Advances in Consumer Research
https://acr-journal.com/
Volume-3 | Issue-1 | Jan: 2026
Original Researcher Article

Design And Development Of An Integrated Hardware And Software System

Zhang Yachao* [1], Oyyappan Duraipandi[2]

[1]Lincoln University College, Petaling jaya Selangor Malaysia)

[2]Lincoln University College, Petaling jaya Selangor Malaysia
Corresponding Author
Zhang Yachao
Lincoln University College, Petaling jaya Selangor Malaysia

ABSTRACT
The main emphasis of research on integrated software and hardware systems has been on design and development methods. The study's goal was to create a single framework. One of the key aims of the project was to figure out how to plan development work while keeping in mind the restrictions of hardware and software. This was very important since there has been a clear rise in the number of initiatives that are seeking to connect a system's digital logic to its physical parts. A close look into co-design methods showed that working together early on between the software and hardware teams had a big effect on the system's efficiency, scalability, performance, and performance. The goal of this assessment was to show how big of an effect this partnership had. This research looks at various successful methods that can make it possible to build things in parallel while also lowering the risks of integration. These include iterative prototyping, hardware/software co-simulation, and model-based design. Since they were possible and somewhat effective, they were the ways that were being looked at. Researchers looked at development tools and platforms like as hardware description languages (HDLs), real-time operating systems (RTOS), and FPGA prototype environments to see whether they might speed up development and make system validation easier. Researchers discovered that these platforms and technologies made development a lot easier. Embedded systems, automotive control units, and Internet of Things apps were looked at as real-world examples to show how integrated development methods may be useful in real life. The point of doing this was to show how development plans may be beneficial.
Keywords Hardware Description Languages, Real-Time Operating Systems, Scalability, Software Capacities, Prototype Environments

1. INTRODUCTION:

The rapid pace of technological advancement, along with the ever-increasing need for intelligent, high-performance systems, has lately necessitated a more integrated strategy between software and hardware. Complexity, demand misalignment, and very lengthy development cycles were some of the well-known drawbacks of traditional development methodologies, which treated software and hardware as separate entities. The authors were able to get beyond these limitations by studying how to build a software and hardware system that works together, paying close attention to details like co-design and synchronised development cycles. In order to increase system performance, optimise resource use, and reduce product launch times, the research emphasised the need of early collaboration between software and hardware developers. It combined pragmatic design methodologies used to study the integration process, which allowed it to execute iterative testing and real-time performance analysis. This was made possible by the combined use of these tactics. As part of this plan, we used approaches including model-based development, hardware/software co-simulation, and prototyping. Embedded systems, vehicle control systems, and Internet of Things devices were among the several industrial applications that the research sought to assess in terms of how integrated approaches improved their resilience and flexibility. Issues that arose during integration were also addressed in the research. System design was examined in a context of commonly used industrial frameworks and tools, such as RTOS, HDLs, FPGA-based platforms, and unified development environments. In order to boost innovation, reliability, and production in modern technological solutions, this research primarily aimed to promote a uniform workflow. To do this, we analysed these aspects and incorporated them into what is currently known about the best practices for integrated system development.

1. BACKGROUND OF THE STUDY

There has been an increasing need for smart, high-performance systems, and technical breakthroughs are happening at an incredible pace. Because of this, it has become important to combine software and hardware more closely. It was recognised that traditional development approaches, which saw hardware and software as two separate areas, had a number of bad outcomes, such as making things more complicated, not meeting demand, and taking a very long time to build. This study

https://acr-journal.com/
These problems by looking at how to create a system that combines hardware and software with a focus on synchronised development cycles and co-design (Przybylla & Grillenberger, 2021).

The research underlined the need for software and hardware engineers to work together early on to make the system perform better, make better use of resources, and speed up the time it takes to bring a product to market. By using practical design methods to look into the integration process, it was possible to execute iterative testing and real-time performance analysis. Using these tactics together made this possible. Some of the methods used in this approach were prototyping, hardware/software co-simulation, and model-based development. The goal of the research was to find out how integrated approaches made a wide range of industrial applications, such as embedded systems, car control systems, and Internet of Things devices, more resilient and adaptable. The research also looked at the problems that came up throughout the integration process. One of the numerous problems is sustaining an environment that can grow and stay in sync, as well as managing how hardware and software interact with one other. System design was looked at in connection to common industrial frameworks and tools such as unified development environments, real-time operating systems (RTOS), hardware description languages (HDLs), FPGA-based platforms, and HDLs. The major purpose of this research was to promote a consistent workflow that would boost creativity, reliability, and production in modern tech solutions. This was done by looking at these aspects and adding them to the body of information that is already being used to find the best ways to construct integrated systems (Ali & Hussain, 202-tra 2023).

2. PURPOSE OF THE RESEARCH

The goal of this study is to look at how the procedures of system design and development affect the performance of the hardware and software components that make up the system when they are all working together. The main goal of the study is to look at how design influences how hardware works and how development processes affect how software works. The goal of this research is to provide information that can help with better planning, building, and combining hardware and software systems. By focused on these two interactions, research able to get these insights. The findings are aimed to help people make better decisions while working on engineering projects that include embedded systems, smart technologies, and real-time applications.

3. LITERATURE REVIEW

As the complexity of embedded and intelligent systems grows, more and more literature on integrated software and hardware system development is talking about co-design methodologies as a way to handle it. It became quite clear that this knowledge existed as I looked at the issue more deeply. Using the sequential development strategy, which has been questioned in earlier research, software and hardware were created at different times and without each other. Following this plan led to conflicts, which then led to the creation of conflicting specifications, delays, and higher costs. Researchers came up with the idea of co-designing hardware and software as a way to solve the issue. This plan is iterative and uses a collaborative approach. With this technique, it is feasible to make both parts at the same time, which allows for design optimisation and feedback throughout the manufacturing process (Chen & Liu, 2022). The research found that this approach improved performance, shortened the time it took to integrate, and made the system more reliable. The study discovered that the most interesting examples of this were in the healthcare, IoT, and auto industries. The study literature has looked at the usage of system-level modelling tools like MATLAB/Simulink and System C. These technologies make it feasible to test and model integrated systems early on. Researchers found that FPGA-based system-on-chip (SoC) platforms and solutions make it feasible to quickly prototype both hardware and software (Kumar & Gupta, 2024).
Software development kits (SDKs), real-time operating systems (RTOS), and cross-compilation environments are all things that might make it easier to deploy and debug embedded applications. Still, there were problems that needed to be addressed, such the need for consistent interfaces, the need to sync the development cycle, version control, and resource sharing. The goal of this post was to look at the real-world engineering methods, tools, and models that are meant to make it easier to integrate and build software and hardware systems together in a way that works well. Because this effort was mostly based on existing literature, the investigation had a good foundation. (Patel and Singh, 2023).

4. RESEARCH QUESTIONS

5.1 How does design affect hardware performance?

5.2 How does development impact software functionality?

5. RESEARCH METHODOLOGY

a. Research Design

For both quantitative and qualitative analyses, we relied on SPSS version 25. We utilised the odds ratio and the 95% confidence interval to find out how strong and which way the statistical association was going. The researchers established a p-value of less than 0.05 as a threshold for statistical significance. A descriptive analysis was performed in order to obtain the most important features from the data. It is common practice to use a combination of quantitative and qualitative approaches when analysing data obtained from polls, questionnaires, and surveys, as well as data cleaned up using computing tools for statistical analysis.

b. Sampling

A total of 1280 surveys were received after 1350 were sent out; 80 were excluded because they were incomplete. The Rao-soft program was used to estimate a sample size of 1123. Researchers in China contacted 1,200 people and surveyed them for the study. A total of 624 females and 576 guys completed out the 1200 questionnaires and interviews.

c. **Data and Measurement**

A questionnaire survey was the primary method of data collection in the study. The poll began with some basic demographic questions, and then moved on to a series of online and offline channel ratings using a 5-point Likert scale. The secondary data was mostly sourced from various internet resources.

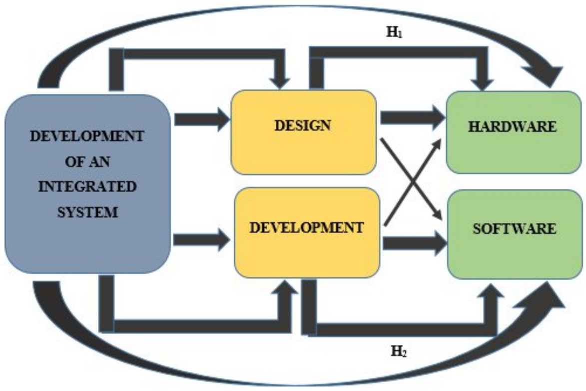
d. **Statistical Software**

The statistical analysis was conducted using SPSS 25 and MS-Excel.

e. **Statistical Tools**

We used descriptive analysis to understand the data on a basic level. The researcher must use ANOVA to analyse the data.

6. CONCEPTUAL FRAMEWORK



7. RESULT

- **DEPENDEENT VARIABLE: HARDWARE**

- **Factor Analysis**

One typical use of Factor Analysis (FA) is to confirm the component structure of a set of measurement items. Most people think that there are hidden elements that affect the scores of the visible variables. The FA method is one example of a model-driven approach. Finding connections between observable events, their causes, and measurement errors is the primary motivation for this research.

It is possible to assess if the data is suitable for factor analysis by using the Kaiser-Meyer-Olkin (KMO) Method. The sample's adequacy is assessed for each individual model variable and for the model as a whole. The statistical measures quantify the inter-variable correlation. When doing factor analysis, it is usually best to work with data that has lower percentages.

KMO yields integers ranging from zero to one. Sampling is considered sufficient if the KMO value is between 0.8 and 1.

Remedial action is required if the KMO is below 0.6, indicating insufficient sampling. Exercise optimal judgement; some writers utilise 0.5 for this purpose, thereby establishing a range of 0.5 to 0.6.

A KMO value around 0 indicates that the partial correlations are substantial relative to the overall correlations. Component analysis is significantly impeded by substantial correlations, to reiterate.

The acceptance thresholds established by Kaiser are as follows:

A bleak range of 0.050 to 0.059.

0.60 - 0.69 subpar

Standard range for a middle grade: 0.70 to 0.79.

A quality point value ranging from 0.80 to 0.89.

The interval from 0.90 to 1.00 is quite impressive.

Table 1: KMO and Bartlett's Test

KMO and Bartlett's Test		
Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.812
Bartlett's Test of Sphericity	Approx. Chi-Square	3252.968
	df	190
	Sig.	.000

This proves that the assertions made about the sample's execution are accurate. To determine if the correlation matrices were statistically significant, the researchers used Bartlett's Test of Sphericity. When the result hits 0.930, the KMO measure deems the sample satisfactory. Thanks to Bartlett's sphericity test, we have a p-value of 0.00. The correlation matrix differs from an identity matrix, as shown by the statistically significant findings of Bartlett's sphericity test.

- **DEPENDENT VARIABLE: SOFTWARE**

- **Factor Analysis**

One typical use of Factor Analysis (FA) is to confirm the component structure of a set of measurement items. Most people think that there are hidden elements that affect the scores of the visible variables. The FA method is one example of a model-driven approach. Finding connections between observable events, their causes, and measurement errors is the primary motivation for this research.

Whether you want to know whether your data is good for factor analysis, you may use the Kaiser-Meyer-Olkin (KMO) Method. We check the sample size for each model variable separately and for the overall model to make sure it's sufficient. The statistical metrics assess the correlation between the variables. Data with smaller percentages is often better to deal with when undertaking factor analysis.

Integers between 0 and 1 are produced by KMO. When the KMO value falls anywhere between 0.8 and 1, it is deemed that the sampling was enough.

If the KMO falls below 0.6, it means that there was not enough sampling and corrective action has to be taken. Use your best discretion; 0.5 is often used by authors for this reason, therefore a range of 0.5 to 0.6 is established.

As a percentage of total correlations, partial correlations are large when the KMO value is close to 0. Research says large correlations greatly hinder component analysis.
Here are the acceptance levels set by Kaiser:

A bleak range of 0.050 to 0.059.

0.60 - 0.69 subpar

Standard range for a middle grade: 0.70 to 0.79.

A quality point value ranging from 0.80 to 0.89.

The interval from 0.90 to 1.00 is quite impressive.

Table 2: KMO and Bartlett's Test

KMO and Bartlett's Test		
Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.930
Bartlett's Test of Sphericity	Approx. Chi-Square	3252.968
	df	190
	Sig.	.000

This proves that the sample's performance was as advertised. The significance of the correlation matrices was determined by the scientists using Bartlett's Test of Sphericity. An acceptable sample, according the KMO standard, is one with a value of 0.930. Using Bartlett's sphericity test, we were able to get a significance level of 0.00. We may infer that the correlation matrix is not an identity matrix due to the statistically significant findings of Bartlett's sphericity test.

❖ **INDEPENDENT VARIABLE**

- **Development of an Integrated System**

"Integrated system development" is the phrase for a step-by-step process for making and using a unified framework in which different parts, usually software and hardware, operate together. You can gain coordinated performance, data sharing, and easier communication by putting together different technologies, modules, or subsystems that function together. There are numerous conceivable uses for integrated systems that focus on dependability, efficiency, and synchronisation. Some of these include embedded technologies, automation, healthcare, smart devices, and manufacturing. System development includes creating interfaces, deciding on the architecture, and choosing the right parts. Systems design, software engineering, and hardware engineering are all related fields that need to work together in this way. The main goals of system design are to make things easier for end users and to make the system work better and be able to handle more users. Taking care to every detail will improve the overall abilities. To build integrated systems that work well in technology that is always developing, it is important to make sure that they are easy to use, cost-effective, and flexible over the long term (Wang & Zhang, 2020).

❖ **MEDIATING VARIABLE**

- **Design**

When planning and structuring a system, product, or solution, you need to think about how it looks, how easy it is to use, how well it works, and how to plan and organise it. After making software or an integrated system, the next step in the design process is to think about possible features, describe how they relate to each other, and plan out how the data flow. There is a lot of information on data structures, algorithms, and how to put components together, in addition to system architecture and user interface design, which are all parts of the bigger picture. By connecting theory to practice, it helps developers come up with solutions that meet both technical and user needs. A well-done design throughout the development life cycle should make things less complicated, cut down on errors, and raise the overall quality and reliability of the product (Backhus, 2023).

- **Development**

Design, planning, and development all employ ways to make sure that a product, system, or solution meets particular performance and functional standards. There are several steps in the process of technological and technical development. Some of these steps include demand analysis, design, implementation, testing, deployment, and maintenance. Using both creative and technological ways, you may turn ideas and concepts into solutions that work. Development has a big effect on many parts of hardware projects, such as building the framework, making the end product easy to use, and how well it works. In most cases, you will be asked to work with interdisciplinary teams that employ development tools and platforms that are based on Agile, Waterfall, or DevOps methods. Development is important for all parts of making software, integrating systems, and making a prototype of a product. Development is necessary to make sure that quality, innovation, and expansion continue. The main goal of development is to turn ideas into reliable and useful ways to solve problems that happen in the real world (Zhao & Wang, 2021).

❖ **DEPENDENT VARIABLE**

- **Hardware**

The hardware is what lets computers and other electronic devices process data, run programs, and do their jobs. In this case, "hardware" is the actual physical parts that make an integrated system work. These parts could require things like sensors, CPUs, RAM, circuit boards, I/O ports, and more. The hardware of a system has a big impact on how well it works, how reliable it is, and how well it performs. This is because hardware gives software programs the structure and mechanics they need to work. Careful planning at every step of development, from ideation to prototype to production, leads to hardware that works with other hardware, lasts a long time, and uses less power. The hardware and software parts of an integrated system are deliberately engineered to work together to control, process data, and communicate in real time. One approach to see how well an integrated system works is to optimise its hardware for particular tasks. Modern technological systems rely heavily on hardware to work, run, and be efficient (Lee & Kim, 2021).

- **Software**

The collection of data, instructions, or programs that an electronic device (such a computer) calls upon in order to carry out its functions is referred to as software. Software, on the other hand, as opposed to hardware, is immaterial and acts as the logic that underpins the operation of a system. Software is responsible for managing all aspects of integrated systems, including the structure of data, the physical components, input, and the user interface. Programs, embedded software, drivers, and middleware are all components of the operating system (OS), and they collaborate to ensure that the system operates in a streamlined and effective manner. In order to assist systems in accomplishing their performance and reliability objectives, software engineers are responsible for the creation, testing, debugging, and maintenance of algorithms. In an integrated system, the ability of the software to connect with the hardware is what decides whether or not one is able to automate activities, make decisions, and carry out operations in real time. Because it makes the system more practical, adaptable, and user-friendly, software that has been designed to a high level increases the value of the system when it is implemented (Prataviera & Norrman, 2024).

➢ **Relationship between design and hardware**

The creation of an integrated system relies heavily on design and hardware, which bridge the gap between conceptualisation and execution. Not only does the physical layout depend on the theoretical underpinnings of hardware component integration and experience, but so do operational behaviour and user engagement. When it comes to spaces that need flawless component coordination, optimal use of energy, user-friendliness, and optimum space utilisation, this relationship is crucial. The

reliability, flexibility, and efficiency of the hardware that emerges from integrated systems are often dictated by the thoroughness and calibre of the design process. Determining functional requirements is the first step in hardware environment design. These include the tasks that the hardware must do, the circumstances in which it must operate, and the limitations imposed by factors like cost, space, temperature, and power. These requirements may be transformed into a workable hardware architecture with the help of several design tools and approaches. Problems with the enclosure, electrical routing, component location, and physical layout are all affected by these first design decisions. Consistent and reliable hardware performance in real-world scenarios is the result of good design (Rodrigues & Smith, 2022).

On the basis of the above discussion, the researcher formulated the following hypothesis, which was analyse the relationship between design and hardware.

"H_{01}: There is no significant relationship between design and hardware."

"H_1: There is a significant relationship between design and hardware."

Table 3:H_1 ANOVA Test

ANOVA					
Sum					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	39588.620	447	5726.135	963.671	.000
Within Groups	492.770	752	5.942		
Total	40081.390	1199			

The outcome is noteworthy in this investigation. A p-value of.000 (less than the.05 alpha level) indicates that the value of F, which is 963.671, is statistically significant. We might conclude that ***"H_1: There is a significant relationship between design and hardware."*** and reject the null hypothesis.

- **Relationship between development and software**

The link between development and software is crucial for current technological innovation, especially in integrated system frameworks where software is the fundamental functional engine. Software development is the process of creating, refining, testing, releasing, and maintaining software. Development offers the necessary foundation, tools, methodologies, procedures, and standards for these activities. This relationship is inherently cyclical and iterative since software development is never linear and always changes in response to feedback, user demands, system requirements, and technical improvements. To ensure optimum performance and smooth integration, the development process takes on even greater importance in integrated systems, where software interacts closely with hardware, sensors, networks, and user interfaces. To ensure that software is suitable for both users and systems, the development process establishes guidelines. The first step is to collect requirements and conduct a system analysis, which will disclose the software's functional demands based on its scope and goals. All of the software architecture is built upon this level. Engineers sketch up software components, data flows, control mechanisms, and integration points as part of a methodical planning and design process. Since any control might cause functional inefficiencies or system failures later in life, this meticulous planning step is crucial. So, development is the skeleton that holds the integrated system together, directing the software to achieve its main objectives. The efficacy and flexibility of software are heavily influenced by the development approach utilised (Gupta & Rao, 2023).

The researcher hypothesised that, given the preceding debate, it would be beneficial to examine the connection between development and software.

"H_{02}: There is no significant relationship between development and software."

"H_2: There is a significant relationship between development and software."

Table 4: H_2 ANOVA Test

ANOVA					
Sum					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	39588.620	387	5268.039	1070.30	.000
Within Groups	492.770	812	4.922		
Total	40081.390	1199			

The outcome of this research is noteworthy. F= 1070.30, and the p-value of.000 (below the.05 alpha threshold) indicates statistical significance. A rejection of the null hypothesis and acceptance of ***"H_2: There is a significant relationship between development and software."*** follow from this.

8. DISCUSSION

This study shows that the two most important things that affect how well hardware and software work together in integrated systems are how they are designed and built. The results show how important it is to carefully plan and organise the design process when making hardware. When the design starts with the hardware needs in mind, the hardware works better, can handle more load, and is more reliable. This finding backs up the trend in engineering towards system-level design thinking, which starts with the hardware's needs right away in the design process. The way software is made also has an effect on how it works. Agile, iterative coding, and model-based development are examples of systematic methods that often result in

software is stable, responsive, and easy to integrate. This shows how important it is to have clear development cycles and ways to test software in real time to find bugs and performance problems early on. The study also showed that it doesn't work to keep the results of hardware and software development separate from the design and development processes. When hardware engineers and design teams can talk to each other and software developers and system architects can work together, the results are more likely to be better. The results of this study are very useful for fields like robotics, automation, embedded systems, and the Internet of Things IoT, where hardware and software need to work together. The study looks at how each step in the complicated process of system integration affects the design as a whole. This gives engineering teams useful tips on how to make their systems work better and be more reliable.

9. CONCLUSION

This study's findings suggest that both the design and development stages are equally critical for finding out how effectively hardware and software operate together in integrated systems. The findings show that a design process that is well thought out creates hardware parts that operate better, are more useful, and are more reliable. Structured development approaches also help software perform better by ensuring sure it is reliable, adaptable, and simple to connect with other software. The research shows how important it is to make sure that design objectives and hardware capabilities are in sync from the start, as well as development methods and software features. It's much more important to set up integrated processes that encourage technical teams to work together to lower risks and improve outcomes. In short, the study is useful for engineers and system developers who wish to create better hardware and software systems that function well together, especially in places where speed, accuracy, and overall performance are very important

REFERENCES

1. Ali, M., & Hussain, S. (2023). Challenges and solutions in hardware-software co-design for wearable devices. *IEEE Transactions on Biomedical Circuits and Systems*, 17(1), 12–25.
2. Backhus, G. (2023, June 13). Hardware/software co-design: The five core principles. *Electronic Design*.
3. Chen, M., & Liu, X. (2022). A survey on hardware-software co-design for neural network accelerators. *Journal of Systems Architecture*, 123, 102345.
4. Gupta, S., & Rao, K. (2023). Performance evaluation of AI-enabled embedded microcontrollers. *Microprocessors and Microsystems*, 94, 104422.
5. Iqbal, U., Davies, T., & Perez, P. (2024). A review of recent hardware and software advances in GPU-accelerated edge-computing Single-Board Computers (SBCs) for computer vision. *Sensors*, *24*(15), 4830.
6. Kumar, S., & Gupta, P. (2024). Hardware-software co-design methodologies for automotive systems: A review. *IEEE Access*, 12, 45678–45690.
7. Lee, J., & Kim, H. (2021). Integrated hardware-software design for real-time embedded systems. *ACM Transactions on Embedded Computing Systems*, 20(5s), 1–24.
8. Patel, R., & Singh, A. (2023). Design and development of integrated hardware-software systems for IoT applications. *International Journal of Embedded Systems*, 15(2), 89–102.
9. Prataviera, L. B., & Norrman, A. (2024). Who changes what, when and where? Elaborating postponement when integrating hardware and software objects in global supply chains. *International Journal of Physical Distribution & Logistics Management*, *54*(4), 355-391.
10. Przybylla, M., & Grillenberger, A. (2021, October). Fundamentals of physical computing: Determining key concepts in embedded systems and hardware/software co-design. In *Proceedings of the 16th Workshop in Primary and Secondary Computing Education* (pp. 1-10).
11. Rodrigues, F., & Smith, T. J. (2022). Cross-domain design thinking in complex system development. *Design and Technology Education*, 27(1), 15–29.
12. Wang, H., Liu, Z., & Zhang, Y. (2021). A system-level approach to intelligent system development. *Journal of Systems Engineering and Electronics*, 32(2), 271–284.
13. Zhao, L., & Wang, H. (2021). Integrated design approaches for cyber-physical systems. *Journal of Systems and Software*, 180, 111012.