iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** V **Month of publication:** May 2024

DOI: https://doi.org/10.22214/ijraset.2024.61421

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

Code Simulator with Real-Time Conferencing and Considerable Users

Mohd Shams Tabrez[1], Riya Sharma[2], Dr. Sadhana Rana[3]
[1, 2]*Computer Science and Engineering, SRMCEM, Lucknow, India*
[3]*Professor, Computer Science and Engineering, SRMCEM, Lucknow, India*

Abstract: ***The Multiuser Live Text Editor represents a dynamic platform fostering collaborative coding endeavors, enabling users to engage in real-time interaction while collectively crafting code. Boasting compatibility with prominent programming languages such as C++, Java, and Python, this innovative tool is founded upon the notion of adaptive functionality, a cornerstone of its shared editing capabilities. At its core lies pair programming, an agile methodology originating from Extreme Programming (XP), wherein two developers synergistically collaborate on a single computing environment. Tasked with jointly conceiving, implementing, and validating user stories, these pairs operate in tandem, leveraging their complementary skills to achieve shared objectives. Notably, this collaborative process thrives on the diversity of the paired developers' systems, with each individual allocated equal time at the keyboard. This integration of differing perspectives enriches problem-solving and innovation. Within this framework, one participant assumes the role of the driver, actively engaging in code composition, while the other adopts the navigator position, providing overarching guidance and strategic direction.***

This collaborative ethos transcends physical boundaries, facilitating seamless cooperation between developers irrespective of their proximity. Whether collaborating face-to-face or remotely, the essence of pair programming remains rooted in robust communication. Through constant dialogue and information exchange, developers navigate complexities collectively, resolving challenges that might otherwise pose significant obstacles. The symbiotic relationship between drivers and navigators mirrors that of travelers sharing a journey, as they communicate, exchange insights, and collectively surmount obstacles. By fostering a collaborative environment conducive to knowledge sharing and problem-solving, pair programming emerges as a potent methodology for enhancing productivity and code quality alike.

Keywords: Code Simulator, Peer Code Simulator, Socket io, PeerJS, Web RTC, Peer Server, API.

I. INTRODUCTION

The concept of 2code embodies the collaboration between multiple individuals and a shared computing environment, facilitating collective coding efforts. Despite its name suggesting two individuals and one machine, it actually accommodates multiple participants, fostering a collaborative workspace where programmers unite to tackle coding challenges. Within this application, each pair of programmers is equipped with both a keyboard and a mouse, enabling seamless interaction as they jointly navigate the coding landscape. In this collaborative setting, one programmer takes on the role of the coder, tasked with actively writing and implementing code, while the other assumes the position of the reviewer, meticulously examining the code to ensure alignment with project requirements. The reviewer's responsibilities extend to noting, analyzing, and pinpointing errors, as well as suggesting potential next steps for refinement or enhancement. Crucially, the coder's focus lies solely on crafting the code effectively, confident in the knowledge that the reviewer will diligently verify its suitability for the project. This division of labor is not static; roles can transition fluidly, with the coder assuming the role of reviewer and vice versa as needed. Such flexibility fosters a symbiotic partnership, maximizing efficiency and facilitating seamless code debugging.

Equality underpins this collaborative framework, with both programmers possessing equal capabilities and sharing keyboard time equitably. Within this context, one iteration of pair programming designates the programmer at the keyboard as the controller, responsible for executing code implementation, while the navigator assumes the role of guiding the programming process towards its desired outcome. This collaboration transcends physical boundaries, accommodating both face-to-face and remote interactions, thereby catering to diverse working preferences and environments.

Communication serves as the lifeblood of this collaborative endeavor, facilitating dialogue between the coder and navigator as they collectively navigate coding challenges, discuss strategies, and troubleshoot issues that may elude individual diagnosis. However, it's worth noting that while pair programming represents an agile software development methodology with myriad benefits, it may not be suitable for every team or project.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

Effective participation in pair programming necessitates a blend of soft skills for teamwork and collaboration, alongside the requisite hard skills for coding and testing. Consequently, the adoption of this collaborative approach may vary among businesses, with some embracing its potential for enhancing productivity and code quality, while others may opt for alternative methodologies.
The process commences with developers defining clear objectives, often in the form of small, manageable tasks, such as counting, measuring, or writing code snippets. Feedback and corrections are provided at designated intervals, ensuring continuous improvement without disrupting the coder's workflow.

II. PROBLEM STATEMENT

Pair programming is a collaborative software development approach where two developers collaborate on a single workstation. Within this setup, one developer assumes the role of the driver, actively writing and implementing code, while the other takes on the role of the navigator, offering guidance and suggesting implementation strategies.
The driver utilizes the workstation to input code, while the navigator concurrently reviews the code in real-time, offering insights and direction. Roles are alternated regularly to ensure both developers have equal opportunities to contribute ideas and translate solutions into executable code.
Typically, the navigator's responsibility encompasses setting project direction and advising on implementation strategies. Pair programming scenarios can involve developers with similar skill levels and experience, or pairings may consist of a senior developer mentoring a junior counterpart.
The collaborative nature of pair programming fosters synergy between team members, enhancing collaboration and leading to the creation of superior products. While pair programming may seem complex at first glance, it serves as a valuable tool for fostering teamwork and elevating development processes.

III. LITERATURE REVIEW

Marina Pimenova. (2001) According to Marina Pimenova Seven out of the ten PP studies regarded paired skill level as one of the determinant factors of PP's effective-ness the two categories of skill level used were actual and perceived skill. The actual skill level was determined based on programming experience, academic background, and students' academic performance. Perceived skill level was measured subjectively according to the skill of a student's partner relative to their own perceived skill (i.e. "better", "about the same", or "weaker"). The consensus from these studies is that PP works best when the pair has a similar skill level. However, two correlation studies show contradictory findings on the association between students' skill level and PP's effectiveness. Muller and Patberg report there is no correlation between the two variables and Made ski refutes this finding.
Yu Kong. (2007) According to Yu Kong the two studies that investigated the effect of Felder-Silverman learning style reported that learning style did not significantly affect pair compatibility or the perception of students towards pairing. In terms of work ethic, Williams et al. report that pairing students of similar work ethic enhances pair compatibility, and Layman reports that students' perception to-wards pairing is not affected by their work ethic. Williams et al. also investigated students' time management ability and found it has no effect on pair compatibility. In 2004, Muller and Patberg coined the term "feel-good" which refers to how comfortable pairs feel during the PP session. They report that the feel-good factor is correlated with a pair's performance. Made ski [S68] had similar findings where a positive correlation between the feel-good factor and pair performance (quality of software) was found.
Deepak Kumar. (2019) According to Deepak Kumar PP's effectiveness was measured using various factors, organized in four categories: technical productivity, pro-gram/design quality, academic performance, and satisfaction. Technical productivity, measured by 31 (44%) of the 70 studies was the most common method used to assess PP's effectiveness, followed by program/design quality (30 studies, 43%). A subset of 16 studies (23%) evaluated PP's effectiveness based on students' academic performance in final exams, mid-terms, assignments, projects, and course grades. Besides the objective measurements, PP's effectiveness was evaluated subjectively in 22 studies (31%) using students' perceived satisfaction experiencing PP sessions.
Shruti Kothari. (2020) According to Shruti Kothari Pair Programming (PP) - all production code is written by two people at one screen/keyboard/mouse. Pair programming is a collaborative approach that makes working in pairs rather than working in individual for code development One programmer writes a software artifact (e.g. program code or UML diagrams) and other programmer continuously assures quality of the software artifact by watching, asking questions, looking for some alternative approaches, helps to avoid defects etc. The two programmers switch their roles after some time: creator, is also called Driver becomes quality assurer, is also called the Navigator, and vice versa

Dybå & Dingsøyr. (2008) In 2008, Dybå & Dingsøyr carried out a SLR of Agile Software Development empirical studies to find the empirical evidence for benefits, limitations, and strengths of agile methods. They found low strength of evidence supporting PP in agile methods. However, in 2007, Dybala et al. conducted a SLR focusing on effectiveness of PP. The study investigated the empirical evidence and supported the claims that PP is more advantageous than solo programming. The PP 's aspects investigated were related to effectiveness focusing ―duration‖ (time spent to produce the system), ―effort‖ (person hours spent), and ―quality of the final product‖. The SLR as an intermediate analysis was extended by Hanna et al as a full-scale analysis in 2009, which summarized pair programming experiments published up to and until 2006. In addition, the studies published up to August 2007 were also taken into account.

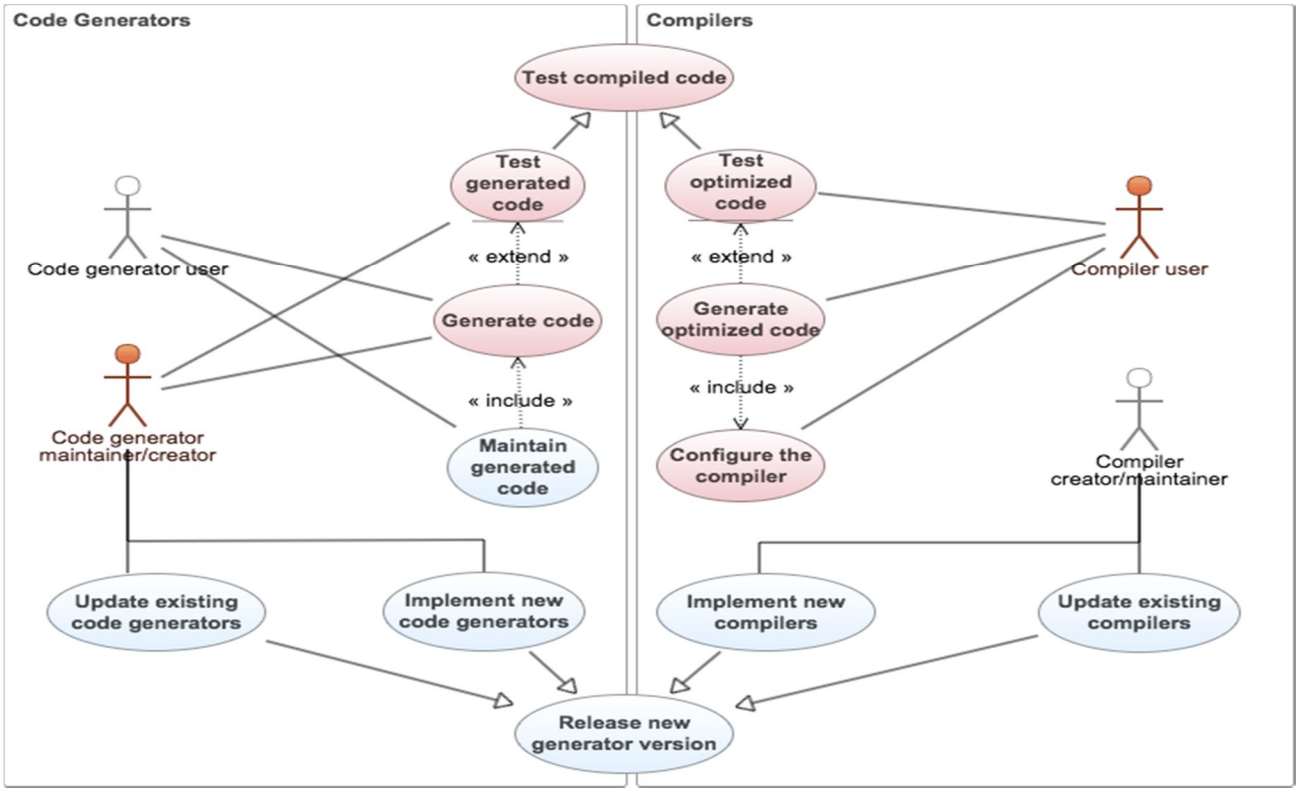
IV. METHODOLOGY

The proposed methodology in order to do Realtime coding with multiple users as follows –

1) A GUI which shows the editor with options of different programming languages, and option of video call and invite too.
2) After opening the site, the user will see the editor, and can start coding with multiple users, with the help of invite button. • The user will share the room code with other users to join the same room.
3) Other user will join the room by invite link, and can start coding in the same room.
4) Users can also do the video call for better understanding and code sharing.
5) In This way The Multiple users can collaborate and code efficiently.
6) Users can also have multiple themes for text editor.
7) It Can Be used for Teaching purpose, interviews, small-level

The main objective of our project is:

a) *Real Time Editing:* Edit code in a real time editor. Input and output are also real time.
b) *Run Code:* Run code against a custom test case in currently supported languages.
c) *Private Channel:* Only the users who have access to the unique URL can edit in the editor.
d) *Video Calling:* Users can interact via video calling.
e) *Rich text Presence:* Highlight the cursor position of another user.



ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

V. MODULE DESCRIPTION

1) *Video Capture Module:* Video capture module includes the detection of real time video frame captured using the user's web camera (live feed).
2) *Code Editor Module*: Code editor Module will help in editing code. We are using ACE (Ajax.org Cloud9 Editor). Ace is a standalone code editor written in JavaScript. Our goal is to create a browser-based editor that matches and extends the features, usability and performance of existing native editors.
3) *Real Time Communication Module*: For Realtime communication we are using socket.IO. Socket.IO allows bi-directional communication between client and server. Bidirectional communications are enabled when a client has Socket.IO in the browser, and a server has also integrated the Socket.IO package. While data can be sent in a number of forms, JSON is the simplest.
4) *Compiler Module*: For Compiler we have use Compile Run library. It has different languages it provides different compiler facilities that are needed for coding.
5) *Audio Module*: Audio capture module includes the detection of real time audio captured using the user's Microphone.

VI. IMPLEMENTATION OF REALTIME COMMUNICATION

Socket.IO facilitates real-time, bidirectional communication between client and server, offering low-latency interaction through event-based messaging. Leveraging the WebSocket protocol as its foundation, Socket.IO extends functionality by supporting fallback mechanisms like HTTP long-polling and seamless reconnection. This versatility ensures robust connectivity over TCP, abstracting the complexities of WebSocket connections for developers.

Peer.js enhances peer-to-peer (P2P) communication by wrapping the WebRTC implementation within browsers. By providing a comprehensive and customizable API, Peer.js simplifies the establishment of direct connections between peers using only unique identifiers. This approach eliminates the need for developers to handle STUN servers, ICE candidates, or maintain their own servers, streamlining the implementation process. Moreover, Peer.js offers a dedicated server, PeerServer, facilitating seamless connection establishment between Peer.js clients.

WebRTC, an open-source project, empowers web browsers and mobile applications with real-time communication capabilities. By furnishing a standardized API, WebRTC enables seamless audio and video communication directly within web browsers, obviating the necessity for external plugins or native applications. Additionally, WebRTC facilitates direct peer-to-peer file sharing, eliminating the reliance on server-side hosting for file transfers. Web Torrent leverages WebRTC transport to enable peer-to-peer file sharing via the BitTorrent protocol directly within web browsers, enabling efficient and decentralized content distribution. This technology is employed by various websites to facilitate user-to-user file transfers, enhancing user experience and efficiency.

VII. CONCLUSION

The adage "two heads are better than one" rings true in the realm of software development. When the primary coder encounters a roadblock in the code, having a partner can often lead to swifter resolutions. While some may argue that pair programming elongates project completion times by assigning two programmers to a single task instead of working independently on separate projects, research indicates otherwise. Studies have revealed that programmers collaborating on the same code are only marginally slower, with a mere 15% decrease in efficiency compared to the presumed 50%.

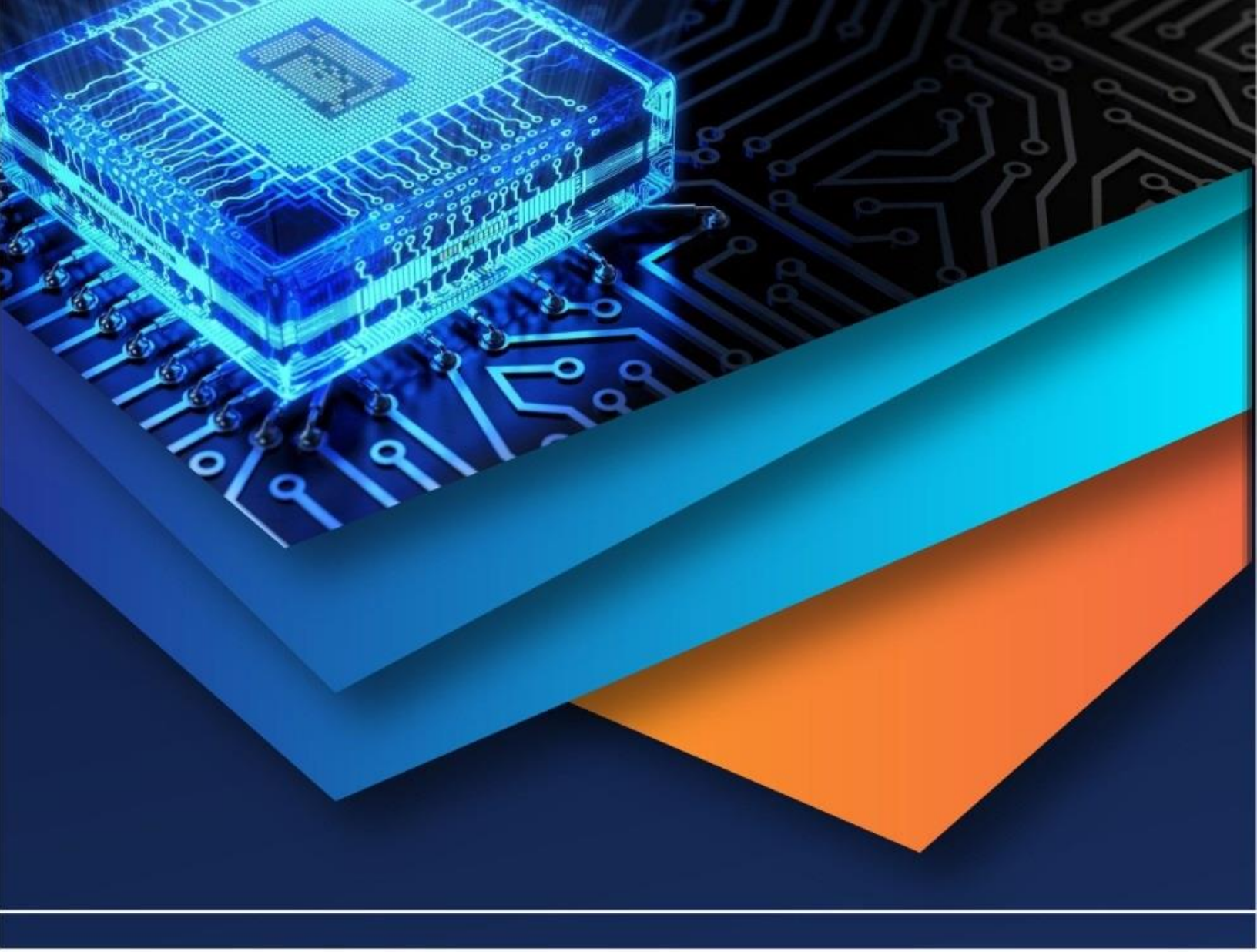
Furthermore, the presence of a second set of eyes results in fewer coding errors, as the partner can provide valuable feedback and catch potential bugs. This collaborative approach allows the driver to maintain focus on the code being written, while the navigator handles external distractions and interruptions.

Pair programming also serves as an effective means of knowledge sharing. By working closely with a peer, programmers can receive immediate, face-to-face instruction, surpassing the efficacy of online tutorials or independent research. This hands-on learning experience allows for rapid skill development, particularly in unfamiliar areas. Additionally, more experienced programmers can impart best practices and advanced techniques to their counterparts, fostering mentorship relationships within the team.

Moreover, pair programming cultivates interpersonal skills among team members. By collaborating on a single project, developers learn the value of effective communication and teamwork, laying the groundwork for stronger collaboration in future endeavours. This collaborative approach not only enhances code quality but also promotes a supportive and cohesive team environment.

REFERENCES

[1] Salomon, G., Distributed Cognitions: Psychological and educational considerations. Learning in doing: Social, cognitive, and computational perspectives, ed. R. Pea and J.S. Brown. 1993, Cambridge: Cambridge University Press.
[2] Constantine, L.L., Constantine on Peopleware. Yourdon Press Computing Series, ed. E. Yourdon. 1995, Englewood Cliffs, NJ: Yourdon Press.
[3] Beck, K., Extreme Programming Explained: Embrace Change. 2000, Reading, Massachusetts: Addison Wesley.
[4] Williams, L., et al., Strengthening the Case for Pair Programming, in IEEE Software. submitted to IEEE Software. Online at http://www.cs.utah.edu/~lwilliam/Papers/ieeeSoftware. PDF
[5] Williams, L.A. and R.R. Kessler. The Collaborative Software Process. in International Conference on Software Engineering 2000. submitted for consideration. Limerick, Ireland. Online at http://www.cs.utah.edu/~lwilliam/Papers/ICSE.pdf
[6] Nose, J.T., The Case for Collaborative Programming, in Communications of the ACM. 1998. p. 105-108. 7. Humphrey, W.S., A Discipline for Software Engineering. SEI Series in Software Engineering, ed. P. Freeman, Musa, John. 1995: Addison Wesley Longman, Inc. 8. Humphrey, W.S., Introduction to the Personal Software Process. 199
[7] Addison-Wesley Flor, N.V. and E.L. Hutchins. Analysing Distributed Cognition in Software Teams: A Case Study of Team Programming During Perfective Software Maintenance. in Empirical Studies of Programmers: Fourth Workshop. 1991: Able Publishing Corporation.
[8] Fagan, M.E., Advances in software inspections to reduce errors in program development. IBM Systems Journal, 1976. 15: p. 182-211.
[9] Johnson, P.M., Reengineering Inspection: The Future of Formal Technical Review, in Communications of the ACM. 1998. p. 49-52.
[10] Lave, J. and E. Wenger, Situated Learning: Legitimate peripheral participation. 1991, New York, NY: Cambridge University Press.
[11] Weinberg, G.M., The Psychology of Computer Programming Silver Anniversary Edition. 1998, New York: Dorset House Publishing. DeMarco, T. and T. Lister, Peopleware. 1977, New York: Dorset House Publishers.
[12] Cockburn, A., Crystal "Clear": A human-powered software development methodology for small teams, Addison-Wesley, 2001, in preparation. Online at http://members.aol.com/humansandt/crystal/clear. 16. Highsmith, J., Adaptive Software Development, Dorset House, 1999.
[13] Cockburn, A., Characterizing People as Non-Linear, First-Order Components in Software Development, in International Conference on Software Engineering 2000. submitted for consideration. Limerick, Ireland. Online as Humans and Technology Technical Report, TR 99.05, http://members.aol.com/humansandt/ papers/ nonlinear/nonlinear.htm.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY