IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Integrating Human Computer Interaction To Software Development Architecture

Defining and Designing Methodology for Website Processing

[1]Mr. Shrey Solanki, [2]Ms. Archita Dube, [3]Ms. Rujuta Lanke, [4]Prof. Sachin Deshpande

[1]Student, [2]Student, [3]Student, [4]Associate Professor
Department of Computer Engineering,
Vidyalankar Institute of Technology, Mumbai.

Abstract: This study has been undertaken to investigate the flaws in the existing software development techniques to make software both more usable and easier to develop. There are an incredible number of benefits to adopting a human computer interaction (HCI) architecture into software development endeavor. The proposed system defines a method for website processing system while incorporating human computer interaction (HCI) as an important aspect.

Index Terms **– Human Computer Interaction, Software Development, Design, Website Development**

I. INTRODUCTION

To successfully implement human computer interaction (HCI), the need is to understand the basic concepts of user experience design, namely user-centered design and rapid prototyping. It is required to be familiar with design patterns that enable HCI functionality and learn how these interact with other technologies but this must be integrated as a part of the software development rather than having design as a different part of the process. Designers of Software Development Architecture wish to incorporate Human Computer Interaction (HCI) into their applications more than ever before but face issues with one-to-one development. With the development of websites, it is more important than ever that designers are capable of creating intuitive, easy to use products. HCI can be defined as "the design of interactive user experiences that encompass visual and haptics considerations into a system" (Webster).

1.1 User-Centered Design

Creating a user-centered design (UCD) is proved to be a successful way to include HCI into product development. UCD is an iterative process that involves multiple user testing and individual interviews with stakeholders to create a common definition of user. It is important for developers to note that the UCD process happens before any decision about the actual software development architecture is made. This does not imply to skip software design, but it means that while choosing software architecture, the users' comfort must be considered. In order to have a successful UCD, a common framework is created that all stakeholders are aware of before starting with any design.

1.2 Human Computer Interaction

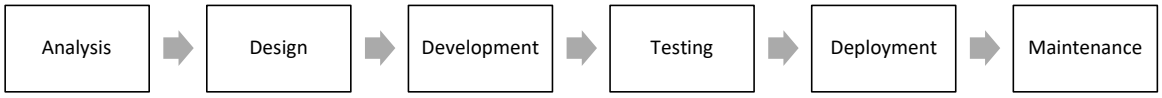
Human–computer interaction, an interdisciplinary subject, has recently emerged as a frontier study area. Human–computer interaction has been progressively used to construction safety management during the fourth industrial revolution, which has considerably accelerated the advancement of hazard detection in the construction sector. However, few researchers have explored the evolution of human–computer interaction in construction hazard identification in depth. [1]

1.3 Software Architecture

A system's architecture is a plan. It offers an associated abstraction for managing system complexity and establishing a communication and coordination mechanism among parts. It specifies an organized approach to meet all technical and operational requirements while improving common quality features such as performance and security. Furthermore, it entails a series of critical decisions about the organization involved with software system development, and each of those decisions will have a significant influence on the quality, maintainability, performance, and overall success of the final product. [2]

1.4 Software Development Cycle

Fig 1.4.1: Phases of Software Development



II. LITERATURE SURVEY

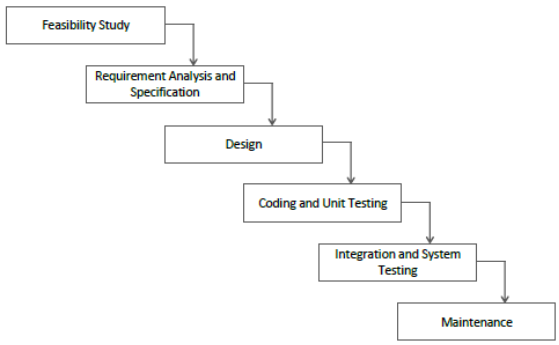
2.1 Software Development Models

A software development model is an expression of the whole software development process that clearly describes the major activities to guide the working tasks on software development that must be done. It is the structural foundation for all of the work and duties that are implemented by system creation, operation, and maintenance across the full software life cycle, as well as the interaction between the many phases of software development activities. As we all know, there are different software development methodologies that have been established and created that are utilized or employed during the software development process. These methods are often known as software development models. However, each model follows a certain life cycle to assure success in the development process, and its advantages and disadvantages are all present. The software development model has a direct impact on the software development cycle and software quality, and it is an essential form of structure and administration for software projects in modern IT enterprises. [3]

2.1.1 Waterfall Model

The Waterfall model is a standard software development approach, although the model's structure is significantly simpler. The paradigm includes development phases such as requirements analysis, design, coding, testing, and maintenance, however each process must be completed sequentially, and requirements must be understood ahead of time. In fact, though, it is difficult to anticipate every element. Most initiatives begin with some ambiguity, and as the project proceeds, more information is revealed. The main downside of the concept is that issues in the system are not detected until late in the process (testing phase), leaving little time for rectification, potentially resulting in severe impacts on project schedule and expense. The use of this paradigm in modern IT enterprises is limited at the moment. [3]

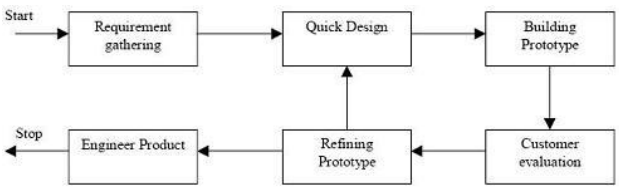
Fig 2.1.1.1: Waterfall Model



2.1.2 Prototype Model

Another basic software development paradigm that is currently frequently utilized is the prototype model. The main concept of the prototype approach is that developers and users quickly define the most critical needs for consumers at first, and a software product prototype may be accomplished quickly. The prototype is regularly changed by developers based on user evaluation and feedback until users are satisfied. Prototype models are appropriate for environments where user needs are unknown at the start of development. Developers should have prior development expertise, so that the most critical criteria may be met by creating a software product prototype. However, applications based on this paradigm may stall the development process, limit developers' inventive ideas, and make updating and managing papers onerous. [3]

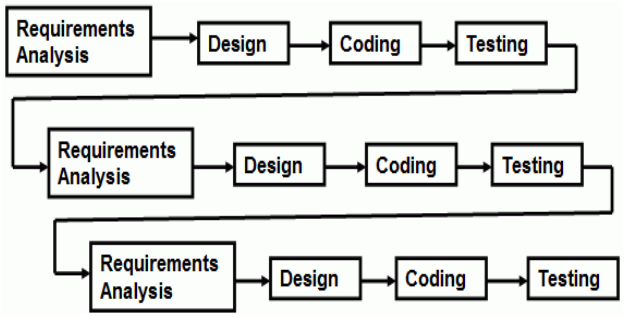
Fig 2.1.1.1: Prototype Model



2.1.3 Incremental Model

The incremental paradigm is not a monolithic model. The model blends the sequential characteristics of the waterfall model with the quick iteration characteristics of the prototype model. The latter increment is built based on the preceding increment, and each increment may be developed utilizing a waterfall or a fast prototype methodology. A software product that is separated into various phases can be released, allowing users to utilize some of the fundamental features while other parts of the product continue to be created by developers at the same time. The methodology is also appropriate for early-stage software projects with design risks and unknown needs. [3]

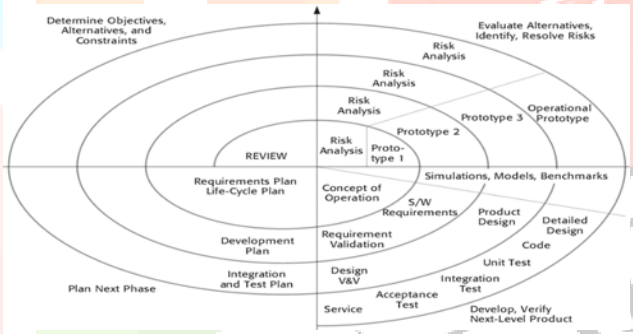
Fig 2.1.3.1: Incremental Model



2.1.4 Spiral Model

The spiral model develops the system using a periodic mechanism. Each cycle is divided into four stages: requirements definition, risk analysis, project implementation, and assessment. Iterations of the four stages are performed. When an iteration phase in the model is completed, software development is promoted to a level. The paradigm has the benefit of emphasizing risk assessments for developers and users in each round of iteration, making it suited for the creation of larger systems. However, the model's downside is that the adoption of the spiral model necessitates the participation of a large number of specialists with competence in risk assessment, and excessive iterations will raise development costs and create submission delays. [3]

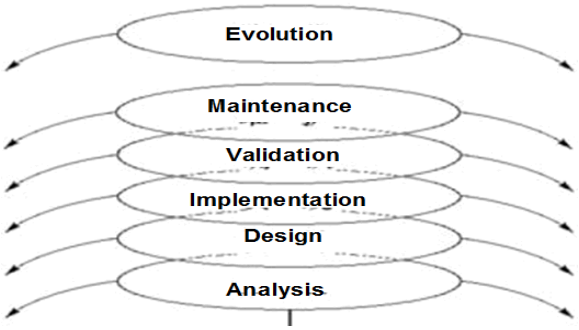
Fig 2.1.4.1: Spiral Model



2.1.5 Fountain Model

The Fountain paradigm is user-driven and object-driven, and thus is appropriate for the object-oriented software development process. In the concept, there are no precise ordering or obvious limits for each phase of growth, and all phases of development might take place on a worldwide scale. The fountain model has the benefit of allowing a progressive piece of software to be introduced on each iterative development, as well as making it easier to fill gaps in other phases of development at any moment throughout one development period. The disadvantage of using the fountain model is that during iterative development, you may encounter the possibility of adding a variety of information, requirements, and information at the same time, which necessitates the use of a large number of developers and is inconvenient for project management. [3]

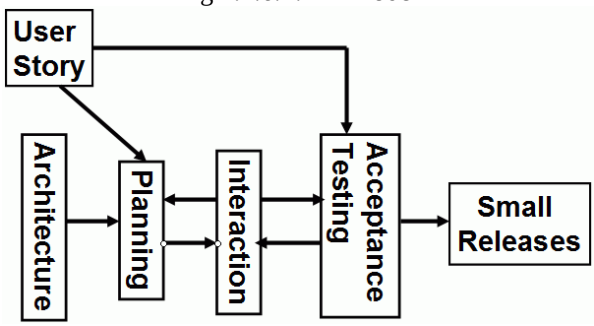
Fig 2.1.5.1: Fountain Model



2.1.6 XP Model

The XP (Extreme Programming) paradigm is a lightweight and progressive approach based on the agile software development process. The whole development process is organized into six phases in the XP model. The model's basic ideas are a collection of beliefs, principles, and methods for rapidly building high-quality software that gives the most value to the client in the shortest amount of time. The XP paradigm is frequently employed in the creation of short, high-risk software projects, and it has the benefit of quickly adjusting to changes in customer needs in modern IT enterprises. Applications based on the XP paradigm allow developers to concentrate on coding while avoiding the time-consuming task of document management. [3]

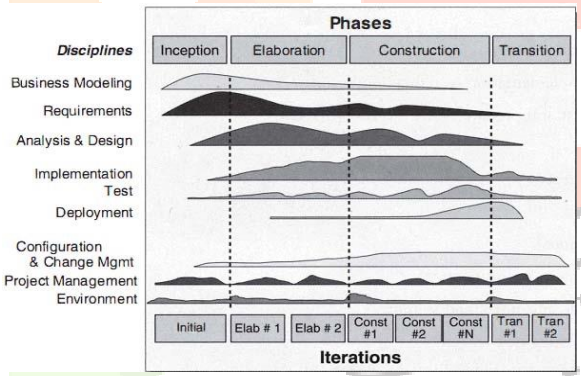
Fig 2.1.6.1: XP Model



2.1.7 RUP Model

The RUP (Rational Unified Process) model explains the life cycle of a complete software development across time via the evolution of project management. The RUP model contains nine main workflows that are employed in sequence and are reinforced on each iteration. RUP's major characteristics include centered architecture, use case driven architecture, iterative and incremental development. However, owing to the complexity of the RUP model's development process and its intimate linkages, it is not appropriate for small software development projects. At the moment, many software engineers are attempting to reduce or simplify various RUP model operations and combine them with the X model in order to successfully apply them to the creation of small software projects. [3]

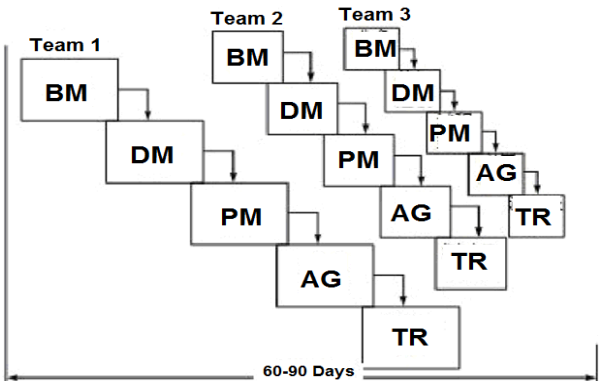
Fig 2.1.7.1: RUP Model



2.1.8 RAD Model

The RAD (Quick Application Creation) paradigm is a version of the waterfall approach that uses reusable components to complete the rapid development of applications. The RAD model is an incremental development methodology that stresses extremely short development cycles that allow a development team to construct a fully working system in as little as 60 to 90 days. Development processes in the RAD model do not use traditional third-generation programming languages to create software; that is, the RAD model is not suitable for development in which the system is not reasonably modularized, and it is also not suitable for development in which the system has a higher risk of requirements, or the interfaces on components must be changed frequently. [3]

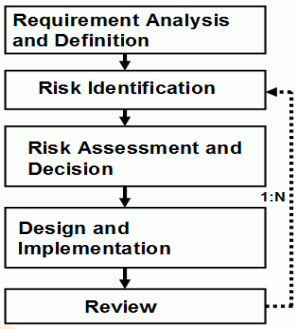
Fig 2.1.8.1: RAD Model



2.1.9 WINWIN Model

WINWIN model combines spiral model features with prototype model, stressing risk analysis and identification, and achieving a double winning outcome for users and developers on software development activities via early agreements. Three objectives on life cycle milestones (defines a set of goals for each significant software engineering activity.), architecture on life cycle milestones (Establishes the goals that must be met when system and software architectures are defined.) Initial operation capability (Represents a set of goals, as well as the preparation for these goals and pre-installation sites for the software to be installed or sold, which is associated with the software help.) are in the WINWIN model, to represent the three different views on the progress of the spiral project development. The WINWIN paradigm has the benefit that both users and developers understand the risk that emerges at each stage of evolution and respond accordingly. They will achieve a balance in terms of developing resources. On the one hand, most of the system's functionalities are accessible to users. Developers, on the other side, may learn about realistic budgets and time frames. [3]

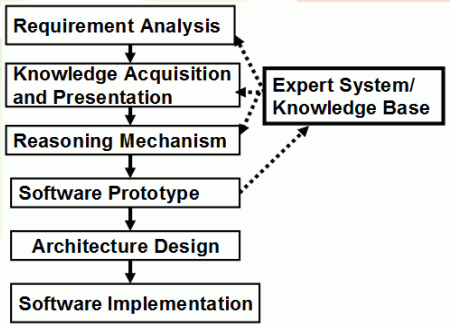
Fig 2.1.9.1: WINWIN Model



2.1.10 Intelligent Model

Intelligent models are also known as "knowledge-based software development models," as they combine a waterfall model with an expert system to integrate knowledge system (based on software engineering knowledge) and expert system (including knowledge rules in the application area), with the expert system assisting software developers in their work. The intelligent model includes a set of tools (such as data query, report generation, data processing, screen definition, code generation, high-level graphics functions and spreadsheets, and so on), each of which enables developers to define certain software characteristics at a high level and automatically generate the developer-defined software as source code. [3]

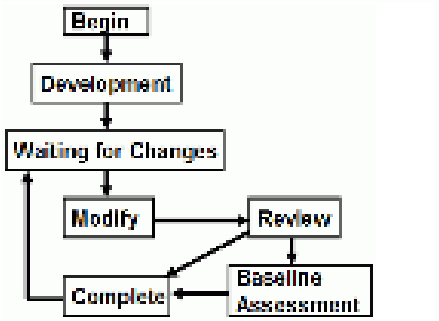
Fig 2.1.10.1: Intelligent Model



2.1.11 Parallel Model

An activity network is a parallel model that focuses on the status of a succession of technical tasks and activities in software engineering. Each online action can occur in parallel with other activities, and this model offers an accurate picture of the project's present status. The parallel model is not driven by time, but by user needs, management choices, and result reviews, allowing for the parallelization of the whole software development process. Because the C/S programme is implemented by numerous components, and each component may be created and implemented in parallel, the model's application is better appropriate for system development with a C/S (Client/Server) structure. [3]

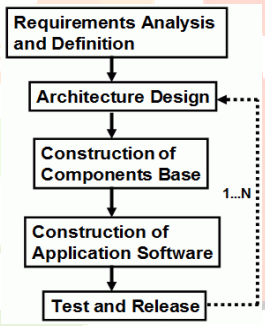
Fig 2.1.11.1: Parallel Model



2.1.12 Component-based Model

The development procedures of a component-based model are iterative, and the model typically consists of five phases. Components, such as EJB (SUN), DCOM (Microsoft), CORBA (OMG), and others, have swiftly evolved into essential software technologies and tools. These are some examples of common new technologies and techniques used in component-based development. The model's fundamental application is to begin the development activity by finding the candidate component by exploring the current component base to determine if the necessary component exists or not. If the component exists previously, it is reused from the component base. If it does not exist, the component must be created using an object-oriented technique. The model's application must modularize the system to be produced and reuse one or more software components in component base by supporting a certain component model. Developers use a comparable combination approach to create an application on a software system. Many software organisations are beginning to construct and apply customised component assembly and interface specification standards to carry out software development operations, however the component interface standards must be generalised further. Furthermore, numerous software professionals have developed several efficient new component assembly methods to finish large-scale software system development operations. The author of the paper proposes that readers who are interested in this read the references on the website. [3]

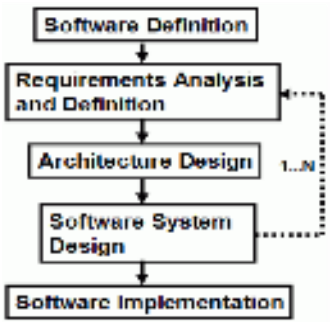
Fig 2.1.12.1: Component-based Model



2.1.13 Architecture-based Model

The heart of this paradigm, which is built on the component-based development process, is software architecture. The analysis and design processes use an iterative incremental approach, translating the functional design space to the structural design space and then back to the system design space. Acquisition and analysis of requirements based on architecture first, in order to give better support for the transition from the analysis to the design phases. The architecture is then designed based on the results of the requirements analysis. It is the same as the component-based development paradigm when creating the architecture to search for matching components based on the syntax and semantics of the constituent elements. The reuse of software components has progressed from code-level reuse to architecture-level reuse. Of course, iterations are present throughout the development process. The benefits of this approach make the framework of the software system more visible and beneficial to system design, development, and maintenance. [3]

Fig 2.1.13.1: Architecture-based Model



2.1.14 Software Development Models in Certain Application Fields

Many software developers and researchers have recently devoted themselves to making appropriate improvements on existing development models based on the characteristics of actual software projects for specific application fields, redesigning processes or activities for new development models, and achieving good results. [3]

2.2 Flaws in Software Development Architecture

The software development architecture entertains a major flaw resulting in a delay during the development and an inaccurate design replica affected by front end development skillset. With the increase in user-centric design and psychological design approach through human computer interaction, designers prepare the software interface with extreme precision and thought process. The prototype made by the designers need to be programmed from scratch by the developers leading to inaccuracies and delay. The flaw in the flow indicates design as a separate task more than the software development.

III. PROPOSED SYSTEM

To successfully implement human computer interaction (HCI) integration into the process, the approach was to come up with a technique to generate code through designs. The vision was to create a superior prototyping methodology for designers and the general users to directly generate the front-end code. Taking a step towards the idea of design to code automation, the concept put fourth is defining and designing methodology for **website processing**.

3.1 Website Processor

To improve the technique of development of a website, we need a platform capable of providing a programmer's precision to any computer literate person. The idea is for the use of making websites to be as easy as creating word documents keeping in mind personalization and customization. The initial step is to make us known to small businesses and website developers so that the team gets enough feedback to expand the advantages for larger scale businesses. The front-end module creates a strong user experience to increase client-side interaction by defining an application's user interface. To apply design thinking and improve the UX of each step of a particular application, it is necessary to empathize with the user.

Websites are built – coded in sections or in blocks. This sets the limitation to the existing system or website builders. Users are allowed to insert their content but within the blocks either as a replacement or at a fixed position, restricting the creativity a person can bring to the canvas. The proposed work presents an idea to overcome this limitation and take a major step towards website processing. A simple comparison can be made with any word processing software which allows users to enter the contents or elements in a line-by-line format. The idea proposes a similar approach to develop web pages which can collectively be linked as a website.

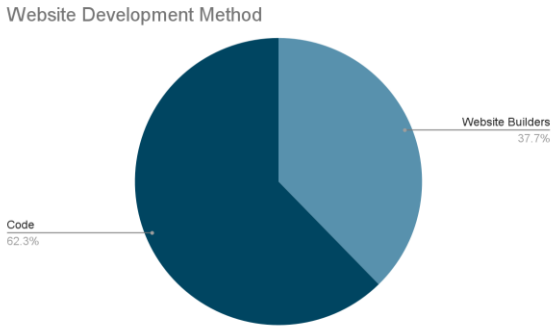
The canvas would include:

- A slew of tools including shapes, texts, graphs, images and icons: The user would have the ability to use combinations of the same to innovate and create a webpage with his/her imagination. Choice of colors as per the content and audience would provide a detailed output as per the requirements and agenda. The users could gain an opportunity to present their idea as planned and structured in their mind without the need of any programming.
- Navigation bar as the header: The header option can have a default name of web pages or can be customized as per the requirements. This place could hold the logo with the links to the webpages or can simply be removed.
- Multiple webpages: A collection of web pages linked together forms a website and the proposed system does that by providing pages. Each page represents one webpage.
- Cloud Storage: All the progress made on current or previous websites can be saved on a cloud storage under the profile of each user.
- Plugins: Provides animations and scripted additions like calculator, forms, interactive interfaces or a converter which can simply convert the designs for mobile preview, etc.

To collect data out of the canvas an AI bot would run through the lines from left to right, embedding each element in the form of HTML code with appropriate spacing and layout in reference to the canvas. A CSS attachment would be made according to a selected template or a customized design system. The user would have the flexibility to plugin JavaScript to enhance their websites with animations and other interactive elements. The inspiration was the interpretation of python that reads code line by line and hence holds several benefits. For our website developer this interpreter would be referred to as a bot. The bot would run line by line converting each element to HTML code. A stylesheet could be implied in the form of CSS. With this the experience of developing a website would be enhanced as the code reading and implementing would be running in the background and the user would only have to focus on their website, personalizing and creating whatever vision they have in mind.

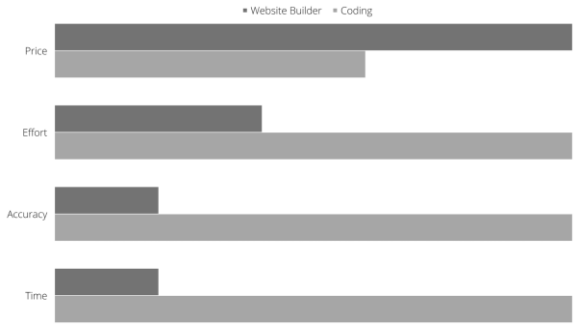
According to the results of a survey made, the performance of website builders among users is rather average.

Fig 3.1.1: Survey: Website Builder against Code



The goal is to improve this user experience such that it meets their demands more efficiently and in less time. The meters against which website processing is done was measured by price, efforts, accuracy and time taken to develop a particular website. The findings showed that the efforts taken using the traditional HTML ways were maximum and hence time taken was relatively high as well. However, the price given for subscribing to website builder tools was high, the accuracy was high and time taken was comparatively lesser as well.

Fig 3.1.2: Survey: Comparison of Website Builder Factors



IV. RESULTS AND ANALYSIS

The website processing technique introduces the user with multiple advantages and features improving the user experience and ease of hosting a website.

Table 4.1 Existing System Comparison

Parameter	Existing Website Builders	Proposed Website Processor
Design to code	Absent	Present
Design Flexibility	Absent	Present
Quick Response Time	Absent	Present
UX Focused	Absent	Present
UI Focused	Present	Present
Design Restrictions	Present	Absent

Our survey shows a high response in difficulty for design to code procedure:

Fig 4.2 Limitations in the existing Software Development Process

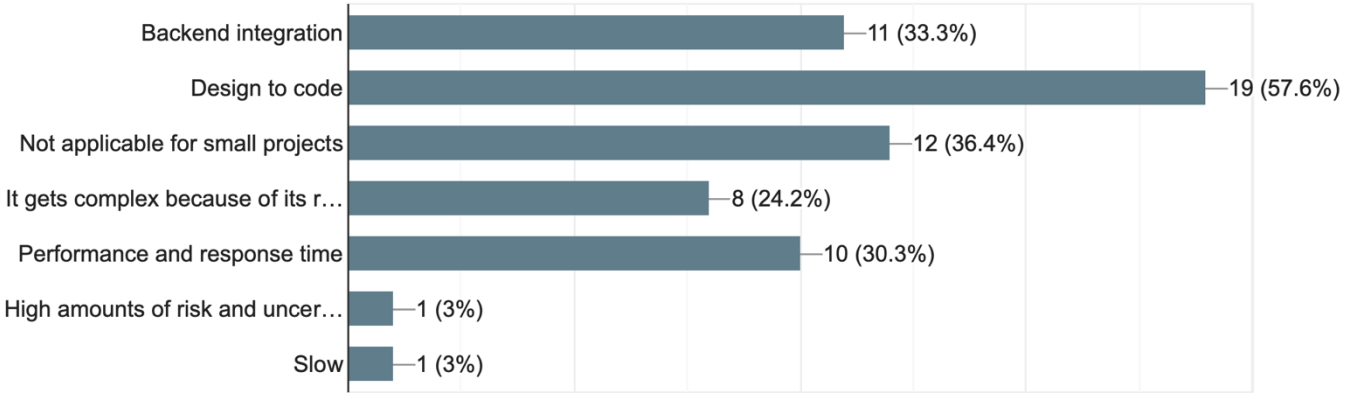
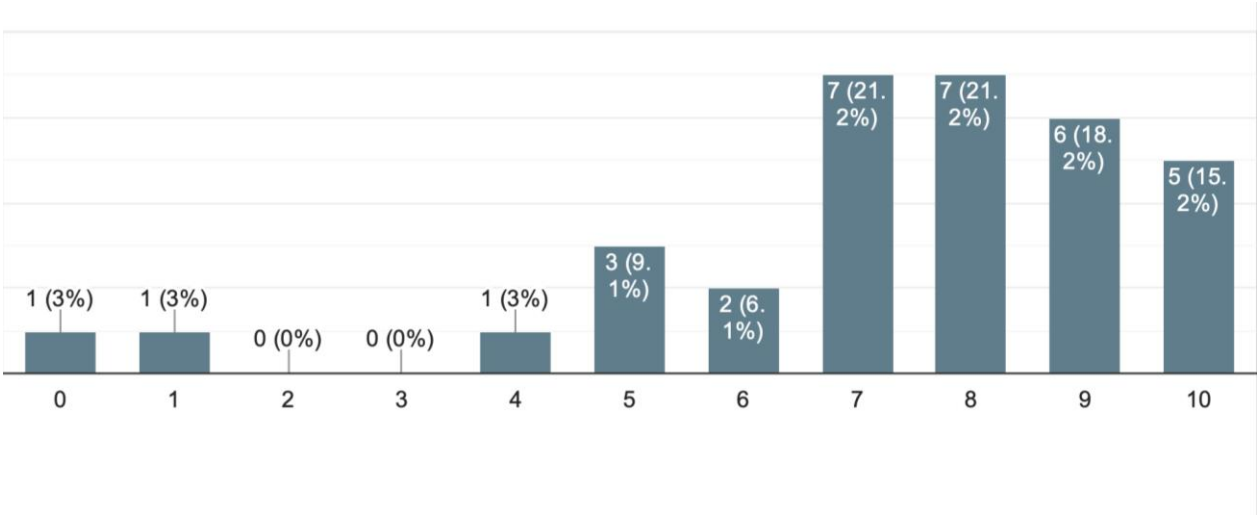


Fig 4.3 Difficulty in Design to code



V. ACKNOWLEDGMENT

We would like to thank our professor and guide – Prof. Sachin Deshpande, through our journey.

REFERENCES

[1] Wang,J.;Cheng,R.; Liu, M.; Liao, P.-C. Research Trends of Human–Computer Interaction Studies in Construction Hazard Recognition: A Bibliometric Review. Sensors2021,21,6172. https:// doi.org/10.3390/s21186172

[2] Jaiswal, Manishaben, Software Architecture and Software Design (November 5, 2019). International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056, p-ISSN: 2395-0072, Volume: 06 Issue: 11, s. no -303 , pp. 2452-2454 , Nov 2019 Available at: https://www.irjet.net/archives/V6/i11/IRJET-V6I11303.pdf, Available at SSRN: https://ssrn.com/abstract=3772387 or http://dx.doi.org/10.2139/ssrn.3772387

[3] Yu, Jiujiu. (2018). Research Process on Software Development Model. IOP Conference Series: Materials Science and Engineering. 394. 032045. 10.1088/1757-899X/394/3/032045.