WIP: Streamer and Viewer Interactions in Software and Game Development Live Streams

Ella Kokinda, Clemson University

Ella Kokinda is a PhD student at Clemson University's Zucker Family Graduate Center in Charleston, South Carolina. Her research surrounds live streaming, software and game development, and developer communities.

Dr. D. Matthew Boyer, Clemson University

Dr. Boyer is a Research Associate Professor in the Department of Engineering & Science Education in the College of Engineering, Computing, and Applied Sciences at Clemson University. His work focuses on how technology supports knowledge building and transfer in a range of learning environments.

WIP: Streamer and Viewer Interactions in Software and Game Development Live Streams

Introduction

Prior work has shown that developers who choose to live stream do so as a way of holding themselves accountable to work, continuing their education, finding a community in which they can build and participate, and having visibility and feedback on their work. In this paper, we aim to investigate the types of interactions between streamers and their viewers, the types of knowledge that are transferred in these interactions, how they lead to informal learning opportunities within these streams, and whether these opportunities affect real-time problem-solving for streamers. Additionally, we aim to identify the different types of interactions within the stream and how they lend themselves to forming an informal learning environment.

Through studying human and social aspects of development, we aim to make several contributions to software engineering education research by:

- observing the types of interactions in development live streams and how they impact a streamers' software development practices
- discussing the benefits of live streaming as a form of knowledge transfer, where knowledge transfer occurs, and who initiates transfer within a live stream

This work aims to better understand the human and social aspects of software development live streaming, knowledge transfer in live streams, and discuss the observed benefits and challenges of live streaming for streamers and viewers. Therefore, we ask the following Research Questions (RQs):

RQ_1: What types of interactions occur during software and game development live streams?

RQ_2: Where is knowledge being transferred during software and game development live streams?

RQ_3: What types of knowledge are being transferred?

RQ_4: How does knowledge transfer affect real-time problem solving for streamers?

We ask RQ_1 to gain a high-level overview of what occurs in software live streams and how streamers and viewers interact with and independently of each other. From there, we ask RQ_2 to understand if and where knowledge might be transferred between participants. Next, we ask RQ_3, which aims to classify the types of knowledge being transferred and if the knowledge is relevant to what happens during the stream, unrelated off-topic knowledge, or general programming

knowledge. Finally, using previous data, we ask RQ_4 to understand if knowledge transfer helps or harms development activities during a live stream.

Background and Related Work

Informal Learning Opportunities

Informal learning opportunities play a crucial part in knowledge sharing and acquisition in undefined and opportunistic ways and are not motivated or bound by specific curriculum [1, 2]. Research suggests that informal learning can be effective when individuals can contextualize and situate their learning experiences into something meaningful and tailored to their expected experiences [3]. Within the workplace, professionals use informal learning for continuing education, seeking help, gathering information, finding support or feedback, collaborating, and gaining further experience for both their career and private lives [4, 5]. However, despite research showing the benefits of informal learning opportunities, many individuals and organizations push for formal education over informal or mixed educational pathways [6].

Informal education and opportunities in STEM help bridge the gap between formal education and real-world experiences and foster continuing education throughout a career and beyond [7, 8]. Specifically within computer science, active learning techniques like "rubber ducking[1]" problems aloud and group problem solving can be effective methods in computer science education [9, 10].

Live Streaming

Online services like Twitch and YouTube Live often feature content geared toward gaming, entertainment, and socialization. However, smaller creative and educational live streams have a place within these online services for artists, makers, musicians, and STEM. In particular, Twitch offers an entire category dedicated to software and game development. Prior research shows that live streaming software and game development provides an outlet and opportunity for self-education, accountability, and perceived skill improvement for developers who stream [11]. These streamers also acknowledge the importance of the community that forms around their stream and within the software and game development streaming category as as whole, as viewers can contribute socially and technically to the streams they choose to participate in [11]. Live streaming has also shown benefits over prerecorded videos that gives streamers a lower barrier to entry to start producing content [12]. For viewers, live streamed content is procedural, giving viewers perspectives from experts or knowledgeable people to follow along with in real-time [12, 13].

Within the medical field, live streamed surgical procedures have been shown to benefit students and viewers through active engagement and immersion that lends itself to medical professionals feeling more confident through this unique procedural learning perspective [14, 13]. Within video game streaming, viewers have the opportunity to participate in a community, find new or continued interests, and find the space to learn through others. Overall, live streaming platforms offer unique opportunities for education and learning [15].

[1] A method of debugging and problem-solving technique where the developer thinks aloud to an inanimate or or abstracted object and explains line by line what they are doing with code; http://lists.ethernal.org/oldarchives/cantlug-0211/msg00174.html

Knowledge Transfer in Software Development

Knowledge transfer in software development is paramount for developers to stay ahead and keep their skills useful and marketable. Knowledge transfer takes many forms, from interpersonal communication, static text documentation like wikis, how-to guides, question and answer forums, prerecorded video lessons and lectures to in-person classes [16, 17, 18]. Prior research shows alternatives to online text documents and documentation that prerecorded software documentation is a valuable way to share knowledge and build a reputation in the developer community [19].

Methodology

Our methodology takes a thematic analysis and content coding approach where the research team has analyzed 26 hours of video content using a video coding software tool to timestamp events within a stream and begin to identify emerging themes related to our research questions [20]. Through the process of re-watching video and timestamps of interactions, the first author has identified 12 codes with sub-codes found in Table 1 as part of the video analysis and deep dive into the actions and interactions on stream. With these codes, we also plan on analyzing the frequency and duration of on- and off-topic conversations and problem-solving duration.

Video Selection and Analysis We identified live streamers by browsing streams on Twitch in the *Software & Game Development* category and YouTube Live using the search and recommendations feature, sampling from streamers with the most viewers to no viewers. Selecting from a varied range of viewership provides a more comprehensive view of live streams and the types of engagement that can occur at all levels of stream popularity. We collected 56.75 hours of live streams from 21 streamers (S1-S21) across Twitch and YouTube, with six streamers presenting as females for 18.5 hours of content and 15 males for 38.25 hours of content. Sixteen streamers were software developers (S2-S4, S7-S11, S13-S16, S18-S21) and 5 were game developers (S1, S5-S6, S12, S17).

Positionality and Limitations This work utilizes qualitative research techniques to analyze software and game development live stream content. As such, it is necessary to acknowledge that the results presented in this paper are based on the experiences and attitudes of the research team. The authors are aware that the demographics of this type of streaming can skew masculine, but took the effort to equalize representation within the sample. We also acknowledge that not all streams are the same; they can depend on the tone set by the streamers through tags or their stream descriptions, as well as the total number of viewers and how frequently they choose to participate within the stream. The research team aimed to gather data from a variety of streamers with large to small audiences to gather a more holistic idea of streamer actions and interactions.

Preliminary Themes and Discussion

RQ_1 **Interaction Types** Several interaction types have appeared in streams: information seeking, expressions of opinion, entertainment, information sharing, and general social conversations. Information sharing interactions on streams trended toward streamers vocalizing what they were doing to their audience, with 9 of 11 streamers actively engaging in think-aloud processes of what they were working on or trying to work through on a technical level. Three of the eleven streams

trended toward being more social, with streamers actively engaging in non-technical questions and topics posed by viewers.

Many viewers offered encouraging sentiments to the streamer, observed in 5 of 11 streams. Interestingly, several streamers chose not to engage with these types of comments from viewers or would go back and look at the chat after working on something for a time. S1, a game developer, would ignore chat but, when frustrated, would scroll through to see if a viewer had said something that might begin to solve what he was working on.

Interactions have been technical, relating to the stream content, technology in general, technical employment, or general encouragement and suggestions from viewers. Some streams had mixed social and technical interactions where streamers might go off-topic due to a chat message or built-in social time.

RQ_2 **Knowledge Transfer** Knowledge transfer in streams occurs most generally through the think-aloud nature of streamers who, at a high level, talk through what they are working on or plan to work on during stream. In most streams, a viewer may pose a question to the streamer seeking information about what they are working on or something entirely different, but still technology related. In S3's stream, a viewer asks why the streamer works on a particular project, S3 responds:

💬 *"I enjoy working on it. It breaks a lot for two reasons. One, Building Python is a big pain in the [butt], there's a lot of stuff that goes into making a correct Python build. Python also changes a lot ... this is just what happens when you work with bleeding edge technologies, you will deal with breakage a lot. ... I [fix] these breakages one at a time, rather than working on a whole month of breaks at one time."*

S3 also monitored chat for these types of questions, readily answering them, and S9 set time aside during the stream for a question-and-answer session. Conversely, some streamers were reluctant to answer technical questions, like S1 and S4, often leaving questions in the air with "it depends" in a way they felt they should not overly influence their viewers. Notably, S2, during a website design stream, shared a collaborative link with a viewer for them to work on the document together while still live. Another streamer made a point to make stream summaries from a previous stream, giving viewers who might have missed something a chance to go back and watch a previous stream if it was something that interested them. Yet still, S9 has a minutes-long monologue on how to approach learning new languages.

For viewers, knowledge transfer often appears in the form of suggestions rather than outright solutions while a streamer works. They are often unprompted by the streamer (meaning they do not ask for suggestions). In other streams, viewers assist other viewers who ask technical questions or about streamers' content, which can partially take a cognitive load off the streamer from having to answer the same questions. S3 had a solution to this with a custom script that viewers could run that would give facts about what he is working on and other frequently asked questions from chat members.

RQ_3 **Technical Knowledge Transfer** Technical knowledge transfer dominates most streams. The various processes of development are the largest piece of knowledge conveyed through a live stream. Viewers are given the chance to see exactly how a software or game developer works,

their thought processes through programming, their environment setup, and their workflow. S6, S7, and S10 made it a point to look at the official documentation for tools they were using as a reference when using them in their code or when they were running into an issue. Another knowledge transfer is the specific tools streamers use in their development. A viewer of S10's stream asked about a specific tool called "Terraform" and S10 responded:

💬 *"I can't speak too much on Terraform, I haven't played too much with it, but I have seen some interesting implementations with it."*

Interestingly, S2 explains on her stream that she conducts herself and her streams in a very specific way to advocate for inclusivity and having a welcoming community. She believes that by setting good examples for herself, her audience will model that behavior elsewhere online or in person. While not explicit like S2, S11 made it a point to emphasize taking breaks and, during streams, taking scheduled breaks and telling her viewers also to take a break if they can.

Problem Solving in Live Streams The amount of knowledge sharing between a streamer and audience leading to problem-solving or forward progress varies between streamers. Many appear to use their audience and stream as a form of *"rubber ducking"*. They think aloud through the problem or task and subsequently arrive with an answer or direction needed. Several streamers appear to ignore chat suggestions while actively problem-solving, only turning to chat when they have exhausted their own options.

Conclusion and Future Work

This paper presents preliminary work aimed at understanding social and informal learning aspects of software development through the lens of live streaming. This work begins situating our understanding of actions and interactions within live streams and enables us to continue to research where informal learning opportunities benefit software and game developers, from novices to experienced professionals. We believe that further research will help formalize the importance of live streaming as a beneficial and low barrier-to-entry means for knowledge transfer, skill development, and community building for developers of all skill levels and abilities. We hope that this work will be received by the education community as an important aspect of continuing education and knowledge transfer in a field with technologies and techniques that change rapidly. We look forward to any feedback and welcome researchers, live streamers, and viewers of live streams looking to collaborate with the remainder of this work.

References

[1] P. Hager and J. Halliday, *Recovering informal learning: Wisdom, judgement and community*, vol. 7. Springer Science & Business Media, 2007.

[2] M. A. Kelly and P. Hager, "Informal learning: relevance and application to health care simulation," *Clinical Simulation in Nursing*, vol. 11, no. 8, pp. 376–382, 2015.

[3] A. Coelho and L. M. Costa, "The integration of augmented reality and the concept of sticker album collection for informal learning in museums," in *Immersive Learning Research*

Code	Subcodes	Definition
social interaction	.technical .non-technical	- How was your day - Did you watch last nights episode of the last of us - Did you see they were updating XYZ API - How do you feel about the changes they made this update
information seeking	.technical (on-topic) .non-technical (off topic)	On-topic conversations about what is going on in the stream - How did you do that - Why are you doing it that way - What if you did it this way
expression of opinions	.on-topic .off-topic	Pieces of conversation expressing options on on- or off-topic things - I think you should do it this way - Doing it that way will not work - I know for a fact that...
entertainment	.technical .non-technical .stream-enhancement	Things that entertain the viewers and streamers - A pop up on screen - Activity where people can interact in chat (like the !catch for the catch Pokemon or !reaction) - Wave of emotes when the audience finds something funny - Donations to make bits fall on the screen - Collaborative activity
information sharing		times where the streamer is giving advice or knowledge sharing to the audience as they work (at a point of being ”an expert”) - “So doing it this way will” - “I really like this tool because” - “Using this algorithm will”
bug/problem	.start .end	noting down how many times a streamer ran into an issue or bug within the code or something was not working the way that they were intending
solution	.streamer .viewer .collab	noting where a solution was found during a stream, who figured out the problem or solution - the streamer, a viewer, a collaboration
streamer to viewer		direction of interaction
viewer to streamer		direction of interaction
viewer to viewer		direction of interaction
streamer to self		direction of interaction, think aloud without expecting an answer, talking to self

Table 1: Codes and sub codes for video content analysis.

Network: Third International Conference, iLRN 2017, Coimbra, Portugal, June 26–29, 2017. Proceedings 3, pp. 107–115, Springer, 2017.

[4] T. J. Conlon, "A review of informal learning literature, theory and implications for practice in developing global professional competence," *Journal of European industrial training*, vol. 28, no. 2/3/4, pp. 283–295, 2004.

[5] E. Schürmann and S. Beausaert, "What are drivers for informal learning?," *European Journal of Training and Development*, vol. 40, no. 3, pp. 130–154, 2016.

[6] A. Manuti, S. Pastore, A. F. Scardigno, M. L. Giancaspro, and D. Morciano, "Formal and informal learning in the workplace: A research review," *International journal of training and development*, vol. 19, no. 1, pp. 1–17, 2015.

[7] K. Sacco, J. H. Falk, and J. Bell, "Informal science education: Lifelong, life-wide, life-deep," *PLoS Biology*, vol. 12, no. 11, p. e1001986, 2014.

[8] K. A. Benjamin and S. McLean, "Change the medium, change the message: creativity is key to battle misinformation," 2022.

[9] W. Zuill and K. Meadows, "Mob programming: A whole team approach," in *Agile 2014 Conference, Orlando, Florida*, vol. 3, 2016.

[10] D. Schweitzer and W. Brown, "Interactive visualization for the active learning classroom," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pp. 208–212, 2007.

[11] E. Kokinda and P. Rodeghero, "Streaming software development: Accountability, community, and learning," *Journal of Systems and Software*, vol. 199, p. 111630, 2023.

[12] Y. Chen, W. S. Lasecki, and T. Dong, "Towards supporting programming education at scale via live streaming," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW3, pp. 1–19, 2021.

[13] A. Gandsas, T. Dorey, and A. Park, "Immersive live streaming of surgery using 360-degree video to head-mounted virtual reality devices: A new paradigm in surgical education," *Surgical Innovation*, p. 15533506231165828, 2023.

[14] M. Abu-Rmaileh, T. Osborn, S. R. Gonzalez, and J. C. Yuen, "The use of live streaming technologies in surgery: a review of the literature," *Annals of Plastic Surgery*, vol. 88, no. 1, pp. 122–127, 2022.

[15] I. Calderon, W. Silva, and E. Feitosa, "Active learning methodologies for teaching programming in undergraduate courses: A systematic mapping study," *Informatics in Education*, 2023.

[16] C. Treude and M.-A. Storey, "Effective communication of software development knowledge through community portals," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 91–101, 2011.

[17] M. Meng, S. Steinhardt, and A. Schubert, "Application programming interface

documentation: What do software developers want?," *Journal of Technical Writing and Communication*, vol. 48, no. 3, pp. 295–330, 2018.

[18] E. Aghajani, C. Nagy, M. Linares-Vásquez, L. Moreno, G. Bavota, M. Lanza, and D. C. Shepherd, "Software documentation: the practitioners' perspective," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 590–601, 2020.

[19] L. MacLeod, M.-A. D. Storey, and A. Bergen, "Code, camera, action: How software developers document and share program knowledge using youtube," *2015 IEEE 23rd International Conference on Program Comprehension*, pp. 104–114, 2015.

[20] M. B. Miles and A. M. Huberman, "Js. qualitative data analysis a methods sourcebook fourth edition. fourth edi," 2020.