dynamic increases and changes in business and stakeholder needs. More attention was paid towards integrating systems and software engineering. Negatives include: issues with scalability, and clashes in models and methods used
2010s	Emphasis on creation and adoption of value-based methods, enterprise architectures, enabled by emerging paradigms. These were geared towards: ease of use for end users to build own

	systems, scalability, global connectivity, agility, and use of diverse processes, components, platforms, skills, and practice and on-demand resources as services. These have introduced: complexity, distribution, proliferation of incompatible tools and inadequate methods, time zones differences, development culture and practice clashes, problems with continuous synchronous integration, compatibility with legacy applications and traditional processes and practices.

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[7]. Reasons for this include: to address the inefficiencies and inconsistencies of traditional processes and environments for tighter synchronous collaboration, as well as, 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; efficient management, automation, context-aware linking and sharing of information [4], [7]. Literature review shows 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 [26]–[29]. 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. However, leveraging the Cloud for a fully collaborative development environment is more than that. Some authors [30] presented an interesting cross-section on existing Collaborative Software development tools and environments grouped by colors based on key characteristic features.

There is need for Cloud-based software development process to incorporate collaborative concepts and technologies, as well as, integrate a management layer to effectively manage the collaboration and resources, aligned with identified constraints, and identified business requirements and goals[56]. Establishing an adequate classification of collaborative approaches would: enhance effective interactions between different aspects of a cloud-based software development process; ensure that appropriate consideration is given to existing and changing contexts of collaboration; enhance and align collaborative activities to defined goals and outcomes; and lend defined structure to the process. This would greatly aid in employing and reinforcing principles and concepts embodied in collaborative approaches in the design and development of collaborative solutions i.e. models, methods, frameworks and architectures for cloud-based collaborative software development [3], [14].

III. CLASSIFYING COLLABORATIVE APPROACHES

The very nature of the software development process as a group activity requiring joint efforts geared towards a common goal implies a need for collaboration. A review and classification of collaborative approaches is necessary to foster better understanding, analysis, and evaluation of ways to align and streamline collaborative activities. Sequel to this would be the investigation of paradigms and technologies to leverage towards addressing challenges in the collaborative software development process.

To be able to identify an adequate collaborative approach for the Cloud-based software development process, it is necessary to identify the components of a typical development process, related aspects and contexts that would be present. In addition, it is also necessary to identify the activities and practices involved, as well as other points of consideration. Below are different ways of analyzing collaboration or collaborative approaches and activities within the software development process. Different schools of thought exist with regards to classification of collaborative work within the software development process [3], [25], [28], [32], [33].

A. Classification based on empirically measured activities within collaborative software development process

This is broken down into smaller categories for easier understanding of the interactions and how best to support or enhance the collaboration within. This classification focuses more on the main actors, rather than both the actors, the process, and other related contexts. The four classes are: Mandatory collaborative activities, Called collaborative activities, ad-hoc collaborative activities and individual collaborative activities [4], [23], [33]. Mandatory collaborative activities refer to formally scheduled activities, and can be either technical or non-technical. Called collaborative activities refer to activities initiated primarily to solve a problem, and are mostly technical in nature. Ad-hoc collaborative activities refer to those activities that require more than one team member or process, working on the same task simultaneously.

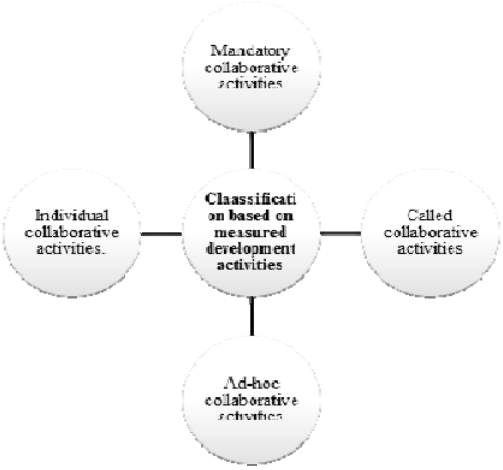

Figure 1: Classification based on empirically measured development activities

B. Classification derived from objectives of activities

The interactions between the components of the software development process provides another perspective for analyzing and classifying collaborative work, as well as

motivation for enhancing the process [34], [35]. This classification stems from the need for effective and efficient interactions between all the aspects and components of the process or activity, in order to ensure meeting the desired goal. As such, classification is done based on interactions according to objectives of the activity[36], [37]. This is depicted in the figure below, showing a generalized view of stages in a typical software development project. Within each stage or parent activity, smaller or sub-activities are carried out, to ensure that the objectives of the parent activity are met. If the need arises, these sub-activities are further broken down into sub-sub-activities, which are further broken down till it gets to the nth activity. This decomposition goes on and on, depending on need. Within each activity, interactions take place to achieve the desired transformation or objective. These interactions may be sequential or concurrent, subject to dependencies, and may be in any, or all of the following forms: human to human; human to non-human; non-human to non-human. These interactions may involve the sharing of artefacts such as code, design specifications, requirement documentation and use cases, test scripts, test specifications, etcetera. Suffice to say, the larger the project, the more the components, the more the number of interactions, and the more the artefacts. Hence the increase in complexity, that would need to be kept track of, and managed properly. Situations like this in any of the stages, say for example, the requirements' stage, could quickly lead to backlogs of inconsistent and ambiguous user stories or use cases. Arising results from this include: inadequate or very poor quality output, oversights, and late schedules[32].

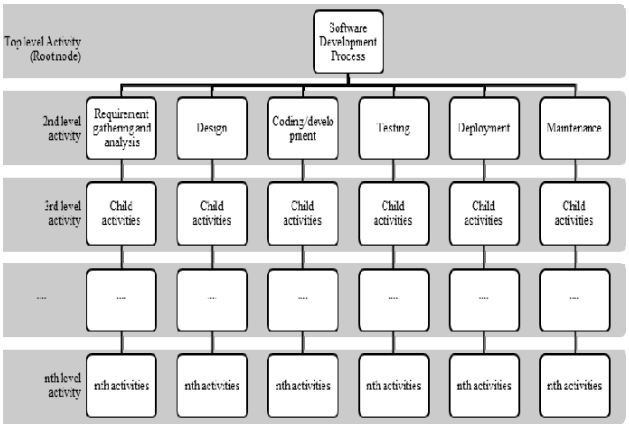

Figure 2: Classification based on objectives of development activities

C. *Classification based on Software development process characteristics*

This classification focuses on characteristics of the process, rather than context or activity levels [3], [24]. These characteristics are grouped into distribution-based characteristics and process-based characteristics. Distribution-based characteristics include organizational, spatial and temporal distributions. Organizational distribution refers to distribution of the development process based on: organizational units – these could be exhibiting inter-organizational or intra-organizational characteristics; or project-related characteristics – these could be inter-individual or inter-team; or business-related - these could be either company-wide or on the scale of the business environment. Spatial distribution makes the distinction between spatial distribution and spatial collocation which can occur either within or across organizations. These characteristics can include: tacit knowledge transfer considerations, personal contact considerations, and coordination considerations, differences in time zones, and development culture and practice; along with potential impact on collaboration contexts. Temporal distribution refers to making the distinction between synchronous and asynchronous characteristics of software development processes or activities. This includes processing of requirements, artefacts, or information. Process-based characteristics include process disciplines, process directions and process intensity. Process discipline refer to the phases, also referred to as the disciplines, of the software development process e.g. requirements gathering/analysis, design, development etc. Process direction encompasses collaboration which occurs either within value-creation phases of the software development process (horizontal), or the collaboration which occurs in-between value-creation phases (vertical). Process intensity distinguishes between higher and lower flow of information and knowledge between the actors of the process. These are referred to as higher intensity and lower intensity respectively. The process intensity is dependent on either work done collectively, or, on collaborative exchange of information or knowledge between disjoint complementary activities.

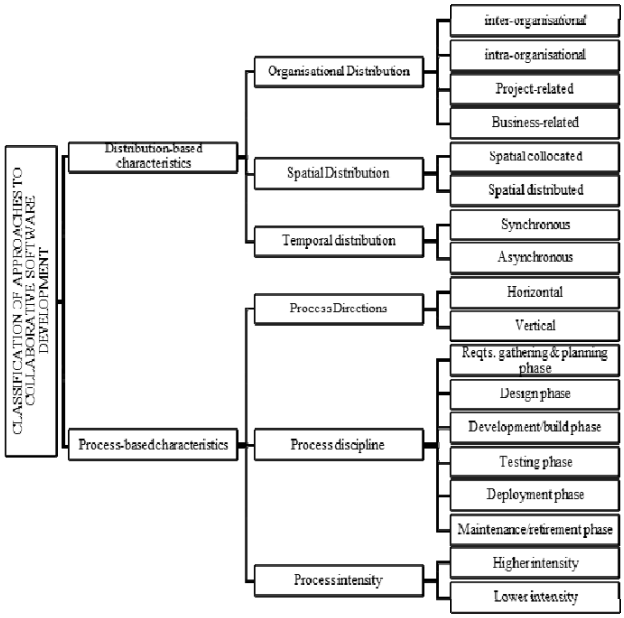

Figure 3: Classification based on both internal and external characteristics of the development process

D. *Classification based on analysis of interactions between all aspects of the process*

The need for more efficient collaboration within the process is driven by increasing distribution within teams, complexity within the process, and need for more efficient

ways of improving quality aspects of software, as well as delivery time, to meet changing needs. However, Cloud-based collaborative software development is yet to reach the level where the practice and interactions amongst all the components of the process is routine [13]. Improving the development process necessitates standardization of collaborative interactions between diverse set of people, skills, activities, processes, tools, configurations, specifications, and other relevant components, across factors such as location, distance, characteristics, objectives and nature of being. Analysis of interactions between all components of the process that contribute to bringing about a successful outcome, yields another basis for classification[4], [35],

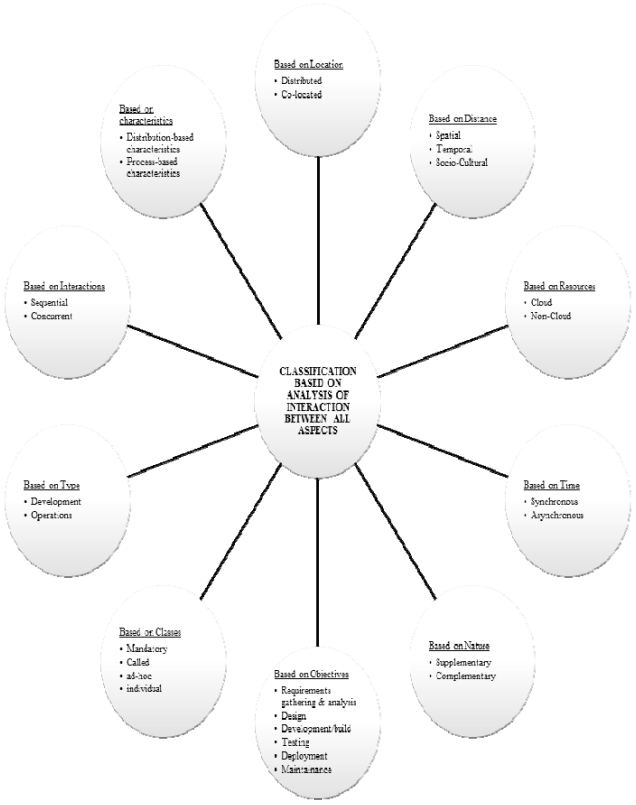

Figure 4: Classification based on analysis of interactions between all development process aspects

IV. Conclusion

Enhancing collaboration in Cloud-based software development requires the ability to recognize and identify various possible collaboration contexts, as well as the need for to vary approaches aimed at these. Knowledge and consideration of the implicit, and sometimes explicit differences present, as a result of varying contextual factors and characteristics, is important for designing collaborative adequately efficient architectures, frameworks and methodologies for the cloud-based development process. It provides awareness of consideration factors, as well as, an understanding of aspects of collaboration and development contexts that are going to influence or impact effectiveness most. This provides a useful means for making trade-offs and selecting most apposite practice and contingencies when seeking to leverage cloud capabilities and design solutions to improve collaboration and efficiency in software development lifecycle process in the cloud.

References

[1] M. Bass, J. D. Herbsleb, and C. Lescher, 'Collaboration in Global Software Projects at Siemens: An Experience Report', in *International Conference on Global Software Engineering (ICGSE 2007)*, 2007, pp. 33–39.

[2] J. D. Herbsleb, D. J. Paulish, and M. Bass, 'Global Software Development at Siemens: Experience from Nine Projects', in *Proceedings of the 27th International Conference on Software Engineering*, New York, NY, USA, 2005, pp. 524–533.

[3] 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.

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

[5] F. Fazli and E. A. C. Bittner, 'Cultural Influences on Collaborative Work in Software Engineering Teams', 2017.

[6] L. M. Riungu, O. Taipale, and K. Smolander, 'Research Issues for Software Testing in the Cloud', in *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, 2010, pp. 557–564.

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

[8] 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.

[9] 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.

[10] 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.

[11] I. Sommerville, *Software Engineering*, 9 edition. Boston: Addison Wesley, 2010.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] B. W. Boehm, 'Some Future Software Engineering Opportunities and Challenges', in *ResearchGate*, 2010, pp. 1–32.

[21] L. Hattori and M. Lanza, 'Syde: A Tool for Collaborative Software Development', in *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 2*, New York, NY, USA, 2010, pp. 235–238.

[22] J. Arsenyan and G. Büyüközkan, 'Modelling collaborative software development using axiomatic design principles', *IAENG Int. J. Comput. Sci.*, vol. 36, no. 3, pp. 234–239, 2009.

[23] T. Clear, 'Dimensions of Collaboration in Global Software Engineering Teams: Explorations of "Collaborative Technology Fit"', in *Fourth IEEE International Conference on Global Software Engineering, 2009. ICGSE 2009*, 2009, pp. 297–298.

[24] 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.

[25] M. Nordio, H.-C. Estler, C. A. Furia, and B. Meyer, 'Collaborative Software Development on the Web', *ArXiv11050768 Cs*, May 2011.

[26] S. Ardaiz, 'Collaborative Communication: Why Methods Matter', *Triple Pundit People Planet Profit*, Dec. 2011.

[27] 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.

[28] 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.

[29] 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.

[30] XebiaLabs, 'Periodic Table of DevOps Tools', *XebiaLabs*, 13-Apr-2015. [Online]. Available: https://xebialabs.com/periodic-table-of-devops-tools/. [Accessed: 26-Feb-2017].

[31] A. Bento and A. K. Aggarwal, Eds., *Cloud Computing Service and Deployment Models: Layers and Management*. IGI Global, 2012.

[32] H. Hajjdiab and Al Shaima Taleb, 'Adopting Agile Software Development: Issues and Challenges', *Int. J. Manag. Value Supply Chains*, vol. 2, no. 3, pp. 1–10, Sep. 2011.

[33] P. N. Robillard and M. P. Robillard, 'Types of collaborative work in software engineering', *J. Syst. Softw.*, vol. 53, no. 3, pp. 219–224, Sep. 2000.

[34] A. M. Magdaleno, 'Balancing collaboration and discipline in software development processes', in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, New York, NY, USA, 2010, pp. 331–332.

[35] D. E. Strode, 'A dependency taxonomy for agile software development projects', *Inf. Syst. Front.*, vol. 18, no. 1, pp. 23–46, Feb. 2016.

[36] P. Barthelmess and K. M. Anderson, 'A View of Software Development Environments Based on Activity Theory', *Comput Support. Coop Work*, vol. 11, no. 1–2, pp. 13–37, Apr. 2002.

[37] 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.