Systematic Review of Trends and Gaps in Collaborative Software Engineering in the Cloud

Stanley Ewenike
School of Computing and Digital Technologies
Staffordshire University
Stoke-on-Trent, United Kingdom
Stanley.Ewenike@staffs.ac.uk

Dr Elhadj Benkhelifa
School of Computing and Digital Technologies
Staffordshire University
Stoke-on-Trent, United Kingdom
E.Benkhelifa@staffs.ac.uk

Prof Claude Chibelushi
School of Computing and Digital Technologies
Staffordshire University
Stoke-on-Trent, United Kingdom
C.C.Chibelushi@staffs.ac.uk

***Abstract*—This paper presents a review of trends and challenges in collaborative Software Engineering. Due to the nature and size of large-scale Software Engineering projects, effective collaboration is important and necessary. Hence, it is not uncommon to see the adoption of a remix of practices, models, methodologies, tools and skills. However, this remix, alongside adoption of emerging paradigms such as Cloud computing, results in factors that undermine collaborative Software Engineering projects. This paper aims to provide a systematic review and analysis of existing trends, models and challenges. This is with a view towards fostering better understanding of factors undermining the collaborative Software Engineering process, as well as, helps to identify motivations, gaps, and issues pertinent to this research area for a more effective process in the Cloud. A systematic approach was employed in this research. This approach is instrumental to identifying relevant primary studies. Its design provides a means for continuity in terms of any future extension to this review.**

Keywords—Collaborative software engineering; software development process; models; trends; cloud computing; collaboration; systematic review

I. INTRODUCTION

Currently in the software development industry, there exist different trends and development environments promoting vendor-specific range of tools. These contribute to the introduction of new factors undermining the collaborative software development process. The result of this include: complexities that undermine collaboration in the process; failure to efficiently capture all related contexts at each stage of the software development lifecycle; oversights, misunderstanding, and lack of synchronized understanding of requirements, artefacts and other related information at each stage of the software development lifecycle; risk of inadequacy of current practices and methodologies; risk of inadequacy of software developed due to lack of explicit theoretically-grounded architectures; increasing chances of vendor lock-in scenarios; standardization issues, compliance, interoperability issues, etcetera.

The factors mentioned above, among others discussed later in this paper, negatively impact collaboration within the development process. This necessitates the need to review and analyze existing trends, models and practices. It is with the aim of identifying motivations, gaps, challenges, and issues pertinent to this research area. This lays the groundwork for synthesizing new approaches, models and theoretical foundations to underpin and adapt suitable methodologies for a more sustainable, context aware, collaborative process.

In order to ensure verifiable findings, as contribution to existing body of knowledge, research in the area of Software Engineering needs the adoption of the structured, systematic literature review approaches [1]. Part of the downside to employing this approach is that it can be resource-intensive and time consuming, thereby, necessitating the need to strike a balance between rigour and requisite effort [1], [2].

II. METHODOLOGICAL APPROACH

This research starts off by systematically reviewing literature from the parent discipline – Software Engineering, to ascertain trends. It then proceeds to review literature related to the research problem area – collaborative software development in the Cloud, along with other related concepts. This was done using an adapted systematic approach [3]. The review analyzes existing body of knowledge with respect to Software Engineering trends, and relating to the software development process. It proceeds to review ccollaborative software development process. It then moves on to review related concepts, and how they could be leveraged to enhance the process. The review builds a case for modifying the existing process, through analysis of existing body of knowledge in the domain area. This is to further strengthen the relevance and need for a more efficient and context-aware development process.

Table 1 presents the query strings used in the search and retrieval of literature for review. This was done using Mendeley, a reference manager useful for finding, storing, managing and correlating academic research materials and libraries [4]. Mendeley was chosen because of its reasonably fair approximation of research databases, such as Scopus. It has one of the largest databases in terms of research articles and journal coverage, and traffic [5]. The search for literature was restricted to a decade timeline. This is to minimize risk of using obsolete and irrelevant information; and to maximize the limited duration of the research project and resources. However, this comes with the risk of missing out on useful foundational knowledge within the research area, due to the restricted scope, in terms of the chosen duration, as well as the research management tool. The effect of this is minimized through additional manual search.

TABLE I. QUERY STRINGS FOR SYSTEMATIC LITERATURE SEARCH

Area of literature search/topic	Query strings	Time span	Number of articles before de-duplication	Number of articles after 1st tier de-duplication
Extending Boehm's Software Engineering trends timeline	(title: "Software engineering trends" AND year: [2010 TO 2017]) OR (title: "trends in Software engineering" AND year: [2010 TO 2017]) OR ((title: "*Software engineering*" AND "*trends*") AND year: [2010 TO 2017])	2010 - 2017	161	**97**
Collaborative Software Development	((title: "distributed software development") OR (title: "collaborative software development") OR (title: "global software development")) AND (year: [2008 TO 2017])	2008 - 2017	1309	**607**
Collaborative Software Development in the Cloud	((((title: "*software development*") OR (title: "*collaborative software development*") OR (title: "*software engineering*") OR (title: "*collaborative software engineering*")) AND (title: "*cloud*")) AND (year: [2008 TO 2017]))	2008 - 2017	118	**76**
Collaboration in Software development	(title: "*collaboration*") AND ((title: "*Software engineering*") OR (title: "*software development*") OR (title: "*cloud*")) AND (year: [2010 TO 2017])	2008 - 2017	356	**277**

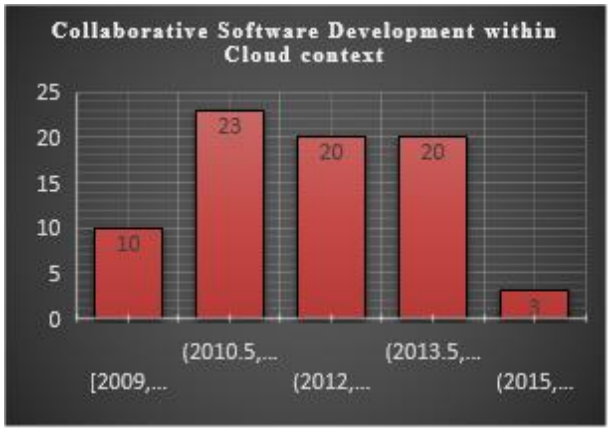

Fig. 1. Decade survey of collaborative software development within cloud context, grouped by year.

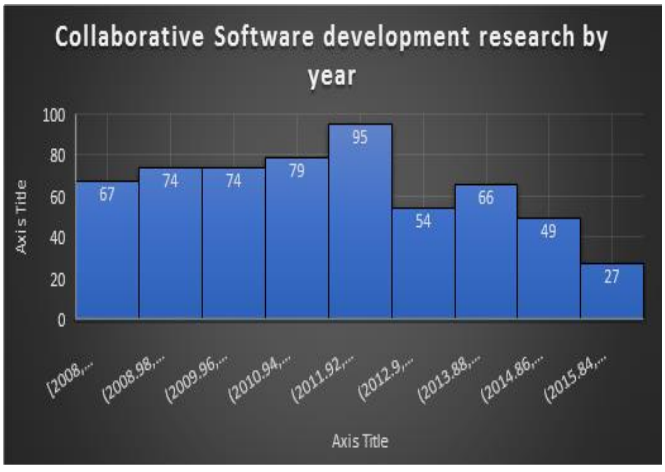

Fig. 2. Decade survey of collaborative software development general research, grouped by year.

1st tier de-duplication involved merging articles with fields where details match, or, are conflicting using the capabilities present in Mendeley [4]. 2nd tier de-duplication involved exporting data in an xml format into Excel. In Excel, it underwent further de-duplication process by using the 'Remove Duplicates' functionality within Excel to easily identify fields that contain duplicate data. Combining these fields to form a composite set allowed further identification and removal of duplicates. This de-duped data table was then normalized, reviewed, and analysed using charts and a combination of methods involving open and axial coding [6]. This helped to identify gaps, challenges, issues, concepts, categories, ideas, and existing relationships and applications. This method was useful for generating themes, patterns and categories, as well as, for testing generated data against any existing data [7]. It was also useful for better understanding and describing the gaps, challenges, issues, concepts, categories, ideas, and existing relationships and applications [8].

The charts in Fig. 1, 2, 3 and 4, highlight the timely relevance of this research project as can be seen from the proximity value of the coefficient of determination, R^2. However, the coefficient of determination, R^2, does not indicate the cause of the relatively lower research effort in this research area, neither does it indicate the level of appropriateness of the chosen independent variable. This approach to literature review, played an important role in definition of research themes, key dimensions, related concepts, as well as facilitating efforts towards the generation of taxonomies and ontologies [9]. The information generated was via analysis and review of data presented, in line with context, experience and understanding of authors, and this research [8].

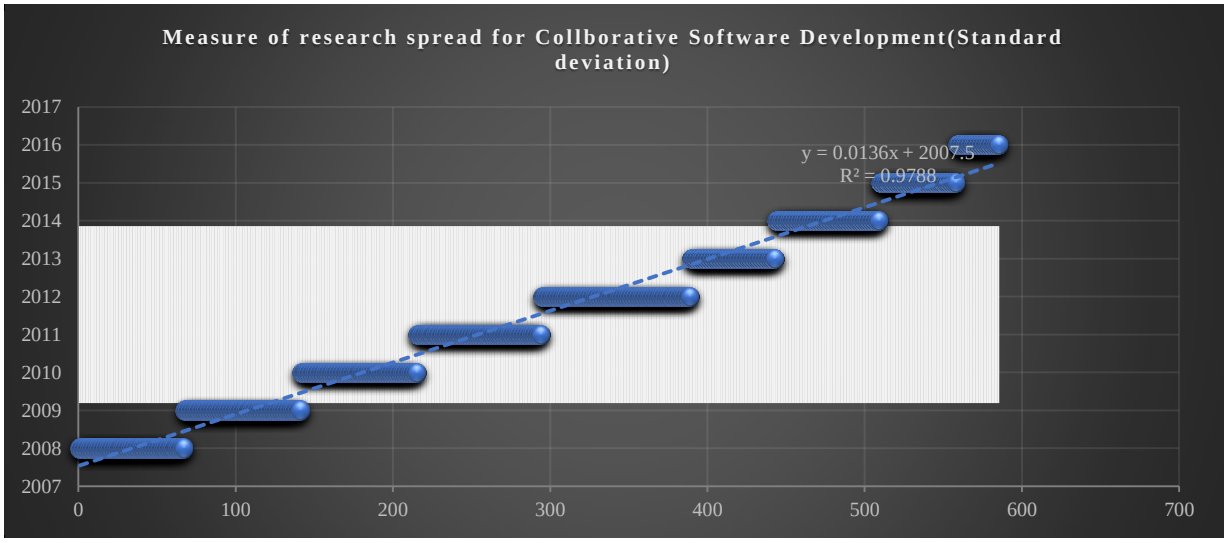

Fig. 3. Measurement of research spread for collaborative software development general research area.

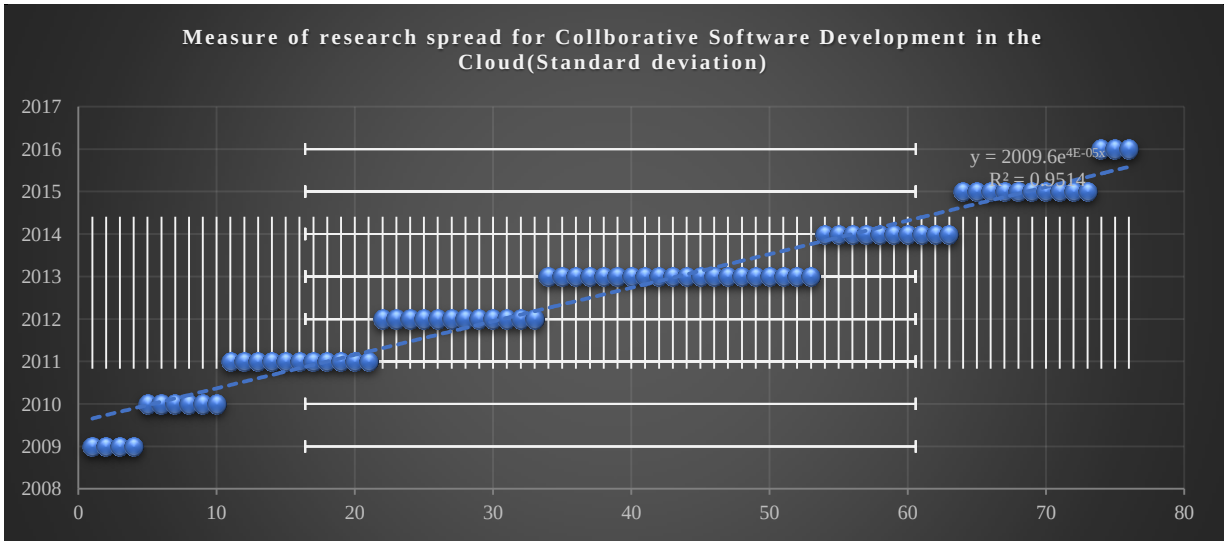

Fig. 4. Measurement of research spread for collaborative software development within cloud context.

III. Review of Software Engineering Trends and its Relevance

Software Engineering is a discipline that seeks to take away randomness in the way software is developed. This is achieved by establishing and applying systematic, disciplined and procedural approaches, principles, practices, frameworks, models, and methodologies to the design, development, and testing of software products and the management of the development process [10], [11]. A typical Software Engineering process involves harmonious interaction between: a set of people with various skills, an environment, tangible and intangible artefacts; towards achieving an end goal [12]. However, factors such as constant changing needs and requirements, affect the harmonious interactions between the different aspect of the process, and ultimately the end goal. This gives rise to a constant need for adequate processes and environments that can adapt or react appropriately to changing contexts, to ensure continuously meeting end goals and outcomes. This need drives Software Engineering trends [13].

The Software Engineering trend timeline in Fig. 5 captures the state of Software Engineering, by identifying various underlying phenomena and trends influencing the evolution of Software Engineering practices. This timeline gives rise to predictions about the future of Software Engineering and the development process, based on observed trend pattern [14]–[16]. Verifying the veracity of these predictions and ascertaining relevance and usefulness, can be done by calibrating the prediction after reviewing the build up to the prediction [17]. Calibrating the prediction helps in identifying the current trends that were predicted, and those that were not predicted. The timeline diagram reveals that the problems of Software Engineering remain fundamentally the same. Over time, these problems have morphed into different forms identified by the different labels or terminologies, and still prevail till date. These include:

- demand, growth and diversity (issues affecting productivity, scalability, collaboration);

- software differences (issues affecting integration, interoperability and compliance); and
- skills shortfall (technological issues).

The timeline also reveals trends that have contributed in ways such as continuous integration, collocation of customers, more simplistic designs, short development builds or increments, and pair programming, etcetera, towards collaborative software development e.g. Agile methods, etc. However, the impact of these contributions have been mostly felt in small projects, but not so much in larger projects [14], [16]. The trends timeline positions collaboration as a spotlight issue of this decade, because of factors such as: scale issues, clashes in models, platforms and technologies, global connectivity issues, business needs and requirements, efficiency, and security issues. All these contribute to spur on research and development efforts, and resulting into new trends [2], [3], [14], [18]–[25].

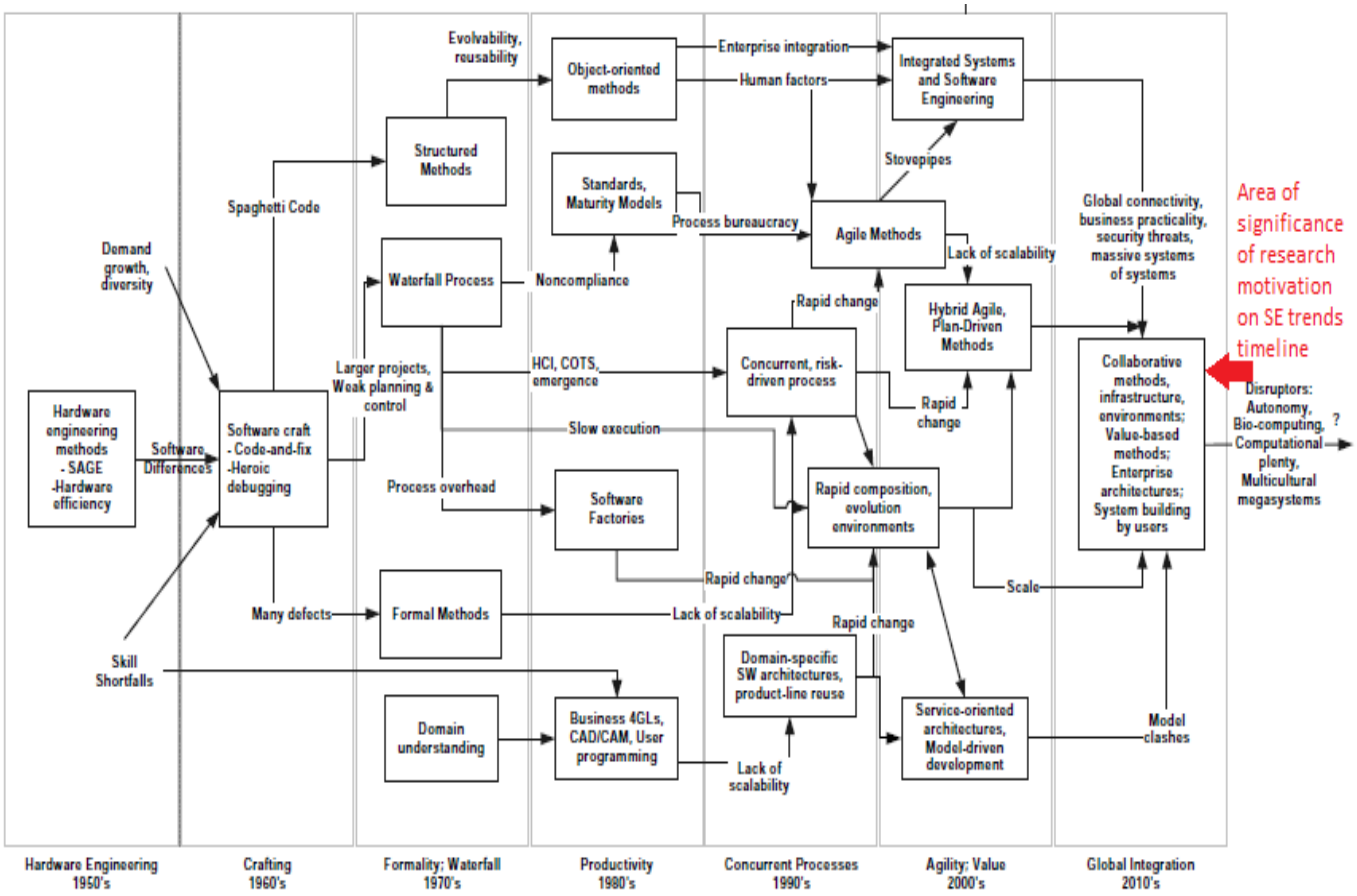

Fig. 5. Timeline of Software Engineering trends.

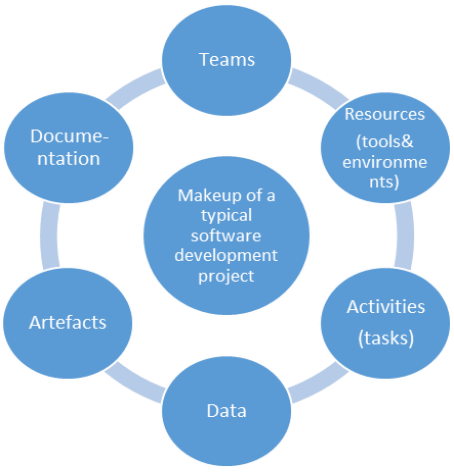

Fig. 6. Typical makeup of a Software development project.

Investigating and developing better ways of tackling issues and challenges in collaborative software development is not just about another trend in Software Engineering. It is more about responding to both existing and evolving Software Engineering and business needs, in alignment with, available resources and technologies of the time [26]. Equally important is the development and implementation of practices and context-aware mechanisms, in line with predicted future trends likely to influence Software Engineering [14]–[16], [27], [28]. This is towards allowing Software development processes and practices to adapt and evolve appropriately. It is also about fostering better understanding of considerations for planning and developing right approach, architecture, strategy, process and support mechanisms, to enhance and sustain collaborative software development processes.

The trends, though a means to an end, introduce factors such as complexity and diversity due to distribution, and

differentiation at different levels. These include: hardware level, the software level, cultural aspects, and the software development activity phases. The identified trends undermine and impact the inherent existing collaboration within collaborative software development processes. This results in the need for more efforts toward supporting the existing collaboration. It also emphasizes the need for more efforts towards enhancing and creating more context-aware collaborative processes that would be adaptive, or harder to undermine. Analysis of the trends timeline and the calibration of the predicted trends highlight increasing dependence of organizations, products, services and systems. It indicates:

- a gradual trend of software-defined or software-enabled ecosystems;
- a need for competitive differentiation;
- rapid adaptability to change;
- a need for facilitation of rapid adaptation of products to align with business and client requirements; and
- a need for reliability and security of software-defined systems or ecosystem.

Addressing these needs will entail changes to the way software is defined, designed, developed, and deployed.

IV. Overview of Collaborative Software Development Lifecycle Process

The Software development process refers to the entire process of developing software. It includes: a team, interactions, framework of activities, set of practices providing guidelines for designing, developing, testing, deploying, maintaining and managing software [29]. This process spans the entire development lifecycle from conceptualization of software, to the retirement or decommissioning of the software. This process is usually embodied in a defined high-level abstraction usually referred to as a software development model [30]. The stages of the development process are not always set in stone, neither are the boundaries of the stages always clearly delineated or differentiated [31], [32]–[34], but the activities are usually, by consensus [40]–[42], centered around addressing questions like:

- What needs to be done – requirement gathering and analysis
- How to do what needs to be done – design
- Doing what needs to be done – development
- Verifying, validating and evaluating the solution – testing
- Deploying or handing over the solution to client or customer or user - deployment
- Ensuring the solution remains useable – maintenance.

Addressing these questions above, gives rise to tasks or activities, which make up stages or phases of the software development process. Software development models are used to facilitate and coordinate these tasks or activities to transform problem definitions and requirements into software [29], [30], [33], [35]. Table 2 presents a cross-sectional summary and comparison of some popular software development models.

The collaborative software development lifecycle process, simply put, refers to how all stakeholders within a software development project, work together on various activities, throughout the software development lifecycle, to achieve a common goal or outcome [36]. The goal in this instance refers to the design, development, and release or deployment of the software. This collaborative development process is one giant activity, made up of sub-activities, involving requirements that undergo transformations via interactions, to yield knowledge-based artefacts. The artefacts from preceding activities mediate and influence succeeding activities. They also form the basis for verifying and validating each stage of the process, until the end goal is achieved. A typical software development project usually comprise: a team, made up of people of diverse cultures, skillset, technical expertise, technological and non-technological viewpoints, either. This team work together on different tasks, or separately on complementary tasks, at different stages of the process, towards a common goal. This calls for efficient collaboration and management in the software development process via a variety of tools or medium [27], [41] (see Fig. 6).

TABLE II. CROSS-SECTIONAL COMPARISON OF SOFTWARE DEVELOPMENT MODELS

Category	**Differentia-ting Aspects**	**Software development models**		
		Plan-driven	***Agile***	***FOSS***
Characteristic Themes	Underlying Philosophical objective	Seeks to establish and ensure reliability, predictability and stability	Seeks quick ways of adding value to business, as well as adaptation to changes	Mainly seeks to ensure freedom for user
	Disciplined definition	Formal – Defined stages and activities	Formal – Agile Manifesto	Informal – works on voluntary collaboration
	Development Cycles	Sequential and Relatively longer	Iterative + relatively shorter + more focus on testing	Iterative + relatively shorter + more focus on testing + *free software*
	Focus of development activities	Sequential processes and documentation	Customer collaboration	User participation and four freedoms – run code, study code, improve code, and distribute code
	Location emphasis	Favors both co-location and geographically distributed stakeholders or team members	Emphasis on co-location	Favors both co-location and geographically distributed stakeholders or team members
	Release period	Relatively less frequent	Relatively more frequent	Same as in Agile
	Documentation	Relatively more documentation	Relatively less documentation	Same as in Agile
	Client involvement	Relatively lower	Relatively higher	Relatively lower
	Reliance on tool support for development tasks	Yes	No	Yes
	Overall goal	Improvement of software development process	Improvement of software development process	Improvement of software development process
	Other			
Examples		Waterfall, Unified process (e.g. as implemented in IBM's Rational)	Extreme, Scrum, Kanban, Crystal, Rapid Application Development (RAD), Lean Development methodology,	GNU, Linux, Apache, Mozilla
Typical number of activity stages	0-4	No	Yes	Yes
	5-9	Yes	No	No
Prominent challenge and issues		Less client involvement, long development times, inflexibility with management of changes in requirements, delays and development backlog, more predictive than reactive	Less concrete planning, size of team is relatively smaller, less emphasis on documentation, can be tasking for the team in terms of time commitment, product evolution may be quite different from that envisaged, more reactive than predictive	Varies
Similarities and dissimilarities	Communication	Emphasis on formal communication	Emphasis on informal communications	Varies
	Control/Management	Approaches and implements control through structure	Approaches and implements control through flexibility	Varies
	Planning	Tends to be more upfront	Tends to be on-going as and when	Varies

V. Collaborative Software Development in the Cloud

One of the most adopted definition of Cloud computing, defines the paradigm as "…. 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" [43], [44]. This definition captures main characteristics Cloud computing, which constitute some of the most attractive features of the Cloud, as well as, represents the strengths from where a lot of the benefits attributed to the Cloud come from. Pre-Cloud setups were characterized by: reliance on silo-like architectures that were difficult to scale; resource waste; complex administrative and management functions; less agility towards change; high capital and operational costs [40], [42], [43]. Benefits of Cloud computing include [37]–[39], [44]–[48]:

- Enabling opportunities for sustainable models of computing and businesses;
- relatively higher degrees of flexibility, productivity and scalability;
- faster and larger scale of computation, processing and sharing;
- wider accessibility and greater availability;
- cost savings and efficiency;
- scalable resources for storage, backup and recovery;
- relatively easier setting up of customized environments and quicker deployments;
- agility;
- facilitation of innovation and R&D;
- adaptability;
- extensibility and opportunities for all stakeholders to collaborate; and
- provides framework or platform for integrating technologies and platforms to promote more sustainable strategies.

The advent of Cloud computing has brought about a myriad of service provisioning options. This has resulted in the consumption of resources as services on a pay-per-use basis. This greatly favours organizations and companies with limited resources [49]. Since the emergence of Cloud computing, more efforts are now directed towards exploiting and leveraging cloud computing for the range of benefits and advantages it offers, mostly as services. This is evident in the range of Cloud applications and services springing up and used by organizations [50], [51]. However, there are certain challenges and issues in Cloud Computing which would need consideration [52], [53], [55], [56]. These include:

- security issues;
- vendor lock-in and interoperability issues;
- portability issues;
- efficiency of automation considerations;
- performance issues;
- availability and integrity of relevant information;
- handling uncertainty about heterogeneity, content type, location of client, bandwidth unpredictability, dynamic workload variations, workflow schedules, architecture and resource optimization; and
- context awareness and reproducibility within contexts.

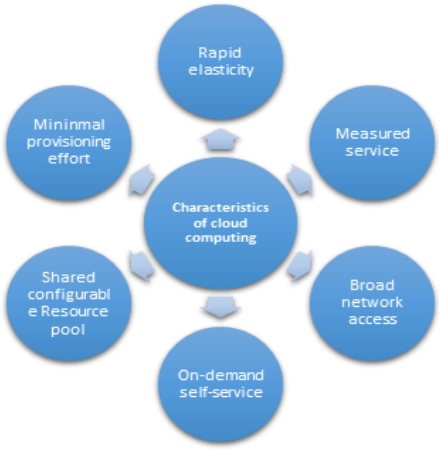

Fig. 7. A representation of Cloud computing characteristics.

Some of these issues mentioned above are partly inherited, due to Cloud Computing being a paradigm that leverages a couple of other technologies [49], [51], [57]. Effect of the changes brought about by the Cloud Computing trend can be seen in the paradigm shift from use of desktop IDEs to Cloud APIs, in building software projects. Various Cloud services providers have their own API offerings, often built on top of their IaaS offerings [54]. The benefits and advantages offered by the Cloud, amongst other features and characteristics makes the case for the suitability of Cloud Computing for Cloud-based Collaborative Software Development. The Cloud allows for rapid provisioning of resources as web services, which could be harnessed to achieve much needed rapid responsive development, and provide the environment that could enhance the collaboration needed [50], [58]. Leveraging the Cloud would require the adaptation of existing collaborative software development processes to align with the capabilities of the Cloud. However, this is likely to raise issues for legacy applications and existing management practices and methodologies of software development projects [54]. Issues to contend with include: considerations regarding existing processes, applications, practices and methodologies used in Software development projects. An attempt is made in this research project to summarize strengths, weaknesses, opportunities, and threats of Cloud computing via SWOT analysis to highlight aspects of Cloud Computing that need to be critically considered and evaluated, and others that need to be further exploited for more benefits. Table 3 summarizes the findings from the SWOT analysis carried out.

Majority of R&D efforts in Collaborative software development process in the Cloud concentrate mostly, on specific aspects of the process, more than others, giving rise to lopsided or unbalanced collaboration within the process. Efforts devoted towards Collaborative Software Development life cycle process in general, and within Cloud context, have been mainly in the areas of [29], [30], [67]:

TABLE III. A SWOT ANALYSIS OF CLOUD COMPUTING

STRENGTHS (*INTERNAL*)	**OPPORTUNITIES (*EXTERNAL*)**
• Scalable and elastic infrastructure • On-demand self-service • Measured usage: pay-as-you go • Agility. Ease of resource provisioning and pooling • Broad network access • Provider assurances of over 95% availability rate • Minimal management effort • Regular and easy update	• Shared resources • Broad network access. Promotes mobility and accessibility • Scalable and elastic infrastructure • Ease of resource provisioning • Ease to setup and ease of implementation • Service-nature of resources e.g. accessing resources as web services
WEAKNESSES (*INTERNAL*)	**THREATS (*EXTERNAL*)**
• Absence of universally accepted Cloud interoperability standards • Requires a fast and constant internet connection for best performance • Dependency on provider, to an extent • Inability to predict peak and trough periods for resource usage • Service agreement changes and API changes • Auditability of services/data	• Legislative issues: lack of international regulatory legal precedents or framework • Security issues, privacy and risks such as insider threat • Ownership of data and services • Scheduled and unscheduled service failures and outages

- asynchronous collaboration;
- more support for coding and deployment stages of the software development process;
- collaborative software development from the standpoints of trust and privacy;
- collaboration in isolated aspects of the software development process, such as coding activities, or design activities, or testing activities, version control repositories;
- non-cloud-based collaborative software development process;
- use of open-source tools for contributing, improving, and managing code; and
- integrating social networking and communication features with the development process.

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 [27], [59]–[61]. Leveraging the Cloud to enhance the development process is necessary for the following reasons:

- To help address inefficiencies and inconsistencies of the traditional process and environments.
- To align the development with current trends and changing business requirements.
- To leverage new concepts, frameworks and methods for a more optimal development process.
- For economies of scale and efficient use of resources, tighter collaboration, efficient management from automation and context-aware linking and sharing of information.

TABLE IV. SUMMARY OF GAPS IN CLOUD-BASED COLLABORATIVE SOFTWARE DEVELOPMENT

Main gaps identified	Summary Comments
➢ Need for Cloud-based context-aware Collaborative software development architectures with explicit theoretical foundation **[3], [55], [59]–[63]**	➢ Emerging technologies change the way software is accessed, utilised, stored and maintained. They introduce or emphasize new considerations such as: distribution, more complexity and more contexts. There is need to develop reliable software for continuous adaptation to changing requirements. ➢ Current innovative solutions rely on results from mix of successful and failed implementations, as well as glitches. **Observed impact include:** i) Randomness in the science of Software Engineering process. ii) Undermined collaboration in collaborative lifecycle development process. iii) Emphasis on need for better and sustainable frameworks, architectures, tools, and strategies, with explicit theoretical foundations for more structured adaptation and sustainable collaboration. iv) Need for adequate methods for managing change in the Cloud-based development process in the Cloud, and knowledge creation.
➢ Need for effective capture and representation of context data and all related data across entire life cycle process, in a Cloud-agnostic format for generation of actionable insights [2], [15], [22]–[25], [64]–[67]	➢ Insufficient context data and other related data are sometimes poorly collected, completely missed, ignored, misunderstood, or poorly applied. ➢ Requirements, artefacts from various activities, action plans, feedback, and other important related information necessary to achieve a goal are sometimes not clearly and accurately defined, and agreed upon by all concerned. **Observed impact include:** i) Negative impact on balancing and optimizing of flow of information within the development environments and teams. ii) Late detection and resolution of issues and bugs that could have been otherwise avoided if sufficient context data are collected, and taken into consideration and applied within activities. iii) Inadequate tracking of project progress. iv) Conflicts in perspectives, understanding, interpretation and execution of activities. This often results in defective software, or software needing more rework.
➢ Need for effective ways for managing complexity across stages of Cloud-based life cycle development process to ensure synchronous collaboration and verifiable outputs/outcomes at various stages of the process [1], [20], [46], [64], [69]	➢ 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 by physical laws. **Observed impact include:** i) Growth in complexity of software artefacts and the life cycle 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.
➢ Need for adequate ways for benchmarking Cloud-based collaborative development and testing [17]–[19], [21], [62], [68]	➢ The existing standards are not adequate and mostly generic. Does not expressly cater for analysis, assessment and measurement of the Cloud-based collaborative development process.

There is growing activity from industry in Cloud-based collaboration, with a lot of emphasis in content management, sharing and storage, but relatively less in collaborative software development. Although some notable industry players, have managed to make breakthroughs in collaborative Cloud-based software development, there is little detailed documentation available [37], [46]. Table 4 presents a summary of the most commonly identified issues and challenges, which presents as gaps.

VI. CONCLUSION

For effective collaboration to occur within the software engineering process in the Cloud, mere communication and coordination are not enough. Companies who have transitioned their development environments to the Cloud, have started realizing benefits such as: cost reduction in hardware; relatively accelerated software development life cycle process via reduction of time and effort needed to set up development and testing environments; unified management; service and functionality expansion; on-demand provisioning and access to resources and development environments. Collaborative Software development lifecycle process in the Cloud, presents complexities and contexts, amidst other factors, that need to be considered during the process. 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 lifecycle development process in the Cloud. This is one of the motivations for this research.

Other motivations include: need for identification of reliable ways of managing and measuring collaboration and other success factors within the process; need for new methodologies and ways of enhancing effective collaboration within the lifecycle development process; need for effective ways of managing complexity and ensuring synchronous regularity, as well as, verifiable outputs and outcomes at the

various stages of the collaborative development process. Also, development of key dimensions for analysing and benchmarking the process is essential for continuous process improvement and sustainability. Being able to consistently reproduce the enhanced process would require standardization in the form of frameworks, architectures and standards.

REFERENCES

[1] H. Zhang and M. Ali Babar, 'Systematic reviews in software engineering: An empirical investigation', Inf. Softw. Technol., vol. 55, no. 7, pp. 1341–1354, Jul. 2013.

[2] B. A. Kitchenham, T. Dyba, and M. Jorgensen, 'Evidence-based software engineering', in Proceedings of the 26th international conference on software engineering, 2004, pp. 273–281.

[3] B. Kitchenham and S. Charters, 'Guidelines for performing systematic literature reviews in software engineering. Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report', 2007.

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

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

[6] C. Auerbach and L. B. Silverstein, Qualitative Data: An Introduction to Coding and Analysis. NYU Press, 2003.

[7] C. Grbich, Qualitative Data Analysis: An Introduction. SAGE, 2012.

[8] T. Basit, 'Manual or electronic? The role of coding in qualitative data analysis', Educ. Res., vol. 45, no. 2, pp. 143–154, Jun. 2003.

[9] E. H. Bradley, L. A. Curry, and K. J. Devers, 'Qualitative Data Analysis for Health Services Research: Developing Taxonomy, Themes, and Theory', Health Serv. Res., vol. 42, no. 4, pp. 1758–1772, Aug. 2007.

[10] K.-J. Stol and B. Fitzgerald, 'Uncovering Theories in Software Engineering', 2013.

[11] C. Ghezzi, M. Jazayeri, and D. Mandrioli, 'Fundamentals of software engineering', 2002.

[12] M. DEVLIN and S. DRUMMOND, 'Software Engineering Students' Cross-Site Collaboration: An Experience Report.', 2007.

[13] S. L. Pfleeger, Software engineering: theory and practice. Prentice Hall, 2001.

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

[15] 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.

[16] B. Boehm, 'Some future trends and implications for systems and software engineering processes', Syst. Eng., vol. 9, no. 1, pp. 1–19, Mar. 2006.

[17] J. Münch and K. Schmid, Perspectives on the Future of Software Engineering: Essays in Honor of Dieter Rombach. Springer Science & Business Media, 2013.

[18] 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.

[19] 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.

[20] M. Mohtashami, V. Kirova, T. Marlowe, and F. Deek, 'A Comparison of Three Modes of Collaboration for Software Development', AMCIS 2009 Proc., Jan. 2009.

[21] T. Marlowe, 'Addressing Change in Collaborative Software Development: Process and Product Agility and Automated Traceability'.

[22] N. Jastroch, 'Advancing Adaptivity in Enterprise Collaboration', Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 1907348, Nov. 2009.

[23] 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.

[24] 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.

[25] 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.

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

[27] Finkelstein and J. Kramer, 'Software engineering: a roadmap', in Proceedings of the Conference on The Future of Software Engineering, New York, NY, USA, 2000, pp. 3–22.

[28] J. D. Herbsleb, 'Global Software Engineering: The Future of Socio-technical Coordination', in 2007 Future of Software Engineering, Washington, DC, USA, 2007, pp. 188–198.

[29] Fuggetta, 'Software process: a roadmap', in Proceedings of the Conference on The Future of Software Engineering, New York, NY, USA, 2000, pp. 25–34.

[30] Sommerville, Software Engineering, 9 edition. Boston: Addison Wesley, 2010.

[31] 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.

[32] 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.

[33] 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.

[34] J. Feller, B. Fitzgerald, and others, Understanding open source software development. Addison-Wesley London, 2002.

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

[36] 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.

[37] L. Barnett and C. E. Schwaber, 'Applying open source processes in corporate development organizations', Forrester Res., pp. 1–15, 2004.

[38] P. Mell and T. Grance, 'The NIST definition of cloud computing', 2011.

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

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

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

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

[43] 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.

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

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

[46] Benkhelifa, M. Abdel-Maguid, S. Ewenike, and D. Heatley, 'The Internet of Things: The eco-system for sustainable growth', in 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA), 2014, pp. 836–842.

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

[48] 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.

[49] 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.

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

[51] S. M. Hashemi and A. K. Bardsiri, 'Cloud Computing Vs. Grid Computing', 2009.

[52] 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.

[53] 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.

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

[55] 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.

[56] 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.

[57] H. K. Buhrer, 'Software Development: What It is, What It Should Be, and How to Get There', SIGSOFT Softw Eng Notes, vol. 28, no. 2, p. 5–, Mar. 2003.

[58] 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.

[59] P. Ralph, 'Software Engineering Process Theory: A Multi-Method Comparison of Sensemaking-CoevoIution-Implementation Theory and Function-Behavior-Structure Theory', ArXiv13071019 Cs, Jul. 2013.

[60] P. Ralph, 'Possible Core Theories for Software Engineering'.

[61] 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.

[62] G. Mark, 'Extreme Collaboration', Commun ACM, vol. 45, no. 6, pp. 89–93, Jun. 2002.

[63] 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.

[64] 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.

[65] M. Mohtashami, T. J. Marlowe, V. D. Kirova, and F. P. Deek, 'Risk-driven Management Contingency Policies in Collaborative Software Development', Int. J. Inf. Technol. Manag., vol. 10, no. 2–4, pp. 247–271, Jan. 2011.

[66] V. Pankratius, Emerging Research Directions in Computer Science: Contributions from the Young Informatics Faculty in Karlsruhe. KIT Scientific Publishing, 2010.

[67] M. Richards, Software architecture patterns, 1st ed. O'Reilly Media, Inc., 20015.

[68] 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.

[69] E. M. Bouwers, 'Metric-based Evaluation of Implemented Software Architectures', 2013.