iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** IV **Month of publication:** April 2021

DOI: https://doi.org/10.22214/ijraset.2021.33836

www.ijraset.com

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

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue IV Apr 2021- Available at www.ijraset.com

Enable Mobile Development of Ordinary Users with Web Components

B. Naveen Kumar[1], V. Sai Kiran[2], G. Sree Harsha[3], Rohit Sutrave[4]
[1, 2, 3, 4]*Department of Computer Science and Engineering, RYME College, Ballari*

Abstract: ***Portable the rapid growth of the Internet has increased the popularity of mobile devices, and the invention of mobile applications has become more common. However, the cutting-edge ecosystem for market development has a significant learning curve. In this article, we consider customer needs and propose a platform component-based mobile interface system for everyday users rather than traditional programming environments. This configuration includes a graphic editor and a web component that can be dragged and dropped. A online component library model suggests standardising custom libraries. A cross-platform programming framework based on web components is implemented to help developers create applications easily. It enables well-known consumers to build packages for various platforms in simple operations. To assist web components in using native functionality, the idea of a native plugin has been proposed. The findings of the experiment show that ordinary users will begin developing smartphone applications in our ecosystem quite soon.***
Keywords: Mobile service, web components, cross-platform, ordinary users, visual development.

I. INTRODUCTION

Mobile innovations have grown at maximum pace in recent years, along with the worldwide adoption of smartphones. In a number of years, several handheld devices have been developed and manufactured. Traditional computers, including laptops and tablets, are now lighter and narrower in line with the trend in mobile phones. Smartphones are an irreplaceable component of the everyday lives of common citizens. In the meantime, the demands of smartphone apps have steadily increased. Smart mobile devices today provide plenty of gadgets, various sensors, Bluetooth, high-definition cameras, etc. They work together and are generated by developers for various innovative mobile apps to transform human lives.

Application production has historically been a technical grammar task: seldom people have programming skills or participate in application development processes. We refer to people with little or no programming expertise as ordinary users in this document. Because of the influential status of programmers, the majority of programming software are developed according to their habit and layout of experience. Assuming that software can be created only through professional programmes, conventional coding techniques have been developed and tailored for programming expertise. Programming needs programming expertise, reasoning skills, expert experience and long-term preparation and it is a difficult and challengesful work. In this conventional view, application construction seems to be equivalent to programming. However, in all circumstances, this view is not true. Although there are small classes in terms of software development capabilities, there is little capacity building. On the opposite, a substantial amount of average people with smartphone applications might generate their own opinions. The challenges in acquiring programming abilities nevertheless retain them. The specifications of ordinary users vary from those of the programmers. First, they choose simple and optimised graphical user interface over complex functions because of their experience. Secondly, they are not susceptible to the success of implementations and their devices are experiments for testing functions and primary education.

The specifications of ordinary users may be a normal condition which we took into account. One designer is a popular consumer and creator of a standard kind. Designers may participate in the process of develop prototypes in conventional software engineering. Similar personnel are designed and executed. Designers or product managers design work and normally have no development capabilities. They use tools such as Photoshop to provide developers with samples. As a result, these individuals will develop applications that cannot be implemented or the end results are lower than expectations. Any revolutionary concepts were excluded from becoming true implementation due to the disparity in specification and implementation. These problems have not been new in recent years[17]. Technologies had previously been developed to meet this problem prior to the spread of mobile internet worldwide. Two typical usable solutions are Microsoft front page and Macromedia's Dreamweaver. They provide comfortable and visual interfaces to support users clear complex codes and build good web sites. Owing to the continuing progress made by mobile Internet hours, the old features of such tools were unable to meet the current demands. The heterogeneity of channels is one of the most important issues for mobile growth [25].

Unlike the Windows scheme, which has undoubtedly played a leadership position on the market for desktop operating systems, there is no absolute leader on the smartphone operating system market in the current scenario.

In the report on market shares in desktop operating systems, Android is the largest proportion (70.24%) in the report on market share for smartphone operating systems, according to latest net- marketshare.com statistics[4]. Next comes iOS, which accounts for 28.34%. It takes less than 2 per cent of Windows Phone, Java ME, BlackBerry, Symvian, and others. These figures reflect the fact that it is a complex project to create apps once covering all mobile devices. Various architectures need separate, mutually contradictory programming mechanisms. Almost difficult to manage all development systems simultaneously, even for experienced programmers.

Therefore, as mobile platforms, the world of programming software reflects the fragmentation. Researchers also suggested many methods in order to solve these problems. The latest development is the use of modern web technologies which is not user-friendly enough and targets expert programmers. Cross-Platform is a requirement for ordinary users' growth

This paper looks at the preferences of everyday users and proposes an environment for mobile application creation based on WYSIWYG web components. The major contributions we made were:This paper looks at the preferences of everyday users and proposes an environment for mobile application creation based on WYSIWYG web components. The major contributions we made were:

1) We describe the current mobile development challenges for everyday users and define specifications that include graphics, automated tools and cross-platform.

2) To solve these problems, we suggest the cross-platform, web-based mobile development environment and models. The graphic editor allows users to visually combine web modules and customise them. The standardisation of personalised libraries is provided by a new component library model. The native plugin paradigm facilitates the invocation of native functionalities of operating systems as a functional extension of web components. The automated framework development technique allows developers to create several platform apps quickly.

3) In a community of everyday people, we execute realistic trials and the findings demonstrate the outstanding usability and comfort of our environments.

II. RELATED SYSTEM

A. Programming-Based Development Environment

Historically, traditional software frameworks aimed at experienced programming developers. The software environment focused on programming incorporates the developer-centered design guidelines, which have been used to improve user interface with a range of actions. The standard programming environments of mobile platforms are typically developer-centered. Android development environments can be created with two options: Eclipse and Android Studio, all of which come from traditional JAVA developer environments and software relevant to the Android SDK.In addition, the iOS developing system, XCode, begins with relative resources and compilers. The iOS development environment starts explicitly. This continuous design above makes sure that mobile apps can be quickly handled by developers with relatively linguistic coding backgrounds while maintaining the restrictions in software creation surroundings. The downside is that programming and creation capabilities are too difficult for everyday users to cultivate. Programming is a systematic capacity requiring not only professional management of programming languages for the required programmers, but also an excess of information layout and operating system expertise.Traditional programming environments are not suitable since their operational methods do not meet the operating patterns and information framework of everyday users. The programme creation phase was facilitated and accelerated by the proposal of code models, project generators, configuration types and other automated coding generations. These approaches to a certain extent minimise the difficulty of execution by helping developers avoid repetitive work and automatic code blocks.

B. Visual Development Environment

A number of strategies are presented to change standard construction environments in order to deal with these problems. In many mobile development environments, a visual editor, a graphics tool or plugin for drag and drop, is commonly used to promote the creation of GUI. On the basis of the designer's prototype, several products have continued to function in the area of graphical programming. The visual component is used as a simple modelling unit typically by current graphical programming software. As an application as modern graphic development equipment, the visual component is seen as an integral device of design. One visual component is an immersive software user conceptualization of one or more features to display and change local and/or remote data packed to allow knowledge to be discovered, installed and combined in a coherent run-time sense. A part can not only be run by itself but has a collaborative gui, which can be compiled and worked together to enable the development of the application.

Visual components are commonly used as a method for visual infrastructure orchestration in functional research in composite services and applications using MashMaker. Semantic composition interface for different software and resources was proposed for reference. In order to create composite services, a reference suggested a graphic composition environment. A service development environment is implemented in MicroApp mobile devices. MIT Software Inventor is an elegant visual programming interface allowing everyone to create completely functioning Android mobile and tablet apps, including for children. This goods have unresolved common issues. First, it supports only single platforms. Secondly, graphic component functionality in GUI design is minimal.

C. Web Components

Production settings, as noted above, accept only one operating system. This condition is brought on by adaption on all platforms; often programmes with the same aspect and functionality need repetitive work. Multiple mobile platforms need a completely different SDK and application development. This job goes far beyond the skills of average users. To solve the problem of convergence of mobile networks, a cross-platform is proposed. It is a term that indicates a mechanism that develops quickly without taking into account device distinctions. Online portion is tailored to web sites and web apps, reusable, encapsulated as HTML tags. Custom modules may be found on any JavaScript repository or application that deals with HTML through modern browsers. A web application is a kind of mobile app that utilises pure web technologies in a browser to imitate a specific app recognised as a native invention in the traditional sense. The Web Interface is an all-embracing solution.A web application is a kind of mobile app that utilises pure web technologies in a browser to imitate a specific app recognised as a native invention in the traditional sense. The Web Interface is an all-embracing solution.

AppGyver Composer is a smartphone production, online cross-platform tool combining graphical programming with a cross-platform web-based approach. It offers GUI and its rationale to users with a graphic programming editor. Drag-outs can be supported and users can adapt their HTML and JavaScript tags. A critical negative feature of the Web App appears, which is that native APIs are not supported, while the issue has been resolved. Native APIs provide a range of mobile platform hardware interfaces such as camera, sensor, wireless Internet access, Bluetooth, etc.without which production of various mobile apps will be seriously limited. Web apps cannot use all the functions of native APIs since they restrict the browser gui.

Via a hybrid programming architecture, PhoneGap bridges the gap between the JavaScript interface and the native functionalities of the mobile app ecosystem, enabling Web technology to create strategies for the development of mobile apps around the platform. Via the use of PhoneGap, developers will use a single code base to generate an application on each platform. It encourages developers not to construct deep compatibility layers but to provide a fantastic experience. Centered on this architecture, various streamlined programming environments are suggested to facilitate mobile device development on a web-based cross-platform basis. iCloud HBuilder is a programmer-oriented web hybrid application creation platform. It incorporates numerous features to speed the phase of development and simplify programming with support for cross-platform application design. It does not go any further in promoting normal users' growth, as do other development environments.

III.PROPOSED SYSTEM AND MODELS

This segment presents the global framework and the detailed theory of the models that we suggest, including the library model for web components, the plug-in model, and the framework implementation model.

A. Architecture Overview

The environment is a web-based application that can be accessed on any device through browsers. Figure 1 displays the overview frame separated into three components: library archive, graphic editor and server module. These modules are based on OSGi, a dynamic Java package that offers an application model consisting of several bundled elements, communicates via system control interfaces, reduces system complications and makes it simple to coupl.

1) *Library Repository:* A collection of site component libraries is the library repository. A library is a web component container containing the fundamental data that describes the web component set, encapsulated in a separate group of OSGI system. All libraries available are loaded and recorded dynamically in the service registry through the OSGi system as seen in Figure 2. The server module's library manager detects libraries from the registry during operating period. The palette of components in the graphic editor defines details of the web components and makes them available to the GUI. An API library offers another service gateway

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue IV Apr 2021- Available at www.ijraset.com

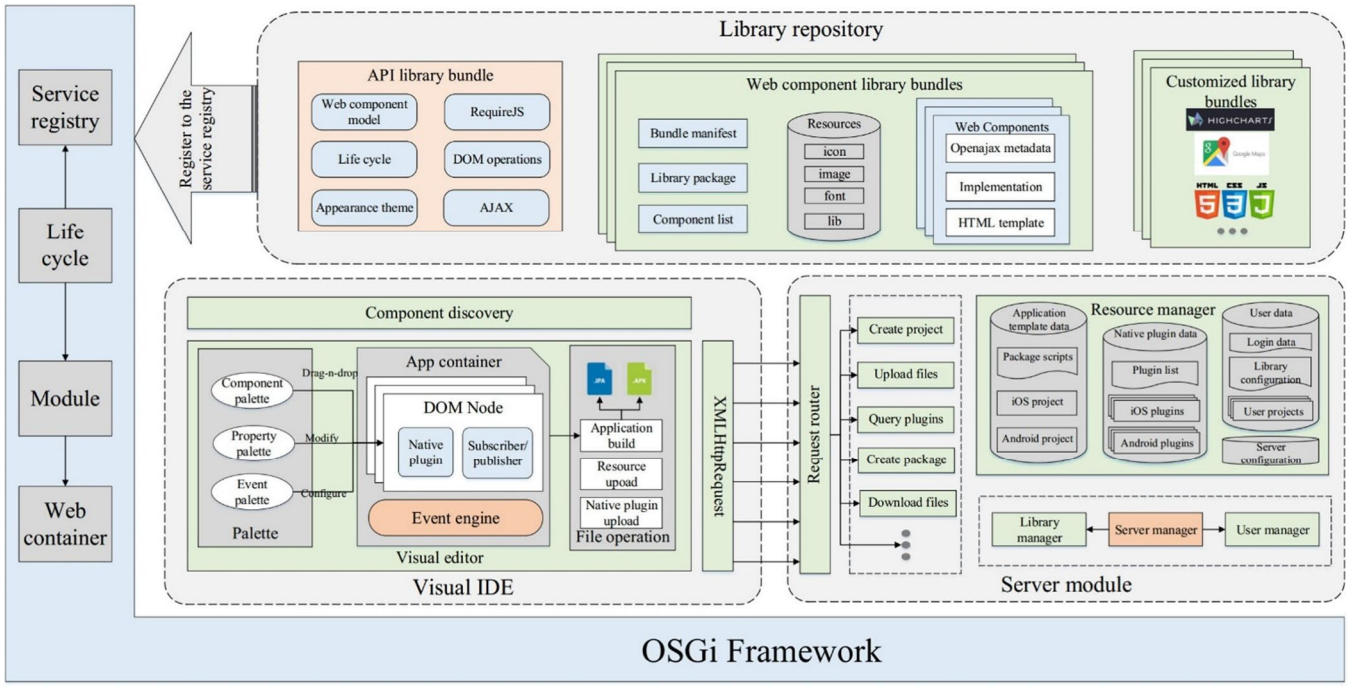


Fig 1. The Architecture Overview of the Environment.

Bibliothèques which include the new web platform model, life-cycle management, extended modules import RequireJS, the DOM interface, the AJAX interface or the presentation issue etc.

2) *Visual IDE:* The IDE visual component offers an interface for web elements to be dragged and dropped. The important component is the graphic editor with three elements, the application jar, the palette and the process. The exploration module of components gathers knowledge on web components. The XMLH5-0Request includes a server framework gui. The graphic editor has a working region for WYSI-WYG. The graphical designer was able to construct a container to enable new project which lets customers and users to establish a working execution environment for the customer's application software. in order to make it visible to users, the site feature is fed into the web page-bound frame container and visually positioned on the page In the item picker, the selection determines if the option is fixed in place and shown, all of which are animated. On the basis of event-controlled model, web modules may exchange information. The application's event engine listens for and responds to publisher/subscheduler event notifications. This package contains a combination and personalization toolbox for the web modules. Elements in the palette: pallet, assets, and cases. We present an array of useful site components on the Creative+ palette. When an icon is moved to the Graphic Editor, an incident is fired. The palette lets you change the look of the application with the Appearance page. Customization of event functions includes functions such as third-party modules like Bluetooth, monitor, etc. The file module supplies various projects such as exporting, uploading, and setting up an interface with built-in software.s. File upload offers an easy way to apply to apps tools such as images and multimedia. Native plug-in upload enables users to customise the native interface to the plug-in model defined in depth below. Application construct is the key component of quick growth, package installation and cross-platform convergence, to achieve the final stage in this phase. It enables end-users to create several mobile platform installation bundles for a single user project via clear visual processes.

3) *Server Module:* The programme module incorporates the back end business logic of the server, which includes server administrator, consumer and librarian manager.. A web container will be instantiated and the server manager will interpret system configurations. The user management is in charge of security and change of users. The database manager finds and loads the collections of registered Web Components from the OSGi Application List. The request router receives and distributes queries from the visual IDE to the relevant handlers. The resource manager saves files, native plugins, system template and server configuration. User data records and project les for each user's metadata. Plugins are fundamental information and infrastructure,

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue IV Apr 2021- Available at www.ijraset.com

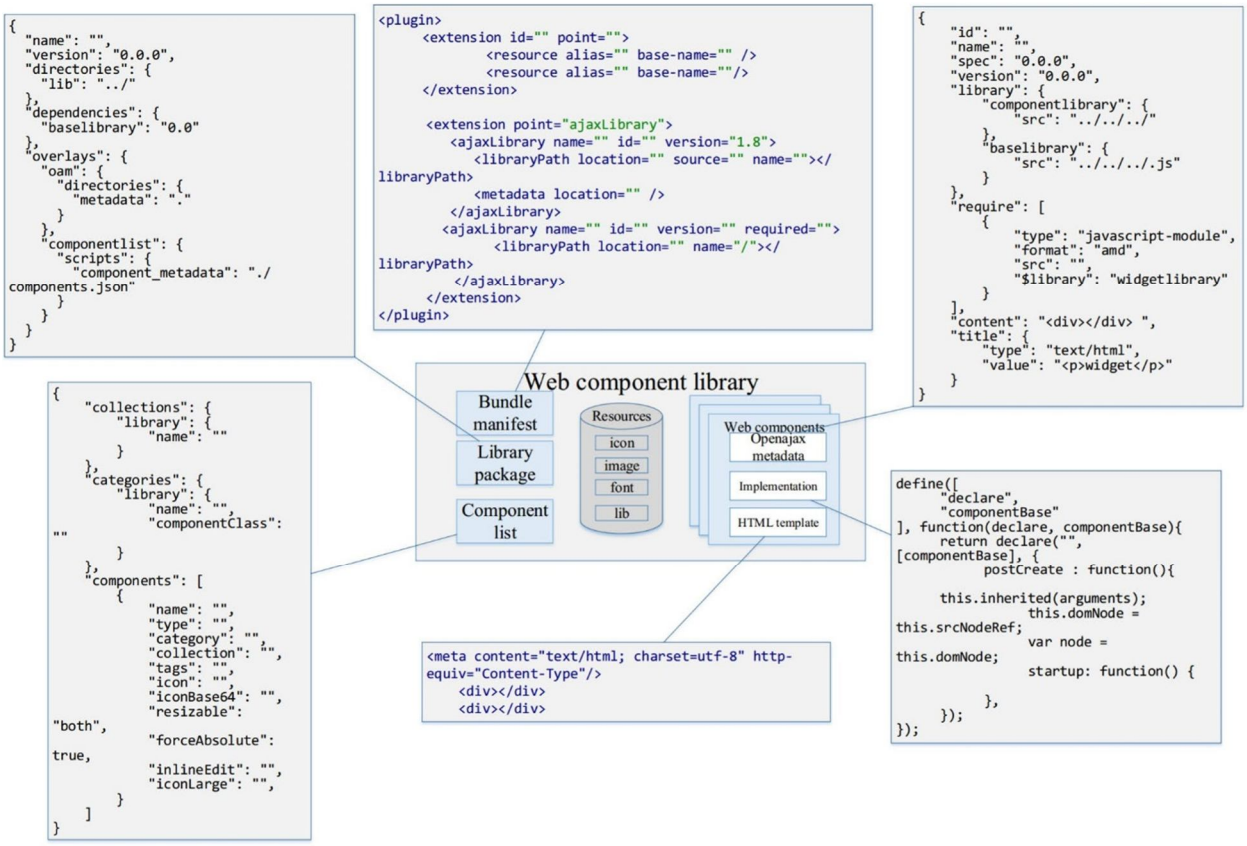


Figure 2. The user experience model.

Includes the name of the plugin and the user's feature list. Some other resources are especially needed for native plugins. Data from the application template is the key resource for creating packages installation. The main building tools include script execution, prototype, and log data. The Android edition and the iOS version for native plugin details and project template information are kept separately..

B. A Web Library Model

In a standardised W3C framework, a part of the Web developed as a reusable DOM HTML element is applied across all devices, from desktop to smartphone. A website can have many purposes. This metaphor describes the 'the library definition' shown in Figure 2.

This is the concept of web components, or building blocks, using the OpenJPA and HXT libraries as foundations, respectively. OAM is an industry-defined standard for enhancing AJAX interoperability An Internet website consists of an OAM file that details the metadata, which includes a web feature in your graphic editor, including your own id number, the library name, the resources of the library, the definition and other optional properties. The application is connected to the web component model inherited from the API library and provides the base interface with a metadata of the AMD bundle. The implementation is linked to the model for the web component. The RequireJS Application imports remote modules asynchronously and uses XMLHµRequest to navigate web resources. The HTML prototype provides a frame in the rendering phase.

As an OSGi package of metadata, tools and web modules a standard library is introduced. Three basic components, bundle manifest, bibliographics and a collection of components are stored on the metadata part. The bundles manifest file registers configuration details, including position of the library, identifier and other data, to establish a service registry OSGi package. The registry loads libraries to the server for the OSGi operation. This component lists all the components used to describe and locate the library as well as their icon name, form, features, and features, including a definition. According to the library, the most valuable library materials included things tomes, data sets, and collection additions.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue IV Apr 2021- Available at www.ijraset.com

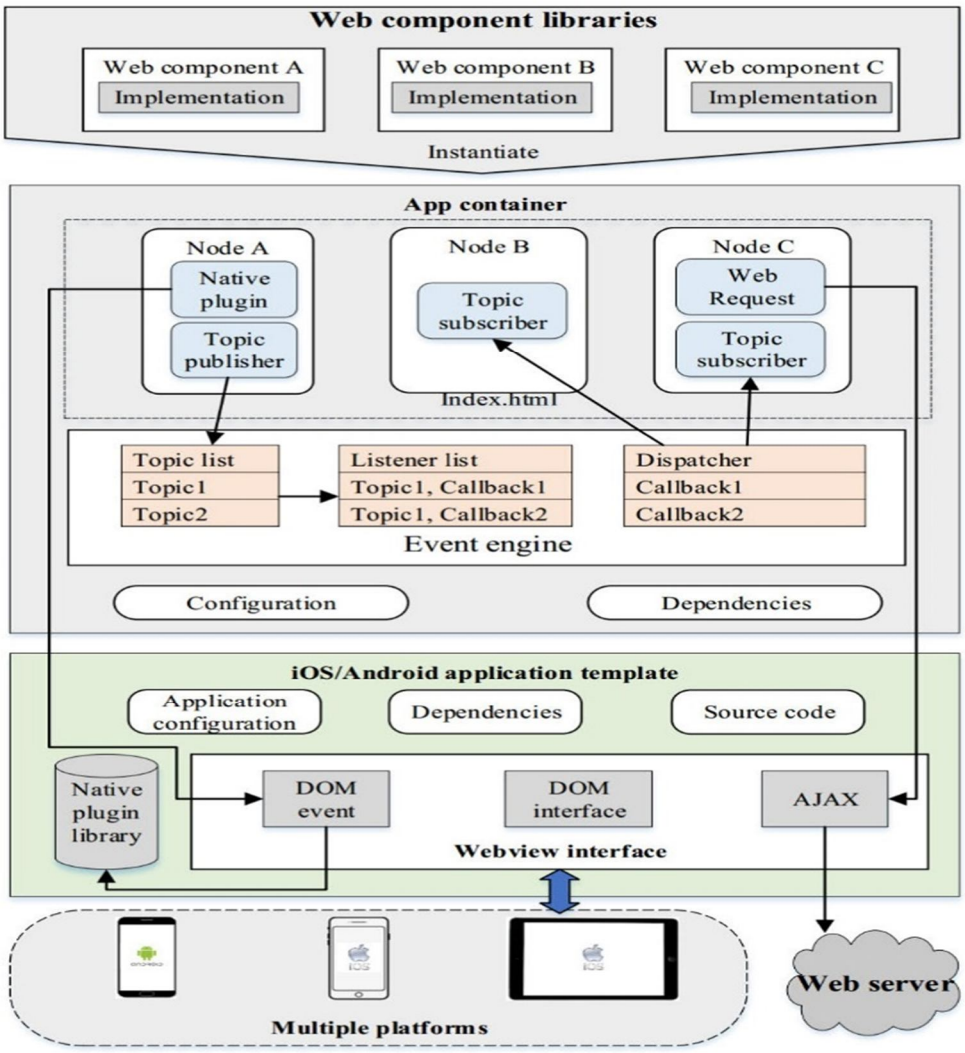


Figure 3. The cross-platform application model.

Directories. The library's resources include all the dependencies, photos, media files, and fonts used for web-based feature instantiation.

C. Native Plugin Model

Native plugins provide user interfaces with a channel that access OS interfaces to link the Web to native features. The following five basic components are a standard native plugin:

A device identifier supported by the plugin is an iOS or Android platform;

A collection of functions showing the interfaces required for invoking web components;

The package template project setup le: It depends on various standards of the network. It reflects the features of authorization and project plugin for the Android platform. For iOS, the root class setup is shown.

Resource files, such as JavaScript libraries, photos, CSS files, and other necessary Webview executed JavaScript files, OS end primary class source files. The resource file JavaScript libraries communicate with the class and supply interfaces for web components which exchange data between web components and OS via the JSBridge protocol [9].

Native plugins may be programmed and activated by the event engine using the corresponding JavaScirpt Interface. The native plugin data are stored in the server layer of the resource module. To get access to the collection of plugins, the view layer sends the request to the graphics editor. When a mobile application is allowed, the respective native plugins may be copied according to the user's preferences.

D. Cross-Platform Application Model

In this part, we present our proposal and internal theory for the Cross Platform Model Application (CAM).

1) *Cam Overview:* A CAM is a free platform framework container and software prototype, as seen in Figure 2. Server container provides the instance and the execution framework of web components. A system container has an HTML file with site components instantiation, an event engine file and a le configuration resource file which record a native plug-in list and the platform resources. The application template for the app containers is a native application wrapper. Templates are split into many variants according to various platforms, but for each template similar sections exist. It offers a WebView environment for running app containers with a page parser as well with a web interface. It protects the assets and permissions of the governing programme. A native plugin library provides interfaces to web components which rely on native features.
2) *Web Component Communication Mechanism:* The web component contact process in the cross-platform application system is described in depth in this part. The coordination process specifies approaches to data transmission for the instantiations of web components. The primary coordination function in runtime is shown in Figure 2. Implementation of Web components generates instantiations through the creation of HTML nodes in the application containers and the provision of data from other components, a Web server or operating systems through multi-user chansons. The key types of communication are the following:

a) Component-to-component: Via an automated publishing and subscribing to an application, web modules can communicate with each other.

Web components can leverage native plugins to communicate with the operating system..

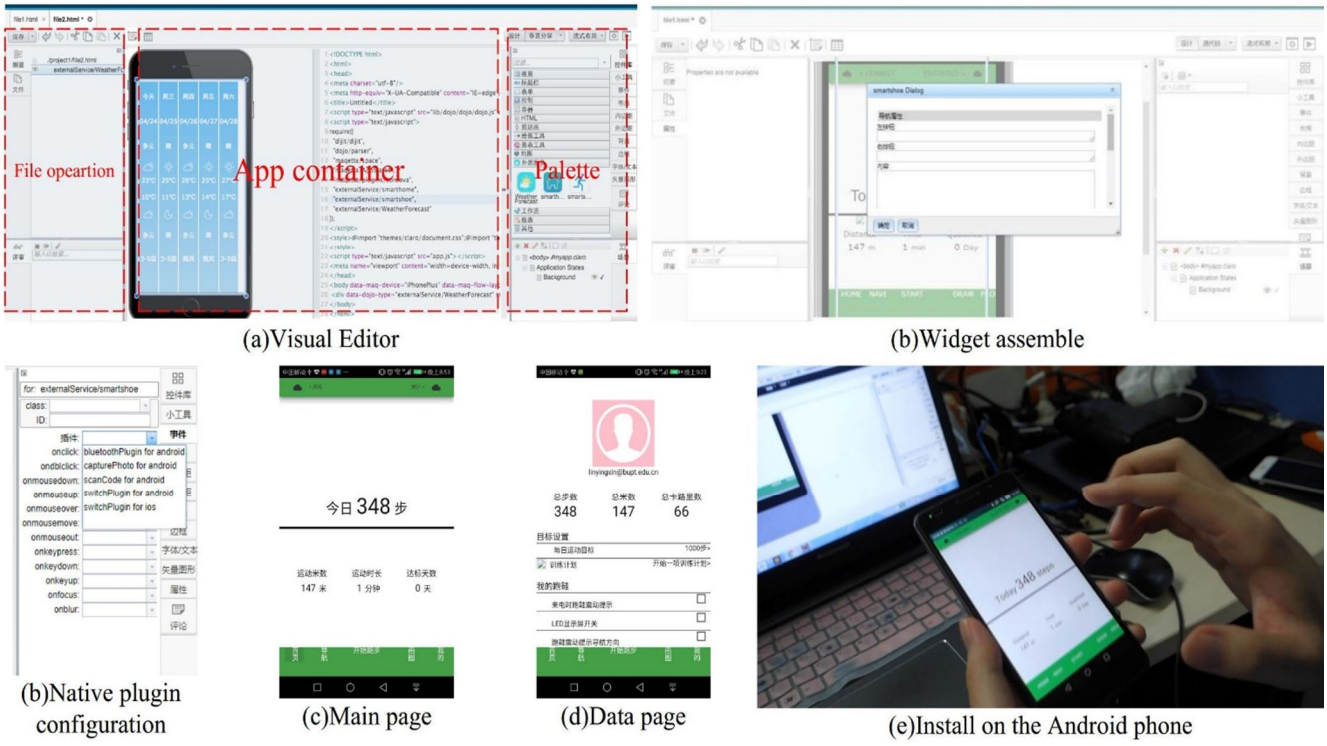


Figure 4. Working theory and practical application.

With this Native Plug-in channel, the OS end can obtain, solve, and react to WebVeiw info.

b) *Component-to-Web:* The AJAX system supplied by libraries and the WebView gui from the web components can be used to send RESTful queries to remote data servers.

c) *Event-based composition Instantiations are assembled and transformed into mobile applications[24] by composing the web section. The composition method consists of two stages:* The second is the development of links, relationships through network elements and the proper functioning of mobile apps through web-button Communications. Firstly, the GUI is to instal each part, in accordance with conventional logic and user requirements, via drag and drop operations.

The facility is event-driven. It brings two audiences together. The source of an occurrence is a part of a site, and others can have the effect of addressing it. the idea that any event is unique:

$$< Target; Topic; Object >$$

A specific instance of the media source aspect serves as a target. The subject details how the incident is sent to the event handler and how it is stored by the user. A JavaScript entity is an object that has its data handled by purpose.

A client-side pub/sub frame module is a JavaScript broadcast mechanism. With Liferay, components may produce topics and send messages or engage with the system to engage with individual subscribers can be built if required. Two functions can be defined with these two implementations in JavaScript.:

Publish(*topic*; *object*); *subscribe*(*topic*; *callback*)

A subject list and a list of listeners are maintained by the event engine. The event engine tests the list of topics and then crosses the list of the listener, invoke the callback functions successively so it can have the web components that subscribe and start a callback. The web components can accept the request. If the event occurs, the web component calls the user to the connection.

3) *Application Build Approach:* In a user-transparent cloud module the programme build service is encapsulated. An application prototype is described as a set of necessary files for the production of the application. The server-side resource manager manages configuration models for various devices as well as appropriate package files. In the service field, users submit requests via GUI checks. The creation page handler on the server side will collect framework choice parameters if a package request is sent and the configuration container will be copied from the user's direction to the applications template concerned. In the meanwhile, a catalogue of plugins held by the

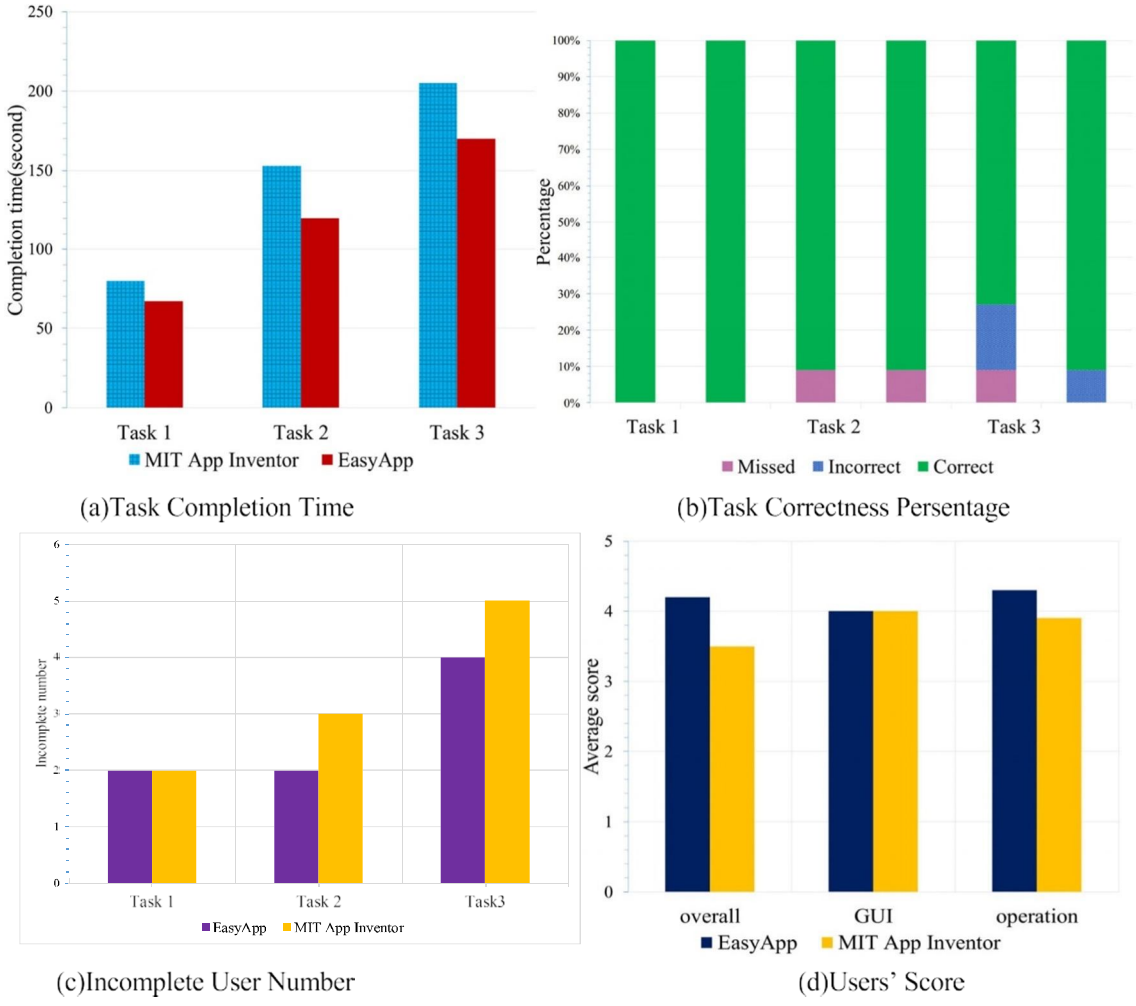


Figure 5. Evaluation results.

a) *Creative Idea:* The result of the analysis is translated and the creative projects are copied into the library, where the indigenous software can use it. Incorporation of the application settings and requirements into the design template The device application kit script is launched to generate the APK or IPA for the application.

IV. WORKING THEORY

We are implementing our proposed ecosystem, based on Eclipse Equinox, the OSGi Architecture. The Mobile Toolkit of DOJO provides a framework for web modules implementation. Apache Cordova is the basis for the development of indigenous plugins and devices. The ecosystem includes several desktop operating systems such as Windows, Mac OS and Linux. It can also be enabled and downloaded through local browsers on remote servers. We show the functions of the Lenovo laptop in this article. Figure 2 shows the visual editing tab. In the centre of the website is the device jar. This is the area of page modules and device templates assembled by users. The display modes are supported in two formats, graphic and source. On the right side of this page is the palette tab bar. On the left hand side of the page is the surgical area.

V. DEMONSTRATION

In this section we demonstrate the personalization and framework development processes using web components and plug-ins. We build relative site modules for the Bluetooth software using the following procedures. Blue tooth switch, amount of mobiles and the detection of smart devices including smart shoes are the main requirements.

Customize Native plugins and web components: This step is easy for regular users in normal circumstances. This approach is illustrated and how extended libraries and local plugins may be imported. First, we built an OSGi kit, with a service activator and most relevant configuration files, in the EasyApp root directory. Secondly, we introduce a web component with meta information, resources and execution le to the library. Finally, in the graphics editor, we complete the GUI. In the JSON setup the details for the component palette is registered. A library table with a part object in the palette context is included in this situation. A new JSON item, displaying a new tab and portion, is created in the palette.

We require a Bluetooth plugin in order to enforce the native features. In order to save native gin resources, we are creating a new directory. Java is used with Bluetooth and for all a JavaSkript interface. In different directories we have put les and Java JavaScript and adapt them in the three directories. At last, we compact and upload the directory to the environment.

Build a new graphics editor project, move your website section to the application container for adjusting size and location as seen in Figures 4(a) and 4(b) and change then your home plug-in. Part assembly & development Tap the Android box icon as shown in figure 4(c) to 4 to get the installer and request for evaluation on an Android smartphone. [e].

VI. EVALUATION

The purpose of this assessment is to show how beneficial our environment is in comparison with other proposals for ordinary users. Although ordinary users have specific criteria, the cross-platform implementation objectives have not been achieved. That is why we use the pace of growth and customer interface as the main goals.

We have an experiment in comparison with two instruments, MIT App Inventor and EasyApp, to test the usefulness of EasyApp. We've arranged 22 users without expert programming at Posts & Telecommunications University in Beijing to view a video tutorial and finish three tasks in order to create related applications in two environments. Figures 5(a) and 5 represent the effect of accuracy and performance (b). EasyApp is more rapid for four functions than the MIT App Inventor. Finally, 64 activities were performed correctly by the participants in EasyApp and 60 in the MIT App Inventor. The average result was calculated and figure 5 shows the effect (c). Following the creation phase, both participants sent a questionnaire in three groups, which included five alternative responses, namely: 1 (strongly disagreeable), two (disagreeable), three (neutral), 4 (agreement) and 5 (subsequently) (strongly agree). The average scores for each group are shown in Figure 5(d). The overall dominance and the service of the MIT App Inventor are apparent and the GUI score receives the same score.

From the result, we can see that EasyApp is user-friendly and superior than some tools in certain areas.

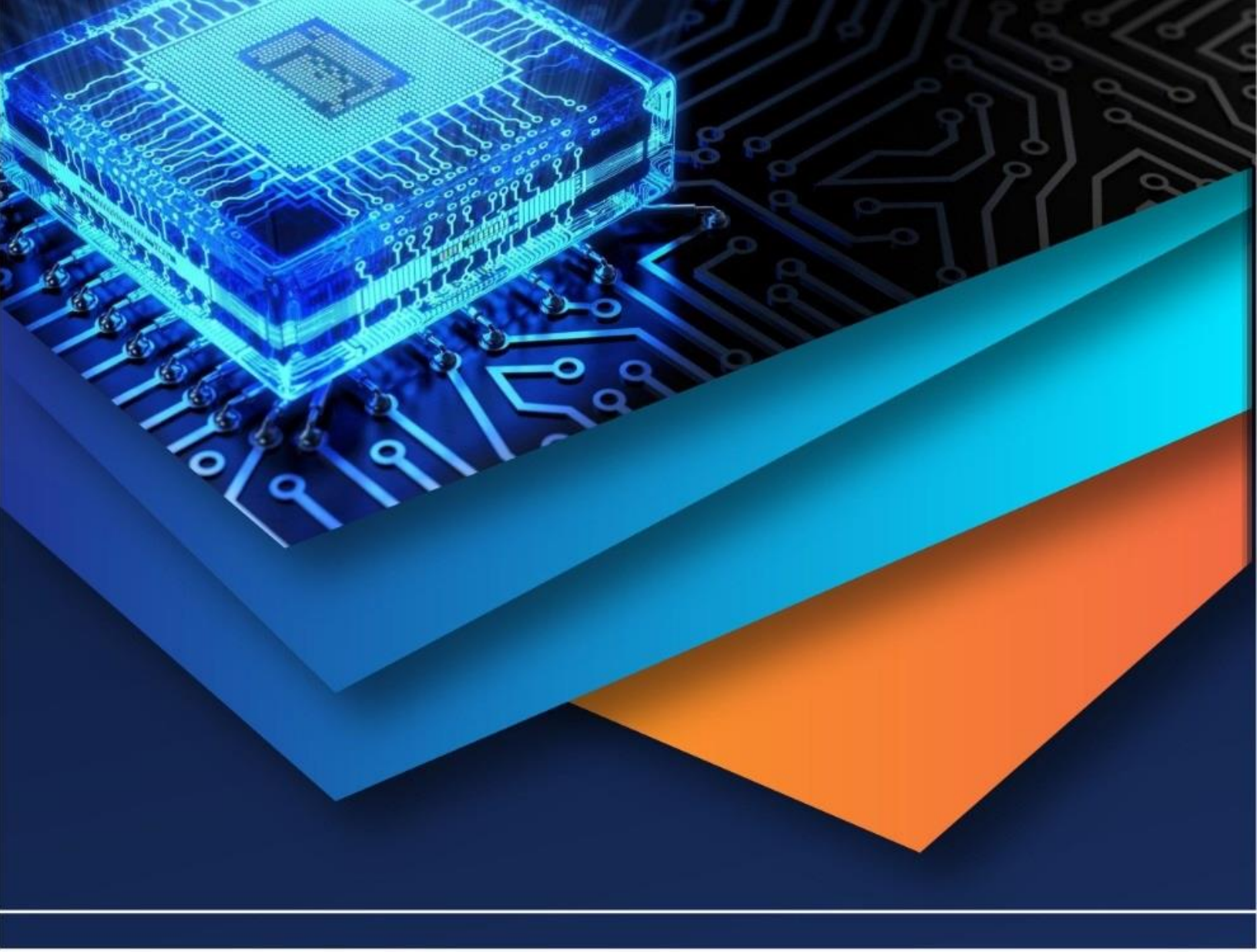
VII. CONCLUSION

This paper suggests a smartphone app creation environment for ordinary users for the cross platform WYSIWYG. We explain in depth the architecture and execution. It offers users an easy-to-operate visual editor, plenty of online and plug-in component libraries, and an automated application development strategy. In summary, our proposed environment learnt from conventional developments and has enhanced the functionality to make it more user-friendly.

In future jobs, three issues are primarily covered. Firstly, the functions that extend web component libraries to satisfy different requirements are targeted at the environment. JavaScript has evolved with the advancement of online techniques and several excellent frames have been written. We are searching at more frameworks and repositories to include end-user applications with a view to expanding the array of web components libraries. Second, to increase execution efficiency and to encourage more inter-platform mobile apps. We expect the growth of mobile cloud apps to be tailored. Third, the creation process is more automatic by planning and learning technology.

REFERENCE

[1] P. Wang, F. Ye, and X. Chen, ''A smart home gateway platform for data collection and awareness,'' IEEE Commun. Mag., vol. 56, no. 9, pp. 87–93, Sep. 2018.
[2] Z. Zhai, B. Cheng, Z. Wang, X. Liu, M. Liu, and J. Chen, ''Design and implementation: The end-user development ecosystem for cross-platform mobile applications,'' in Proc. 25th Int. Conf. Companion World Wide Web (WWW), 2016, pp. 143–144.
[3] Y. Maezawa, K. Nishiura, H. Washizaki, and S. Honiden, ''Validating ajax applications using a delay-based mutation technique,'' in Proc. 29th ACM/IEEE Int. Conf. Automated Softw. Eng. (ASE), 2014.
[4] D. Liu, L. Khoukhi, and A. Hafid, ''Prediction–based mobile data offload- ing in mobile cloud computing,'' IEEE Trans. Wireless Commun., vol. 17, no. 7, pp. 4660–4673, Jul. 2018.
[5] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, ''McCloud: A context-aware offloading framework for heterogeneous mobile cloud,'' IEEE Trans. Serv. Comput., vol. 10, no. 5, pp. 797–810, Sep. 2017.
[6] R. Francese, M. Risi, G. Tortora, and G. Scanniello, ''Supporting the development of multi-platform mobile applications,'' in Proc. 15th IEEE Int. Symp. Web Syst. Evol. (WSE), Sep. 2013, pp. 87–90.
[7] R. Francese, M. Risi, G. Tortora, and M. Tucci, ''Visual mobile computing for mobile end–users,'' IEEE Trans. Mobile Comput., vol. 15, no. 4, pp. 1033–1046, Apr. 2016.
[8] L. He, G. Meng, Y. Gu, C. Liu, J. Sun, T. Zhu, Y. Liu, and K. G. Shin, ''Battery–aware mobile data service,'' IEEE Trans. Mobile Comput., vol. 16, no. 6, pp. 1544–1558, Jun. 2017.
[9] N. Laga, E. Bertin, R. Glitho, and N. Crespi, ''Widgets and composition mechanism for service creation by ordinary users,'' IEEE Commun. Mag., vol. 50, no. 3, pp. 52–60, Mar. 2012.
[10] A. H. Ngu, M. P. Carlson, Q. Z. Sheng, and H.-Y. Paik, ''Semantic-based mashup of composite applications,'' IEEE Trans. Serv. Comput., vol. 3, no. 1, pp. 2–15, Jan. 2010.
[11] S. C. Pokress and J. J. D. Veiga, ''MIT app inventor: Enabling per- sonal mobile computing,'' 2013, arXiv:1310.2830. [Online]. Available: https://arxiv.org/abs/1310.2830
[12] Z. Tang, S. Guo, P. Li, T. Miyazhaki, H. Jin, and X. Liao, ''Energy-efficient transmission scheduling in mobile phones using machine learning and participatory sensing,'' IEEE Trans. Veh. Technol., vol. 64, no. 7, pp. 3167–3176, Jul. 2015.
[13] Y. Zhang, J.-L. Chen, and B. Cheng, ''Integrating events into SOA for IoTservices,'' IEEE Commun. Mag., vol. 55, no. 9, pp. 180–186, Sep. 2017.
[14] Y. Cheng, S. Zhao, B. Cheng, S. Hou, X. Zhang, and J. Chen, ''A distributed event-centric collaborative workflows development system for IoT application,'' in Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS), Jun. 2017.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY