International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Towards the Definition of Application Developer Persona and Developer Experience Conceptualization

Abdul Saboor[1], Lukman Ab Rahim[2], Noreen Izza Arshad[3], Kamarul Zaman Abdul Rashid[4], Tat Kin Tan[5], Idris Ismail[6], Ahmad Kamil Bin Mahmood[7]

[1,2]Department of Computer and Information Sciences
High-Performance Cloud Computing Centre
Universiti Teknologi Petronas
32610 Sri Iskandar, Malaysia
abdul_19001745@utp.edu.my , lukmanrahim@utp.edu.my
[3]Positive Computing Research Center
Universiti Teknologi Petronas
32610 Sri Iskandar, Malaysia
noreenizza@utp.edu.my
[4,5]Intel Corporation (Programmable Solutions Group)
c/o Intel Microelectronics (M) Sdn. Bhd., Bayan Lepas Technoplex, Medan Bayan Lepas,
11900 Bayan Lepas, Penang, Malaysia
kamarul.zaman.abdul.rashid@intel.com , tat.kin.tan@intel.com
[6]Department of Electrical Engineering,
Universiti Teknologi Petronas,
32610 Sri Iskandar, Malaysia
idrisim@utp.edu.my
[7]Interventure Tech Enterprise
Laluan Tronoh 9
31750 Tronoh, Malaysia
ahmadkamilmahmood@outlook.com

Abstract— For decades application developers (ADs) encountered numerous challenges in employing hardware acceleration capabilities as they are difficult to abstract and consume with ease, thus ADs are avoiding such capabilities and eventually ignoring them altogether and fully dependent on the operating system and virtualization vendors to abstract and provision those acceleration features. With the advent of microservices, there is a need to research this topic in order to comprehend ADs requirements and expectations. This paper aims to give additional conceptualization of ADs persona description and experience around the inclusion of software frameworks/libraries among established technology strategy leaders and developers. The qualitative research method was used which led to conducting in-depth individual interviews associated with the domain of application development. These in-depth interviews, based on the Unified Theory of Acceptance and Use of Technology (UTAUT) paradigm, investigate strategic leaders' and technical specialists' intentions to accept and use hardware features. The study finally produced a conceptual framework comprising four aspects that describe ADs persona. The framework provided high-level descriptions of how certain properties can be implemented. The conceptual framework can be utilized by new or established ADs to identify specific traits to focus on. The highlighted features will lead to further studies quantifying their future influence.

Keywords- Application Developer Persona; Developer Experience Conceptualization; Developer Obsession; Hardware Acceleration Capabilities; Hardware Feature Accessibility; Microservices

I. INTRODUCTION

With the advancement of information and communication technology, application developers have gained prominence to design and deliver solutions that change lifestyles and improve productivity [1], [2]. However, application developers (ADs) encountered multiple issues in employing existing hardware acceleration features, libraries, and application programming interfaces (API). As increased and rich developer experience (DX) has been linked to increased productivity, it is no surprise that many businesses, technology companies, and developer communities view it as a priority to promote a more DX-oriented work environment that promotes ease of use for their developers.

This study, therefore, aims to get a better understanding of the characteristics that are thought to best represent developer obsession and associated persona definition that would define the new DX centered around the use of hardware acceleration features with ease, while focusing specifically on how and why these characteristics are significant in forming the personas of ADs. The persona concept was initially introduced by Alan Cooper and focused on the utilization of personas, their objectives, and scenarios in design [3]. It has since been

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

emerging as a promising and new paradigm in user needs modeling.

Personas are made up of precise, and tangible depictions of target users [4]. They are designed to seem like actual people, thus they contain information such as names, ages, educational backgrounds, jobs, skills, ambitions, worries, surroundings, system usage habits, and so on [5]. Personas capture rich user behavior models and can assist in gaining a deeper knowledge of the target audience and making better design decisions based on these personas [6]. It is therefore needed to research this topic to comprehend application developer personas. This paper aims to give additional conceptualization of application developer personas description and experience around acceptance of hardware acceleration features among established companies, technology strategy leaders, and developers alike. In this vein, the objective and aims of this project are as follows:

Objective: The objective of this research study is to comprehensively examine the operational functionalities and engagement determinants of ADs concerning their utilization of software-integrated development hardware acceleration features.

Aim: This research study aims to raise understanding of the attributes that are perceived to be characterized as DX namely on how and why these attributes are important in shaping the ADs personas.

II. LITERATURE REVIEW

Previous studies have attempted to determine the elements that may influence the levels of satisfaction, developer obsession, quality of work life, productivity, and motivation experienced by application developers. This section outlines some of the existing research studies performed.

An investigational study highlighted the factors that affect the productivity of software development [7]. The authors conducted an empirical study using the data available at the International Software Benchmarking Standards Group which is a software project repository. The authors concluded that four factors that influence productivity are computer-aided software engineering tool usage, choice of programming language, architectural types, and business areas.

A study highlights that while software developers implement secure practices, they often overlook their usability [8]. Through interviews with software professionals, contextual factors such as stakeholder pressure, expertise availability, collaboration culture, and the implementation of the software development process were found to significantly influence the usability of security features in software products. The study concludes by suggesting the need to study and improve these contextual factors to enhance usable security in software.

The effects of various styles of transformative leadership on the motivation of software developers are also investigated [9]. It makes use of the full-range theory of leadership and zeroes down on the impact that transformational leadership has on the creative actions of software engineers. The findings indicate that charisma, inspirational motivation, intellectual stimulation, and individualized consideration positively contribute to affective organizational commitment, which in turn enhances innovative behaviors among software engineers.

In another research study, the authors analyzed 825 online forum postings posted by developers on a platform designed for the developers' community to share their frustrations and joys [10]. The postings were written in the early months of the pandemic which explains their condition and well-being. Negative comments make up around 49 percent of all threads, while positive ones make up approximately 26 percent. Authors discover evidence that developers have difficulties with working remotely due to a lack of documentation and dealing with loneliness while working from home.

The accumulation of technical debt (TD) can be a barrier to progress in software development and ultimately impact the software developer [11]. According to the findings of the investigation study, software engineers lose an average of 23 percent of their working time owing to TD, and they are regularly required to implement new TD. Additional testing is often the work that consumes most of the additional time that is made available. The findings of the study show proof that TD is detrimental to developers since it results in an excessive waste of working time, and wasted time has a detrimental effect on productivity.

Greiler et al. [12] conducted semi-structured interviews with software developers working in various industries, transcribed those interviews, and then iteratively coded the results to gain a better understanding of the factors that influence the developer experience. The results of the study shed light on the aspects that influence the developer experience as well as the traits that determine the relative significance of those factors to individual developers. The authors proposed a developer experience framework that offered a comprehensive reference that can be utilized by businesses that are interested in fostering more efficient and fruitful working conditions for their software engineers. It is suggested that a three-step method be used to make use of the DX Framework to methodically enhance the development experience. The three-step method was based on i) Ask: making the problem visible and learning about the developer's day-to-day experience; ii) Plan: identifying the specific owners of each area that requires improvement and delegating responsibility for its implementation to the individual, team, and organization levels; iii) Act: developers don't have to rely exclusively on their teams or the direction of their managers to force change; they may utilize their particular tactics to achieve changes instead, this working on continuous small improvements is the key to success.

Another study examined how cultural lag theory might be used in software development to better understand the effects of technological advancements [13]. The Gioia method was used to evaluate interviews with subject matter experts, which revealed key developments in software engineering (Gioia et al., 2020). The findings identified four trends that will significantly alter the software development industry, which are: the rise of scalable solutions, the importance of data, the merging of the IT and non-IT sectors, and the widespread adoption of cloud computing.

For a software engineer to have a productive day, they need to be able to choose their activities, use their preferred tools, and have few interruptions. A survey of professional developers revealed that email had a negligible effect on unproductive days, but meetings and interruptions might be beneficial at specific stages of development [14]. Researchers and managers may boost productivity by encouraging developers to take ownership of their work and suggest changes to procedures and tools.

A study to support developers' productivity built the SPACE framework to capture the many aspects of productivity, since, without it, prevalent and even destructive opinions about productivity may continue to prevail [15]. The dimensions of the SPACE framework were based on Satisfaction, Performance,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

Activity, Communication, and Efficiency. The SPACE framework offered a method for logically and methodically thinking about productivity in a much larger space, as well as a defined method for carefully choosing balanced metrics that are linked to goals. It also well-defined the method for understanding how these metrics may have limitations if used alone or in the wrong context.

Another research study examines the challenges faced by software developers in implementing privacy measures [16]. It draws attention to the fact that privacy isn't always a top priority for developers and that there aren't always sufficient resources available to aid in the creation of apps that respect users' right to anonymity. To address privacy concerns, the paper reviews existing frameworks and tools for developers, finding them to be clumsy, narrow in scope, and inadequate to address privacy issues.

The human factors in software security and their impact on vulnerabilities were examined through an online survey of 123 software developers, and the authors concluded that organizational and process support, rather than individual developers, play a significant role in addressing security issues [17]. The authors emphasized the need for a holistic approach that considers organizational factors when addressing software security.

The existing studies lack the representation of an application developer persona about employing hardware acceleration capabilities. Therefore, it is evident that there is a clear gap in investigating the working processes of application developer persona and the factors that are thought to keep them engaged with software-integrated development environments (IDE), libraries, and frameworks to use the hardware libraries.

III. METHODOLOGY

The qualitative research approach was chosen because it appears to be the best fit for the study's aim and objective. In-depth one-on-one interviews with a predetermined number of respondents who were purposefully chosen depending on how many years of experience they had playing the role of ADs were conducted as part of this study [18]. This was crucial so that the interviews could concentrate on specific traits of interest, which has made it possible to achieve the research objectives, namely the identification and selection of respondents with rich information about the definition of the ADs personas and their experiences with accepting software IDE, libraries, and frameworks. Given are the details of the interview settings and the theory used for the analysis of the conducted interviews.

A. Semi-structured Interviews

Semi-structured interviews with a series of key questions were used to help identify the areas to be investigated and to allow interviewers or interviewees to deviate and follow a concept or response in greater depth. This interview style is most commonly used in the social science disciplines since it gives participants some direction on what to talk about, which many people find useful [19]. This method allows for researchers to acquire in-depth information and evidence from interviewees and its adaptability enables the context-based discovery or enrichment of information, providing a more thorough understanding of what occurred and why.

The interviews took place over the course of a month. The Unified Theory of Acceptance and Use of Technology (UTAUT) model, developed by Venkatesh et al., (2003), was adapted for use in these in-depth interviews by Venkatesh et al., (2012) as a lens to examine the strategy leaders' and developers' intentions to accept and use hardware architecture libraries or its equivalent in their implementations. The respondents' profiles are shown in Table 1. The names of the respondents are kept hidden to maintain confidentiality and anonymity.

TABLE I. PARTICIPANTS DEMOGRAPHICS PROFILES

Respondent ID	Role	Years of Experience	Group
S1	Application Developer (Desktop)	12	Technical Specialist
S2	Platform Software Developer	20	Technical Specialist
S3	Middle-Tier Integration Developer	8	Technical Specialist
S4	Application Developer (Web)	10	Technical Specialist
S5	Technology Strategy Leader	30	Strategic Leader
S6	Application Developer (Desktop)	5	Technical Specialist
S7	Engineering Manager/Director	27	Strategic Leader
S8	Technology Strategy Leader	32	Strategic Leader
S9	Hardware Driver and Middleware Developer	8	Technical Specialist
S10	Hardware Driver and Middleware Developer	24	Technical Specialist
S11	Hardware Driver and Middleware Developer	15	Technical Specialist

B. The Interview Settings

The interviews were planned according to the respondents' availability and convenience, and they were all performed online using MS Team owing to social distancing constraints. The questions were designed in such a manner that they are likely to produce as much information on the study phenomenon as feasible while also addressing the research's aims and objectives. In the interview, technology strategy executives and technical specialists were asked 22 open-ended (i.e., requiring more than a yes/no answer), impartial, sensitive, and intelligible questions, as well as probing questions. It started with questions that responders could readily answer before moving on to more difficult or sensitive topics. This strategy helped to put respondents and interviewers at ease, create confidence and trust, and provide rich data that was later used to refine the interview further. The length of the interviews varied based on the respondents, but on average lasted 70 minutes.

Before the actual data collection, the questions were piloted on a few respondents. Pilot interviews with two experienced ADs were done. This exercise has given the study team confidence that the interview questionnaires are clear, comprehensible, and capable of being answered by respondents.

C. Developing the Interview

Before the interview, respondents were informed about the study's specifics and assured of ethical norms such as anonymity and confidentiality. This offers respondents a sense of what to expect from the interview, enhances the possibility of honesty, and is a critical component of the informed consent process. Establishing mutual understanding before the interview is also

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

vital since it has a beneficial influence on the interview's future progress.

The interviewer acquainted themselves with some of the interview questionnaires before doing the real interview, so that the process seems natural and less rehearsed. To ensure that the interview was as beneficial as possible, the researchers made certain that thorough and representative data were captured throughout the interview with researchers taking notes and video and audio recording as backup. Because the interviews were performed online, open, emotional/neutral body language such as nodding, smiling, and seeming engaged were somewhat limited compared to when the interview was done in person. Nonetheless, the strategic use of silence (the interviewer intentionally remaining silent as a subtle hint to the interviewee to talk more) was employed, which was highly helpful in persuading respondents to think about their comments, explain, or clarify difficulties.

At the end of each interview, the participants were acknowledged for their time and asked if there was anything they would like to add. This allowed the respondents to deal with any issues that they thought were important but had not been dealt with leading to the discovery of new, unanticipated information. Field notes were prepared during and immediately after each interview about any important observations, key thoughts, and ideas as this can help during the data analysis process.

D. UTAUT Theory

UTAUT was used as the theoretical foundation for this study to better understand how and why various software IDEs, libraries, and frameworks are chosen by ADs, as well as their acceptance, resistance, preferences, and common behavior [20]. As a result, the four UTAUT factors have been adopted and used to build the semi-structured interview questions [20]. The effort was to learn more about how and why ADs choose to work with software IDEs, libraries, and frameworks and how these choices are linked to various performance and effort expectations, as well as social influences and enabling circumstances. It is also important to note that the intention was not to find which factors are accepted or rejected. The main intention of using the four factors of UTAUT was to have an initial lens that guides this study to probe how and why the decision is made that could be associated with AD's performance expectancy, effort expectancy, social influences, and any possible facilitating conditions, briefly explained as follows [20]:

1) Performance expectancy: In what ways do the ADs believe that using the software IDE, libraries and frameworks will help him or her attain gains in job performance?

2) Effort expectancy: In what ways does the use of the software IDE, libraries, and frameworks ease the ADs tasks?

3) Social influence: In what ways is the ADs choice of using a software IDE, libraries, and frameworks influenced by others and the surroundings?

4) Facilitating conditions: In what ways do the ADs use of software IDE, libraries, and frameworks motivated or influenced by certain conditions that include organizational and infrastructure?

Using the unified theory of acceptance and use of technology helped with qualitative research focused on the application developer persona and conceptualization of developer experience in the given ways [22], [23]:

1) Theoretical Framework: UTAUT served as a theoretical framework to guide the qualitative research study exploring developer experience. Leveraging UTAUT's constructs related to technology adoption, such as performance expectancy, effort expectancy, social influence, and facilitating conditions, provided insights into how software developers conceptualize and perceive DX.

2) Research Design: It helped in the development of discussion topics and interview questions that align with UTAUT's constructs. This ensured that the study probes into the relevant aspects of DX, including attributes that influence developers' acceptance and use of features, tools, platforms, and APIs.

3) Data Analysis: UTAUT served as a filter through which the qualitative data was categorized and interpreted during the data analysis phase. The codes and themes that emerge from the data can be related to UTAUT's constructs, allowing for a structured analysis of the factors that shape DX conceptualization among developers.

4) Interpretation of Findings: UTAUT's relationships between its constructs were explored and interpreted using the contextual data obtained through qualitative research. This helped to grasp how performance expectancy, effort expectancy, social influence, and facilitating conditions interact and confluence developers' perceptions of DX.

IV. RESULTS AND DISCUSSION

Figure 1 depicts the qualitative data analysis methods employed in this study, beginning with data transcription and the tools used, analysis regions, coding, sorting, and then presenting the findings in terms of themes.

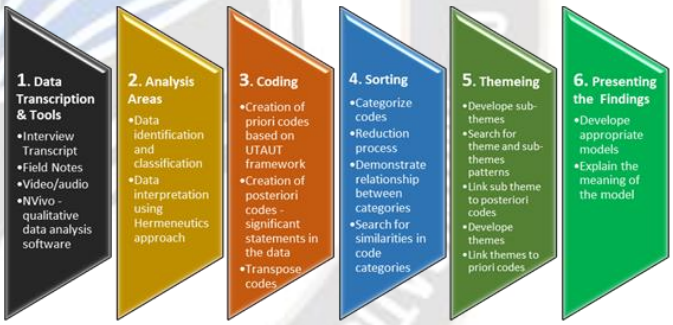

Figure 1. Qualitative data analysis processes.

To begin with data transcription, the interview transcript (transcribed from the video/audio) and field notes were uploaded into the NVivo* program, which was extensively utilized to assure data integrity. The hermeneutic method [24] was employed for the analysis section to emphasize the "sense-making" or comprehension of the ADs persona in context based on the respondents' broad knowledge of software frameworks/libraries. Following that, coding was performed, which is an analytical technique of classifying data by grouping similar material into a container known as a node. All relevant references may be seen and traced inside a node. A node is therefore a code that reflects themes or subjects identified in data. Both deductive and inductive approaches were used. The sorting procedure started after the nodes were constructed. This approach classified, reduced, and depicted links between categories, and looked for commonalities between code categories. Finally, the findings were exhibited via modeling the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

qualitative data in NVivo*, which resulted in a visual representation of what was happening in the data, specifically how various items may be associated. It provided a visual sense that might be related to theoretical models (UTAUT) or just a technique to step back from the data to articulate, illustrate, and explain linkages that are emerging to be recognized.

The coding basis, laid the groundwork for further improving the thematic analysis and working in an emerging and evolving mode for the focused codes, i.e., having some initial codes and then developing the others in an emergent manner. In either case, it provided the opportunity for coding on a broad level. In the Mind Map, analyzers find what the individuals think of their initial codes and transform them into more specific codes. Later, went into each code, reviewed the range of topics covered, and afterward coded more finely within the code. This activity leads to the emergence of the developer experience quadrant and the recommendations for the application developer persona and developer experience conceptualization, as explained below.

A. Developer Experience Quadrant

As explained earlier, the personas were derived using cluster descriptions and interview transcripts. After the personas were determined, the interviews were thoroughly checked to substantiate the identified personas. Finally, the visual representation of the findings emerged as a developer experience quadrant. A developer experience quadrant with tiers is presented in Figure 2. It illustrates the meaning and comprehension of the ADs persona based on their expertise with various frameworks and libraries.

Figure 2. Developer Experience Quadrant

Following is the detailed elaboration of the four identified quadrants for "Application Developer Persona and Developer Experience Conceptualization" from the respondents' point of view.

1) Ease of Use: A technology strategic leader who is also a CEO of a well-reputed organization, infers the ease of use by saying that:

"I truly believe from a developer's perspective it is the 'ease of use' or the 'ease of build' that drives the behavior. No matter whom you talk to eventually you will realize that conclusion that it is the ease of build. And in the industry, if something becomes too difficult to do from a coding standpoint, either there is a new language that comes out that's fairly easy or someone builds a tool that will build the code for them" [S8].

The same is projected by a technical specialist:

"You know, one of the most important things is how easy it is to develop in the framework, how easy to test, and how easy to debug. All of these criteria that are mentioned before are important so that our developers will focus on their main problem and not on the ecosystem of how do I monitor, how do we build, and how do we deploy, and all those other areas. We want them to focus on the problem" [S10].

According to the developed coding schema, the ease of use is based on:

- API Hooks on Soft/Hard IP Features - *"I mean the coding we write, we put into the server and let others call. The other implement functions and then they will be charged on a per-request basis. So, they are more focused on the service, the requests, I mean, to provide the service, this and that for the whole server. They will provide until the very small parts like functions, like the applications service, then the performance, the server, then I mean, the things that will be getting big." [S6]*
  "For example, when we leave the integration on the application, A to Z, you just have to subscribe to the service and then we do our coding there, and the coding there becomes easy when you just have to select the edit tool to manage the application that we adapted than we do mapping in the invitation group, that you'll be working" [S1]
  "I like the idea of kind of creating those high-level libraries that expose features as part of your product thinking and your product rollout. Figuring out how to go up the stack and creating things that are easier to click into at a high level." [S5]
- SECO Agnostic - *"And then the other is just ecosystem So, as you know, there is a whole Node JS ecosystem, right, and then there is you know whole .net ecosystem, and then there's sort of you know the whole Python ecosystem. And organizations really, you know, that they align with one of those ecosystems" [S5]*
- Instant Reward - *"The newer generation that comes out, they are looking for an instant reward, you know, immediate gratification" [S8]*
- Open-source - *"We tend to focus more on the open source. So, what we consider is, if there is a free license for you to use then we will prioritize that one." [S4]*
  "For an open source, if there's a bug that somebody found out there, they will just fix it, and then they will just put it up there so that people can actually run the working one." [S11]
- Easy Debugging and Instrumentation - *"…ability of developers to quickly understand, quickly use. And, you know, instrument and debug and all those things is a major driver" [S5]*
  "…the debugging capabilities that are available, how easy it will be to debug the application in the production environment." [S10]

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

2) Ecosystem: The ecosystem is described as an environment of space of work where future evolution, maturity, and extension of the strategy happens. It is a collaborative innovation (co-innovation) approach by developers, software organizations, and third parties that share a common interest in the development of software technology.

To shed light on the ecosystem, a technology strategic leader highlighted that:

"Another thing to think about is the ecosystem too. It is like you have to make these ecosystem plays or become one yourself. And there is a low-level hardware ecosystem. And you know, growing up the stack so that you are a 'hardware-software ecosystem' is probably your only defense around the software companies that are growing down the stack to build their own silicon" [S8].

Another technology strategic leader highlighted that:

"You have to be, you know, you have to be part of those, you know, those dominant ecosystems" [S5].

To build a firm ecosystem, the coded data featured the following key areas:

- Developer Community - *"Community forums are very often where people get whatever information that is not documented" [S11]*
- *"I do a deep survey to see if the community is big or not. So, if the community is huge enough, I will pick that one. I would prefer a community that is common in the market" [S4]*
- Documentation - *"Let us say we are getting some library from the community, so usually I will look through the document first before I decide whether I want to go in or not. Usually, I spend around one week reviewing that document before I set a goal for my team to go in and use that library" [S2]*
- Emerging Technology - *"I always do that, regardless of where I am. I am always looking at the competency landscape. I'm looking at emerging technologies" [S8]*
- Integration - *"It's the entire ecosystem that we have on top of feed aggression, integration, and let the amount of you know custom code libraries that we created to share between the different applications that we have" [S10]*
  "Compared to the previous implementation, you will have to write Java code, or SQL to do the integration. Now, with the technology change, it becomes very easy to build and easy to learn" [S1]
- Loyalty - *"...where you are much focused on one architecture and then you enable and you support that, you know, you can provide more comprehensive support. So, I would say, in that sense that may be better proprietary, and you guys know it well, and you can build loyalty and all that" [S7]*
  "Some companies are loyal to a certain environment probably because it helps them to develop their product fast" [S11]
- Support - *"If we can have a software that you know, you can support backward compatibility, you support future enhancements, future expansion, that would very much be easy for them to just port over their existing software on whatever hardware platform" [S11]*
  "Provide different types of support, either if you have a server issue or some other issue. They have multiple teams, sometimes it's a server issue and sometimes it's a code issue. Sometimes if you have a coding issue, you can ask for support too." [S3]

3) Framework: The framework deals with the delivery of mature and stable software interfaces that provide seamless access to Silicone/hardware features. As quoted by [S6]

"Maybe why we choose the framework is that we like the flexibility that people can adapt very fast, very flexible. Then they can join the project very fast".

A technical specialist quoted:

"So, engineering programming and business programming are two. I see that as mainly two drastic worlds. So, in business programming, I do not care about what is the underlying hardware. I know that frameworks like .Net or even Java can do the so-called abstraction or isolation. So, they will do the optimization at the intermediate layer. But in engineering, this is different because when we code right, we already have a mindset of what kind of architecture we want to target" [S2].

According to the coded data of this study, the framework choice and features are based on:

- CASE Tools Alike - *"Computer Aided Software Engineering 'CASE' tools. Some application platform providers did a phenomenal job of creating a set of tools that wrote applications. It was able to take your data sets, take your painted screens, tie them all together, generate an application from scratch, write all the code needed, and deploy it. So, my suggestion would be if one can create something like that, not using the yesteryear tooling and technologies, but can counter the case tool, that our local tool that's able to harness the power of your hardware, and generate applications, using the latest coding JavaScript, React, Native or any of the new frameworks that you see." [S8]*
- Coding Beyond Syntax - *"To me, I feel that that kind of visualization is very good because you don't have to read through code set to understand what or which function, where do I call, and when do I call it?" [S11]*
  "I think it will lead the companies into that one. Try to create something that everyone can use. Like adapters, or all the tools, and hardware that everyone can use within a few clicks. That is the most important thing. So, customers can use it without having any deep technical experience" [S3]
- Partnership - *"I need to follow the company's commercial directions and partnership concerns. In the current situation, so many limitations for us to choose our framework or need to take care of other things. We need to get approval from the companies as well." [S6]*
- Quick Learning Curve - *"...when we decide to choose how fast the ramp-up of a new developer, how easy it will be for him to start to develop on this new platform?" [S10]*
  "Maybe why we choose the framework is like the flexibility that people can adapt very fast, very flexible. Then we can join the project very fast." [S6]
- Hardware Access for Special Purposes - *"However, if there will be a huge motivation for the developer to understand and benefit from the low level and from the hardware feature, then there is a reason to use it, but there should be a very good reason" [S10]*
- Brand Perception - *"I really choose based on how easy for us to use, because time is very important. You can't lose time. So, if even if you buy from a big brand, and*

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

they can give you 10 people to support, but you still cannot get things working within a certain time, then it is still of no use." [S11]
"I feel that if you are a big brand, it's not necessary that you can develop good software. There are some others, I mean some startups that can develop better software than those big brands, and so on. So, for me, I don't choose based on brand." [S11]

- Si Microservices Catalogue - *"Hardware itself is not sustainable. And there's a need to add all the software and content, for example, diversifying from what used to be a pure-play hardware company to a broader, you know, so many different areas that they're getting into." [S8]*

"So, by handling that part they will provide a service that they will act as the hardware part and provide a service to let others use minor things like you can create libraries and put into the local repository and then let others call and implement it" [S6].

4) Quality and Maturity: Quality defines the practices and behaviors that are desirable to meet the developer's expectations. One of the technical specialists addressed quality as:

"…you know, at the end of the day, the quality of life, the overall quality that will be achieved by developing it, the debugging capabilities that are available, how easy it will be to debug the application in the production environment. This is probably the main criteria that will help us to choose which framework to use" [S10].

Another technical specialist said:

"Loyalty will eventually come with the quality" [S9].

The quality is focused on enriching the experience of the target end-user with measurable quality protocols. Given the developed coding schema, the quality protocols fall upon as follows: -

- Performance - *"What is the overhead of using this platform, for example, from a communication perspective, and things like that? Do we have some hitting in performance, and you know, it can be the memory size, the footprint that is required just launching the framework, and the CPU consumption? These are the factors that we're looking at." [S10]*
- Portability - *"…we can have software that supports backward compatibility, support future enhancements, future expansion. That would make it very much easy to just port over their existing software on whatever hardware platform" [S11]*
- Interoperability - *"Because of the upgrade version, we keep on having some problems with this kind of solution. Because the solution always sticks to the old version and when it becomes the new version, I cannot use it anymore. So, every time I need to strive, struggle." [S4]*
- Reliability - *"We need to make sure the new library works. We do not know what bug will happen eventually along the line. So, we always find a stable one, most supported by the footprint of how many users are using that library" [S9]*
- Security - *"I chose the framework because there are a lot of things that are already ready. So, I do not need to worry about security. The security I apply is from the framework" [S4]*
- Auditable/Origin Traceable - *"Once we need to be working as a team, we need to know control version, which means once you have done some work, how the people, how your team member knows what are the things that you have done, and then they can get what's the latest coding from your side or another team side." [S4]*

B. Recommendation

Following the theoretical background of the UTAUT model, this research has conducted interviews to explore the thoughts and vast experience of the strategy leaders and AD's intention to accept and use hardware architecture features or their equivalent in their implementations. The theory drives this work in exploring ways ADs function and raising the understanding of what makes them engaged with software IDE, libraries, and frameworks. The four factors of UTAUT including performance expectancy, effort expectancy, social influence, and facilitating conditions are investigated through interview questionnaires. The collected interview data provide evidence and insights that characterized ADs personas. The data provide narratives of what is considered important in shaping ADs obsessions. Based on this evidence, four important recommendations are made:

1) Building Ease of Things: The Ease-of-Things (EoT) refers to the convergence of developer experience. ADs are looking for software IDEs, libraries, and frameworks that prioritize EASE-OF-USE. According to the interviews, ADs have requested a mature ecosystem, which is defined as the proliferation of high-productivity, high-performance software across all spectrums to enrich and enhance their experiences. As a result, this study concluded that it is time to develop EoT using the following strategies:

- Developing a broad strategy centered on developer experience and a software-first approach.
- Putting the developer first in all product, tool, and service decisions.
- Every Soft or Hard IPs on silicone must include the API hook that can be exposed to the software interface when needed.
- Software Ecosystem (SECO) agnostic, whereby the design blocks of the microservices can be consumed or worked on across multiple SECOs such as Android, dotNet, etc.

2) Building an ecosystem that incorporates collaborative innovation: It has been noted that social factors play a very crucial role. In this arena, it was discovered that ADs obsessions are formed to some part by social factors such as other users and groups' activities. This is significant since ADs thrive on information exchange for the most recent best practices. As a result, moving ahead, plans should include collaborative innovation initiatives for future evolution and maturity, such as:

- Creating a work environment that fosters co-innovation approaches among developers, software companies, and third parties with a mutual interest in the advancement of software technology.
- Develop strategies to eliminate the technological and operational hurdles that stand in the way of ecosystem monetization.
- Accelerating value realization and customer success through regular interactions, education, and sharing of expertise with essential and potential stakeholders

3) Creating stable and mature software interfaces (microservices) to seamlessly connect to hardware features: It has been shown that both external and internal factors affect ADs

usage of technology. These requirements, which are frequently entangled between technological, infrastructural, and other characteristics, make it easier to choose a certain software IDE, libraries, and frameworks. This motivates ADs to use the tools at their disposal, which aided implementation in a variety of ways, including having:

- Tools assistance, and technologies that provide mature and robust software interfaces/APIs, as well as hardware microservices that give smooth access to silicone/hardware features.
- A drag-and-drop programming IDE, which is a visual interface that allows ADs to program by dragging components/constructs with minimum syntactic knowledge, might drastically reduce implementation time.
- Available silicone features in the form of consumable features or an interface for ADs consumptions for quick and easy implementations.

4) Performance augmentation and strengthening framework for high productivity, secure, and high-performance AD-driven experience: According to the data gathered, the concept of AD performance expectations appears to be relevant. ADs are motivated by gaining enriched expertise during implementation that might improve software quality such as security and performance efficiency. ADs sought engaging experiences to boost their productivity, which included quantifiable factors like compilation time and coding help, among other things. This has prompted the following recommendations for designing software IDEs, libraries, and frameworks that allow for:

- Interoperability, which allows codes and output from multiple development platforms to communicate information and call functionalities.
- Auditability and traceability which enable created codes to be completely auditable and traceable back to their source.

V. CONCLUSION AND FUTURE WORK

This research employs a cross-sectional approach to capture a snapshot of what is happening at one point in time, supported by empirical evidence from strategy leaders and technical specialists describing their past events and experiences. This empirical evidence and understanding derived from the participants are then used to discover insights on what is perceived as "ADs obsession" that drives them to be engaged in software IDE, libraries, and frameworks. This study provided an enhanced conceptualization of application developer persona descriptions and experiences related to the use of hardware libraries and frameworks. Through the use of qualitative research methodology, extensive one-on-one interviews of persons related to the field of application development were conducted.

The findings of this research provide new dimensions to the field of study in the form of a conceptual framework consisting of four quadrants that describe the ADs personas. The framework provides high-level examples of how these characteristics might theoretically be converted into actions to enhance the application developer experience. This study has implications for both practice and research in the sense that the conceptual framework may be utilized by new or existing ADs to highlight the traits they should focus on. The highlighted features will also result in a further investigation, including the quantification of their influence on potential outcomes.

ACKNOWLEDGMENT

The authors acknowledge the support provided by the Intel Corporation (Programmable Solutions Group). The authors would like to express their gratitude and appreciation to Universiti Teknologi PETRONAS (UTP), Perak, Malaysia, for providing the resources and computing environment.

REFERENCES

[1] M. Audi, A. Ali, and R. Al-Masri, "Determinants of Advancement in Information Communication Technologies and its Prospect under the role of Aggregate and Disaggregate Globalization," Sci. Ann. Econ. Bus., 2022.

[2] M. Audi and A. Ali, "The advancement in Information and Communication Technologies (ICT) and economic development: a panel analysis," 2019.

[3] A. Cooper, "The inmates are running the asylum. Indianapolis, IA: SAMS," Macmillan, 1999.

[4] P. D. Marshall, C. Moore, and K. Barbour, Persona Studies: An Introduction. John Wiley & Sons, 2019.

[5] J. Salminen, K. Guan, S.-G. Jung, S. A. Chowdhury, and B. J. Jansen, "A Literature Review of Quantitative Persona Creation," in Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, in CHI '20. New York, NY, USA: Association for Computing Machinery, Apr. 2020, pp. 1–14. doi: 10.1145/3313831.3376502.

[6] B. Jansen, S.-G. Jung, L. Nielsen, K. W. Guan, and J. Salminen, "How to Create Personas: Three Persona Creation Methodologies with Implications for Practical Employment," Pac. Asia J. Assoc. Inf. Syst., vol. 14, no. 3, Mar. 2022, doi: 10.17705/1pais.14301.

[7] L. Lavazza, S. Morasca, and D. Tosi, "An empirical study on the factors affecting software development productivity," E-Inform. Softw. Eng. J., vol. 12, no. 1, pp. 27–49, 2018.

[8] M. Gutfleisch, J. H. Klemmer, N. Busch, Y. Acar, M. A. Sasse, and S. Fahl, "How Does Usable Security (Not) End Up in Software Products? Results From a Qualitative Interview Study," in 2022 IEEE Symposium on Security and Privacy (SP), May 2022, pp. 893–910. doi: 10.1109/SP46214.2022.9833756.

[9] Y. Choi, "How does Leadership Motivate the Innovative Behaviors of Software Developers?," in Research Anthology on Human Resource Practices for the Modern Workforce, IGI Global, 2022, pp. 1727–1742. doi: 10.4018/978-1-6684-3873-2.ch087.

[10] G. Uddin, O. Alam, and A. Serebrenik, "A qualitative study of developers' discussions of their problems and joys during the early COVID-19 months," Empir. Softw. Eng., vol. 27, no. 5, pp. 1–52, 2022.

[11] T. Besker, A. Martini, and J. Bosch, "Software developer productivity loss due to technical debt—A replication and extension study examining developers' development work," J. Syst. Softw., vol. 156, pp. 41–61, 2019.

[12] M. Greiler, M.-A. Storey, and A. Noda, "An Actionable Framework for Understanding and Improving Developer Experience," IEEE Trans. Softw. Eng., 2022.

[13] S. Laato, M. Mäntymäki, A. K. M. N. Islam, S. Hyrynsalmi, and T. Birkstedt, "Trends and Trajectories in the Software Industry: implications for the future of work," Inf. Syst. Front., vol. 25, no. 2, pp. 929–944, Apr. 2023, doi: 10.1007/s10796-022-10267-4.

[14] A. N. Meyer, E. T. Barr, C. Bird, and T. Zimmermann, "Today Was a Good Day: The Daily Life of Software Developers," IEEE Trans. Softw. Eng., vol. 47, no. 5, pp. 863–880, May 2021, doi: 10.1109/TSE.2019.2904957.

[15] N. Forsgren, M.-A. Storey, C. Maddila, T. Zimmermann, B. Houck, and J. Butler, "The SPACE of Developer Productivity: There's more to it than you think.," Queue, vol. 19, no. 1, pp. 20–48, 2021.

[16] P. Kühtreiber, V. Pak, and D. Reinhardt, "A survey on solutions to support developers in privacy-preserving IoT development," Pervasive Mob. Comput., vol. 85, p. 101656, Sep. 2022, doi: 10.1016/j.pmcj.2022.101656.

[17] H. Assal and S. Chiasson, "'Think secure from the beginning': A Survey with Software Developers," in Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, in CHI '19. New York, NY, USA: Association for Computing Machinery, May 2019, pp. 1–13. doi: 10.1145/3290605.3300519.

[18] M. Q. Patton, Qualitative research & evaluation methods: Integrating theory and practice. Sage publications, 2014.

[19] R. Ruslin, S. Mashuri, M. S. A. Rasak, F. Alhabsyi, and H. Syam, "Semi-structured Interview: A Methodological Reflection on the Development of a Qualitative Research Instrument in Educational Studies," IOSR J. Res. Method Educ. IOSR-JRME, vol. 12, no. 1, Art. no. 1, 2022.

[20] V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis, "User acceptance of information technology: Toward a unified view," MIS Q., pp. 425–478, 2003.

[21] V. Venkatesh, J. Y. Thong, and X. Xu, "Consumer acceptance and use of information technology: extending the unified theory of acceptance and use of technology," MIS Q., pp. 157–178, 2012.

[22] Ö. F. Ursavaş, "Unified Theory of Acceptance and Use of Technology Model (UTAUT)," in Conducting Technology Acceptance Research in Education : Theory, Models, Implementation, and Analysis, Ö. F. Ursavaş, Ed., in Springer Texts in Education. , Cham: Springer International Publishing, 2022, pp. 111–133. doi: 10.1007/978-3-031-10846-4_6.

[23] M. D. Williams, N. P. Rana, and Y. K. Dwivedi, "The unified theory of acceptance and use of technology (UTAUT): a literature review," J. Enterp. Inf. Manag., vol. 28, no. 3, pp. 443–488, Jan. 2015, doi: 10.1108/JEIM-09-2014-0088.

[24] B. Kutsyuruba and S. McWatters, "Hermeneutics," in Varieties of Qualitative Research Methods: Selected Contextual Perspectives, J. M. Okoko, S. Tunison, and K. D. Walker, Eds., in Springer Texts in Education. , Cham: Springer International Publishing, 2023, pp. 217–223. doi: 10.1007/978-3-031-04394-9_35.