INFOLINE

VOLUME XIV ISSUE III January 2024

DEPARTMENT OF COMPUTER TECHNOLOGY AND INFORMATION TECHNOLOGY

KONGU ARTS AND SCIENCE COLLEGE
(Autonomous)
Affiliated to Bharathiar University, Coimbatore
Accredited with A+ Grade - 3.49 CGPA by NAAC
NANJANAPURAM, ERODE - 638 107

INFOLINE

EDITORIAL BOARD

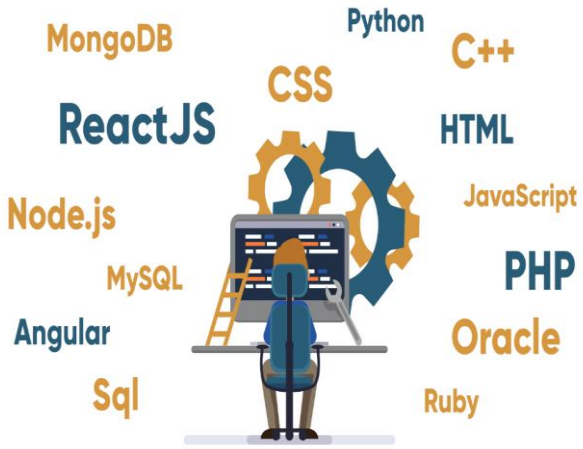
CONTENTS

FULL STACK DEVELOPMENT

Full Stack Development refers to the development of both front end (client side) and back end (server side) portions of web application.

Full stack web Developers

Full stack web developers have the ability to design complete web applications and websites. They work on the frontend, backend, database and debugging of web applications or websites.



Front-end Development

It is the visible part of website or web application which is responsible for user experience. The user directly interacts with the front end portion of the web application or website.

Front-end Technologies

- **HTML:** HTML stands for Hyper Text Markup Language. It is used to design the front end portion of web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text documentation within tag which defines the structure of web pages.
- **CSS:** Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.
- **JavaScript:** JavaScript is a famous scripting language used to create the magic on the sites to make the site interactive for the user. It is used to enhancing the functionality of a website to running cool games and web-based software.

Front End Libraries and Frameworks

- **AngularJS:** AngularJs is a JavaScript open source front-end framework that is mainly used to develop single page web applications(SPAs). It is a continuously growing and expanding framework which provides better ways for developing web applications. It changes the static HTML to dynamic HTML. It is an open source project which can be freely used and changed by anyone. It extends HTML attributes with Directives, and data is bound with HTML.
- **React.js:** React is a declarative, efficient, and flexible JavaScript library for building user interfaces. ReactJS is an open-source, component-based front end library responsible only for the view layer of the application. It is maintained by Facebook.

- **Bootstrap:** Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites.
- **jQuery:** jQuery is an open source JavaScript library that simplifies the interactions between an HTML/CSS document, or more precisely the Document Object Model (DOM), and JavaScript. Elaborating the terms, jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.
- **SASS:** It is the most reliable, mature and robust CSS extension language. It is used to extend the functionality of an existing CSS of a site including everything from variables, inheritance, and nesting with ease.
- Some other libraries and frameworks are: Semantic-UI, Foundation, Materialize, Backbone.js, Express.js, Ember.js etc.

Back-end Technologies

It refers to the server-side development of web application or website with a primary focus on how the website works. It is responsible for managing the database through queries and APIs by client-side commands. This type of website mainly consists of three parts front end, back end, and database.

The back end portion is built by using some libraries, frameworks, and languages which are discussed below:

- **PHP:** PHP is a server-side scripting language designed specifically for web development. Since, PHP code executed on server side so it is called server side scripting language.
- **C++ :** It is a general purpose programming language and widely used now a days for competitive programming. It is also used as backend language.
- **Java:** Java is one of the most popular and widely used programming language and platform. It is highly scalable. Java components are easily available.
- **Python:** Python is a programming language that lets you work quickly and integrate systems more efficiently.
- **Node.js:** Node.js is an open source and cross-platform runtime environment for executing JavaScript code outside of a browser. You need to remember that NodeJS is not a framework and it's not a programming language. Most of the people are confused and understand it's a framework or a programming language. We often use Node.js for building back-end services like APIs like Web App or Mobile App. It's used in production by large companies such as Paypal, Uber, Netflix, Walmart and so on.

Back End Frameworks

The list of back end frameworks are:

- Express
- Django

- Rails
- Laravel, Spring etc.

The other back end program/scripting languages are: C#, Ruby, REST, GO etc.

P.S. MOHANKUMAR

III B.Sc. (Computer Technology)

TAILWIND CSS

Tailwind CSS can be used to style websites in the fastest and easiest way. Tailwind CSS is basically a utility-first CSS framework for rapidly building custom user interfaces. It is a highly customizable, low-level CSS framework that gives you all of the building blocks you need to build bespoke designs without any annoying opinionated styles you have to fight to override.

The beauty of this thing called tailwind is it doesn't impose design specifications or how your site should look, you simply bring tiny components together to construct a user interface that is unique. What Tailwind simply does is take a 'raw' CSS file, process this CSS file over a configuration file, and produce an output.

Why Tailwind CSS?

- The faster UI building process.
- It is a utility-first CSS framework which means we can use utility classes to build custom designs without writing CSS as in the traditional approach.

Advantages of Tailwind CSS

- No more names for CSS classes and Id.
- Minimum lines of Code in CSS file.
- We can customize the designs to make the components.
- Makes the website responsive.
- Makes the changes in the desired manner. CSS is global in nature and if make changes in the file the property is changed in all the HTML files linked to it. But with the help of Tailwind CSS, we can use utility classes and make local changes.

Supported Browser

- Google Chrome
- Microsoft Edge
- Firefox
- Safari

K.Bharathkumar

II B.Sc. (Information Technology)

MOMENT.JS

Moment.js is a JavaScript package that makes it simple to parse, validate, manipulate, and display date/time in JavaScript. It allows you to display dates in a human-readable way based on your location. It can be used in a browser using the script approach. Moment.js is also compatible with Node.js and can be installed via npm. MomentJS is a JavaScript library which helps is parsing, validating, manipulating and displaying date/time in JavaScript in a very easy way. This chapter will provide an overview of MomentJS and discusses its features in detail.

Moment JS allows displaying of date as per localization and in human readable format.

You can use MomentJS inside a browser using the script method. It is also available with Node.js and can be installed using npm. In MomentJS, you can find many easy to use methods to add, subtract, validate date, get the maximum, minimum date etc. It is an open source project and you can easily contribute to the library and add features in the form of plugins and make it available on GitHub and in Node.js.

Parsing

Parsing allows you to parse the date in the format required. Parsing of date is available in string, object and array. It allows you to clone the moment using moment. clone. There are methods available which gives the date output in UTC format.

Date Validation

Date Validation is very easy with MomentJS. You can use the method isValid() and check whether the date is valid or not. MomentJS also provides many parsing flags which can be used to check for date validation.

Manipulation

There are various methods to manipulate Date and Time on the moment object. add, subtract, startoftime, endoftime, local, utc, utcoffset etc., are the methods available which gives details required on date/time in MomentJS.

Get/Set

Get/Set allows to read and set the units in the date. It allows to change as well as read hour, minute, seconds, millisecond, date of month, day of week, day of year, week of year, month, year, quarter, week year, weeks in year, get/set, maximum, minimum etc. Get /Set is a very helpful feature available in MomentJS.

Display

Display provides formats to display date in different ways. There are methods available which tells the time from a given moment, from the current moment, difference between two moments etc. It allows to display date in JSON format, Array, Object, String etc.

Date Queries

Date Queries has easy to use methods which tells if the date is greater or lesser than the input, in between the dates given, is a leap year, is a moment, is a date etc. It is very useful with date validation.

Durations

Durations is one of the important features in MomentJS. It basically handles length of the time for given units. The **humanize** method available displays date in a human readable format.

Internationalization

Internationalization is yet another important features in MomentJS. You can display Date and Time based on locale. The locale can be applied to a specific moment if required. You will get a minified file from the MomentJS home site which has all the locales. In case you are dealing with a specific locale, you can also add just that locale file and work with it. The names of months,

weeks and days are displayed in the locale specified.

Customization

MomentJS allows customization to the locale created. You can customize month names, month abbreviation, weekday names, weekday abbreviation, long date format, and calendar format for a defined locale as per your requirements.

Utilities

Utilities comes with two methods: normalize units and invalid. They are used with the moment and helps us change or customize the output as we need. It also allows to set our own custom validation on the moment object.

Plugins

Plugins are additional features of MomentJS. There are many plugins added to calendars, date format, parsing, date ranges, precise range etc.

P.Logesh

I B.Sc. (Information Technology)

◆••••••••••••••••••••••••••••••••••◆

LARAVEL

Laravel is a web application framework created by Taylor Otwell in 2011 and like all other modern frameworks, it also follows the Model-View-Controller (MVC) architectural pattern. Laravel values Elegance, Simplicity, and Readability and if one talks of building applications with PHP frameworks, Laravel is second to none. Since Laravel is open-source, one can find the source code in its Github. Laravel is power-packed with its own ready-to-use first-party packages some of which are:

- **Scheduler:** It includes support for scheduling periodically executed tasks.
- **Cashier:** For managing subscription billing services.
- **Flysystem:** Allows remote storage to be used as in the same way as local file systems.
- **Socialite:** Simplified mechanism for OAuth authentication with providers like Facebook, Github, Google, etc.

Laravel is as a web application framework based on a few key points:

- **Language Support:** PHP Version -5.5.9
- **MVC Framework:** Yes (from Laravel 2 onwards).
- **Object Relational Mapping:** Needed to enforce constraints on the relationship between database objects.
- **Testing:** Unit Testing is provided as an integral part of Laravel that prevents regressions in the framework. PHP Unit
- **DataBase Migration:** It helps in simplifying the deployment and updating of applications.
- **Security:** SSH (Secure Shell) is used as an encrypted network protocol for execution of CLI (command-line interface) commands.
- **Caching:** available

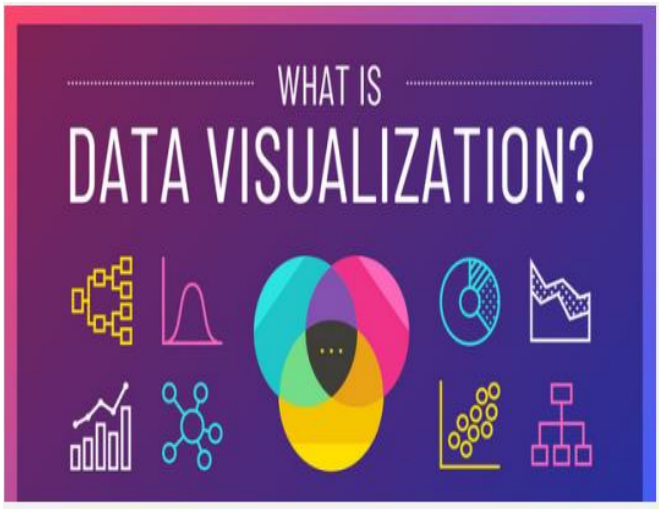
- **Form Validation:** Event listeners are bind internally which invokes the form validation methods and thus the actual form is generated.
- **Scaffolding:** In Laravel, the programmer can specify how the application database may be used.
- **Rapid Application Development:** available
- **Mobility**: available

N.Lavanya

II B.Sc. (Information Technology)

DATA VISUALIZATION AND POWER BUSINESS INTELLIGENCE

Every day a huge amount of data is generated. This data can even vary in nature and structure. A business, for example, can have data on sales revenue, marketing performance, customer interactions, inventory levels, production metrics, staffing levels, costs, etc. But with so much data to sift through, it can be difficult for people to see the story it tells. Data visualization helps you turn all that granular data into easily understood, visually compelling and useful business information. Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. In the world of Big Data, data visualization tools and technologies are essential to analyse massive amounts of information and make data-driven decisions. Hidden within your data lie important insights that can help drive the business forward. But the challenge is that you can't always connect the dots by looking at raw numbers alone. When you look at your data presented in a visual format, patterns, connections, and other insights emerge that would otherwise remain out of sight.



Our eyes are drawn to colours and patterns. We can quickly identify red from blue, and a square from a circle. Our culture is visual, including everything from art and advertisements to TV and movies. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When we see a chart, we quickly see trends and outliers. If we can see something, we internalize it quickly. It's storytelling with a purpose. If you've ever stared at a massive spreadsheet of data and couldn't see a trend, you know how much more effective a visualization can be. It's hard to think of a professional industry that doesn't benefit from making data more understandable. Every STEM field benefits from understanding data and so do

fields in government, finance, marketing, history, consumer goods, service industries, education, sports, and so on. While we always increasing talk about data visualization there are practical, real-life applications that are undeniable. And, since visualization is so prolific, it's also one of the most useful professional skills to develop. The better you can convey your points visually, whether in a dashboard or a slide deck, the better you can leverage that information. Skill sets are changing to accommodate a data-driven world. It is increasingly valuable for professionals to be able to use data to make decisions and use visuals to tell stories of when data informs the who, what, when, where, and how. While traditional education typically draws a distinct line between creative storytelling and technical analysis, the modern professional world also values those who can cross between the two. Today, data visualization tools run the gamut from free versions you can access with a browser to feature-rich platforms that integrate with a wide variety of mainstream business applications. One such tool is Power BI, an interactive data visualization software product developed by Microsoft with a primary focus on business intelligence (BI). Power BI offers cloud-based services for interactive visualizations with a simple interface for end-users to create their own reports and dashboards.

Power BI was first conceptualized by Ruler and Dhers Netz of the SQL server coverage services team at Microsoft. It was further designed by West Chadic George in the year 2010 and named Project Crescent. In 2011, It was bundled with SQL Server Codenamed Mount McKinley. Microsoft unveiled the first preview to Power BI in September 2014. And finally, the first version of Power BI was released on 24 July 2015. It was based on Excel Based Add-ins like Power Query, Pivot, view, and Map. Today Power BI comes across as one of the most powerful and efficient data visualization and analytical tool. Some of the many advantages it offers include pre-built dashboards, real-time updates, secure and reliable connection to your data sources in the cloud or on-premises, integration with both Python and R coding, etc. Moreover, it is backed by artificial intelligence and machine learning. This tool, however, currently has some disadvantages in terms of sharing the reports made and certain types not being compatible with it. These are likely to be overcome in the future as Power BI is further developed.

S.Kaviya

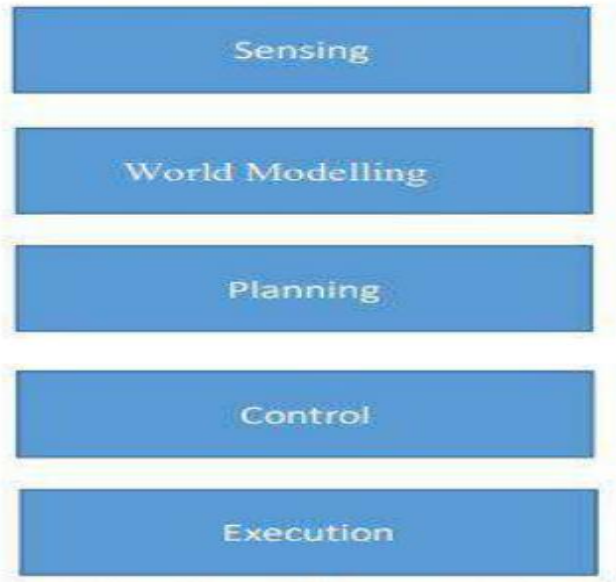
III B.Sc. (Computer Technology)

SPACE ROBOTICS

Space robotics is the development of general purpose machines that are capable of surviving in the space environment, performing exploration, construction, maintenance, servicing or other tasks. Humans control space robots from either a "local" control console or "remotely" controlled from human operators on Earth. Space robots are generally designed to do multiple tasks.

Space Research: “SPACE”, the word itself signifies something infinite. Space travel has always been dangerous and any unexpected event can cause death. It is here that the robots play a huge role and help mankind in his research process. How Robots Work in Space? Working principle of Space robots are based on the SPA algorithm. SPA stands for sense, plan and action. It is used in built world modules to match and worked accordingly.

Flowchart



Technologies Used

Mapping and navigation One of the basic functions of a space robot is to navigate its way cleverly through all obstacles that come in its way. Mapping and navigation comprise of three more technologies.

1.Obstacle avoidance

2.Mapping

3.Path planning

4.Planning: It is a feature by which a robot understands the situation and

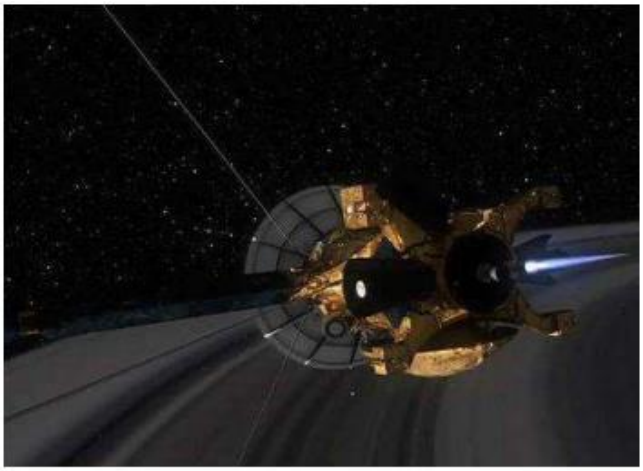
5.decides a strategy to tackle it.

6.Sequencing: Selection of a particular skill set which would result in perfect execution of a plan.

7.Control: Performing the selected skill set to perfection.

Types of Space Robots

1. **Planetary Rovers:** It is the most advanced form of robotics technology used in space research. They are the robots, which explore, navigate and research themselves with the least human intervention; they analyze the data collected and send the results back to earth.



2.**IN-Orbit Operators:** They are the robots, which assist an astronaut during his space mission. For example, a robot can be designed specially to

refuel a shuttle thus helping the astronaut to remain in his shuttle and accomplish various tasks without any risk to their lives.

3.**Probes:** A similar class of robots explores the system without actually physically landing anywhere. These typically use cameras and variety of instruments to measure other planets, moons, and the sun from distance. Most of these use solar cells to their instruments.

4.**Astronaut Assistance**: Besides acting as explorers, space robots can also assist astronauts in manned spaceflight. One of the most notable a device known as the Canadarm. with funding from the Canadian Space the Candarm became a permanent fixture many American space shuttles and the international space station.

M.Harini

II B.Sc. (Computer Technology)

EXPLAINABLE ARTIFICIAL INTELLIGENCE

Explainable AI (XAI), often overlapping with Interpretable AI, or Explainable Machine Learning (XML), either refers to an AI system over which it is possible for humans to retain intellectual oversight, or to the methods to achieve this.[1] The main focus is usually on the reasoning behind the decisions or predictions made by the AI which are made more understandable and transparent. XAI counters the "black box" tendency of machine learning, where even the AI's designers cannot explain why it arrived at a specific decision.

XAI hopes to help users of AI-powered systems perform more effectively by improving their understanding of how those systems reason. XAI may be an implementation of the social right to explanation. Even if there is no such legal right or regulatory requirement, XAI can improve the user experience of a product or service by helping end users trust that the AI is making good decisions. XAI aims to explain what has been done, what is being done, and what will be done next, and to unveil which information these actions are based on.This makes it possible to confirm existing knowledge, challenge existing knowledge, and generate new assumptions.

Machine learning (ML) algorithms used in AI can be categorized as white-box or black-box. White-box models provide results that are understandable to experts in the domain. Black-box models, on the other hand, are extremely hard to explain and can hardly be understood even by domain experts. XAI algorithms follow the three principles of transparency, interpretability, and explainability. A model is transparent “if the processes that extract model parameters from training data and generate labels from testing data can be described and motivated by the approach designer.” Interpretability describes the possibility of comprehending the ML model and presenting the underlying basis for decision-making in a way that is understandable to humans. Explainability is a concept that is recognized as important, but a consensus definition is not available. One possibility is “the collection of features of the interpretable domain that have contributed, for a

given example, to producing a decision (e.g., classification or regression)" If algorithms fulfill these principles, they provide a basis for justifying decisions, tracking them and thereby verifying them, improving the algorithms, and exploring new facts.

Sometimes it is also possible to achieve a high-accuracy result with white-box ML algorithms. These algorithms have an interpretable structure that can be used to explain predictions. Concept Bottleneck Models, which use concept-level abstractions to explain model reasoning, are examples of this and can be applied in both image and text prediction tasks. This is especially important in domains like medicine, defense, finance, and law, where it is crucial to understand decisions and build trust in the algorithms. Many researchers argue that, at least for supervised machine learning, the way forward is symbolic regression, where the algorithm searches the space of mathematical expressions to find the model that best fits a given dataset.

AI systems optimize behaviour to satisfy a mathematically specified goal system chosen by the system designers, such as the command "maximize the accuracy of assessing how positive film reviews are in the test dataset." The AI may learn useful general rules from the test set, such as "reviews containing the word "horrible" are likely to be negative." However, it may also learn inappropriate rules, such as "reviews containing 'Daniel Day-Lewis' are usually positive"; such rules may be undesirable if they are likely to fail to generalize outside the training set, or if people consider the rule to be "cheating" or "unfair." A human can audit rules in an XAI to get an idea of how likely the system is to generalize to future real-world data outside the test set.

G.Aakash
III B.Sc. (Information Technology)

TECHNOLOGICAL CONVERGENCE

Technological convergence is a term that describes bringing previously unrelated technologies together, often in a single device. Smartphones might be the best possible example of such a convergence. Prior to the widespread adoption of smartphones, consumers generally relied on a collection of single-purpose devices. Some of these devices included telephones, wrist watches, digital cameras and global positioning system (GPS) navigators. Today, even low-end smartphones combine the functionality of all these separate devices, easily replacing them in a single device.

Why is technological convergence important?

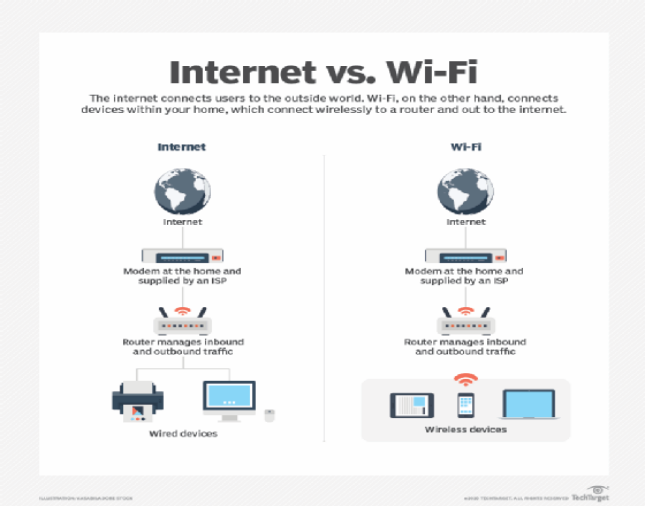
From a consumer perspective, technological convergence is often synonymous with innovation. Technologies rarely converge in their current form. Improvements are often introduced as a part of the convergence. Consider the popularity of video cameras a generation ago. Today, consumer-grade video

cameras are almost nonexistent. Most people record videos on their mobile devices. Although they can eliminate the need to carry a separate device -- a video camera, in this case they also deliver superior video quality as compared to what was once available. Video cameras from the early 2000s, usually, had a maximum resolution of 480i (720 x 480) and, often, experienced poor battery life. In contrast, a modern mobile device can record in 4K resolution. Additionally, their batteries can sometimes last for days, depending on how the device is being used.

Another reason why technological convergence is important from a consumer perspective is because it results in easier access to technology at a lower cost. This trend is sometimes referred to as the consumerization of information technology (IT).

Wi-Fi is a perfect example of this. Wi-Fi was available in the late 1990s, but at the time a Wi-Fi router cost thousands of dollars and, typically, could only be set up by an IT pro. Today, Wi-Fi is ubiquitous. Nearly every consumer electronic device is Wi-Fi enabled. Additionally, prices are far lower than they once were, and Wi-Fi routers have been greatly simplified to the point that a non-tech-savvy person can set them up. Such benefits stem directly from technological convergence.

Businesses, technological convergence means companies are more easily able to connect to their customers and to learn more about customer's buying habits. In some cases, technological convergence even makes it possible for a business to influence a customer's purchases. Some retailers track customer's smartphone locations. If a customer is standing in a particular area of the store for a certain amount of time, the retailer might send the customer a coupon via text message or pop-up notification for the item they're looking at, thus, further enticing the customer to make a purchase.



Types of technological convergence

A good way to evaluate the importance of technological convergence is to consider innovations from previous generations. Items such as CD players, cassette decks, console TVs or corded telephones served only one function, whereas a single modern handheld computing device can meld several of those functions, with hardly any user intervention required. People who aren't computer-literate are more likely to embrace the internet and video on demand if they can access these technologies through their television. TV is familiar and nonthreatening. Displays are

large and TVs are easy to operate. Using them to access the web requires almost no training.

PCs, in spite of their graphical user interfaces, tend to be more text oriented. They are interactive, geared toward business and education uses, and their displays are smaller. Computers can be challenging for some and, often, require formal education or come with a personal learning curve.

Using a smartphone to make calls and take digital photos and using your digital TV to perform computing tasks, such as surfing the web while watching a movie, are two more examples of technological convergence.

M.S.K Manassha

I B.Sc. (Information Technology)

◆••••••••••••••••••••••••••••••••••◆

META PROGRAMMING

Meta programming is a programming technique in which computer programs have the ability to treat other programs as their data. It means that a program can be designed to read, generate, analyse or transform other programs, and even modify itself while running. In some cases, this allows programmers to minimize the number of lines of code to express a solution, in turn reducing development time. It also allows programs a greater flexibility to efficiently handle new situations without recompilation. Metaprogramming can be used to move computations from run-time to compile-time, to generate code using compile time computations, and to enable self-modifying code. The ability of a programming language to be its own metalanguage is called reflection. Reflection is a valuable language feature to facilitate metaprogramming.

Metaprogramming was popular in the 1970s and 1980s using list processing languages such as LISP. LISP hardware machines were popular in the 1980s and enabled applications that could process code. They were frequently used for artificial intelligence applications. Metaprogramming enables developers to write programs and develop code that falls under the generic programming paradigm. Having the programming language itself as a first-class data type (as in Lisp, Prolog, SNOBOL, or Rebol) is also very useful; this is known as homoiconicity. Generic programming invokes a metaprogramming facility within a language by allowing one to write code without the concern of specifying data types since they can be supplied as parameters when used.

Metaprogramming usually works in one of three ways.

1. The first approach is to expose the internals of the run-time engine to the programming code through application programming interfaces (APIs) like that for the .NET IL emitter.
2. The second approach is dynamic execution of expressions that contain programming commands, often composed

from strings, but can also be from other methods using arguments or context, like JavaScript.[6] Thus, "programs can write programs." Although both approaches can be used in the same language, most languages tend to lean toward one or the other.

3. The third approach is to step outside the language entirely. General purpose program transformation systems such as compilers, which accept language descriptions and carry out arbitrary transformations on those languages, are direct implementations of general metaprogramming. This allows metaprogramming to be applied to virtually any target language without regard to whether that target language has any metaprogramming abilities of its own. One can see this at work with Scheme and how it allows tackling some limitations faced in C by using constructs that were part of the Scheme language itself to extend C.

Lisp is probably the quintessential language with metaprogramming facilities, both because of its historical precedence and because of the simplicity and power of its metaprogramming. In Lisp metaprogramming, the unquote operator (typically a comma) introduces code that is evaluated at program definition time rather than at run time; see Self-evaluating forms and quoting in Lisp. The metaprogramming language is thus identical to the host programming language, and existing Lisp routines can be directly reused for metaprogramming, if desired. This approach has been implemented in other languages by incorporating an interpreter in the program, which works directly with the program's data.

V.B Krishna Prabu

I B.Sc. (Computer Technology)

RACKET (PROGRAMMING LANGUAGE)

Racket is a general-purpose, multi-paradigm programming language and a multi-platform distribution that includes the Racket language, compiler, large standard library, IDE, development tools, and a set of additional languages including Typed Racket (a sister language of Racket with a static type-checker), Swindle, FrTime, Lazy Racket, R5RS & R6RS Scheme, Scribble, Datalog, Racklog, Algol 60 and several teaching languages. The Racket language is a modern dialect of Lisp and a descendant of Scheme. It is designed as a platform for programming language design and implementation. In addition to the core Racket language, Racket is also used to refer to the family of programming languages and set of tools supporting development on and with Racket. Racket is also used for scripting, computer science education, and research.

The Racket platform provides an implementation of the Racket language (including a runtime system, libraries, and compiler supporting several

compilation modes: machine code, machine-independent, interpreted, and JIT) along with the DrRacket integrated development environment (IDE) written in Racket. Racket is used by the ProgramByDesign outreach program, which aims to turn computer science into "an indispensable part of the liberal arts curriculum".

The core Racket language is known for its extensive macro system which enables creating embedded and domain-specific languages, language constructs such as classes or modules, and separate dialects of Racket with different semantics. The platform distribution is free and open-source software distributed under the Apache 2.0 and MIT licenses. Extensions and packages written by the community may be uploaded to Racket's package catalog.

Features

Racket's core language includes macros, modules, lexical closures, tail calls, delimited continuations, parameters (fluid variables), software contracts, green threads and OS threads, and more. The language also comes with primitives, such as event spaces and custodians, which control resource management and enables the language to act like an operating system for loading and managing other programs. Further extensions to the language are created with the powerful macro system, which together with the module system and custom parsers can control all aspects of a language. Most language constructs in Racket are implemented as macros in the base language. These include a mixin class system, a component (or module) system as expressive as opaque ascription in the ML module system, and pattern matching.

Further, the language features the first contract system for a higher-order programming language.[44] Racket's contract system is inspired by the Design by Contract work for Eiffel and extends it to work for higher-order values such as first-class functions, objects, reference cells, and so on. For example, an object that is checked by a contract can be ensured to make contract checks when its methods are eventually invoked. Racket includes both bytecode and JIT (JIT) compilers. The bytecode compiler produces an internal bytecode format run by the Racket virtual machine, and the JIT compiler translates bytecode to machine code at runtime.

Since 2004, the language has also shipped with PLaneT, a package manager that is integrated into the module system so that third-party libraries can be transparently imported and used. Also, PLaneT has a built-in versioning policy to prevent dependency hell. At the end of 2014, much of Racket's code was moved into a new packaging system separate from the main code base. This new packaging system is serviced by a client program named raco. The new package system provides fewer features than PLaneT; a blog post by Jay McCarthy on the Racket blog explains the rationale for the change and how to duplicate the older system.

Programming Environment

The language platform provides a self-hosted IDE named DrRacket, a continuation-based web server, a graphical user interface, and other tools. As a viable scripting tool with libraries like common scripting languages, it can be used for scripting the Unix shell. It can parse command-line arguments and execute external tools.

DrRacket IDE

DrRacket (formerly DrScheme) is widely used among introductory computer science courses that teach Scheme or Racket and is lauded for its simplicity and appeal to beginner programmers. The IDE was originally built for use with the TeachScheme! project (now ProgramByDesign), an outreach effort by Northeastern University and a number of affiliated universities for attracting high school students to computer science courses at the college level.

The editor provides highlighting for syntax and run-time errors, parenthesis matching, a debugger and an algebraic stepper. Its student-friendly features include support for multiple "language levels" (Beginning Student, Intermediate Student and so on). It also has integrated library support, and sophisticated analysis tools for advanced programmers. Further, module-oriented programming is supported with the module browser, a contour view, integrated testing and coverage measurements, and refactoring support. It provides integrated, context-sensitive access to an extensive hyper-linked help system named "Help Desk". DrRacket is available for Windows, macOS, Unix, and Linux with the X Window System and programs behave similarly on all these platforms.

Implementation

Racket currently has two implementations. Both support Linux, Windows and MacOS on a variety of architectures and are supported as at version 8.8 (2023). The default implementation uses the Chez Scheme incremental compiler and runtime. The alternate implementation generates platform-independent bytecode and uses Just-in-time compilation to generate machine code as it is loaded.

In addition, there are experimental implementations:

- RacketScript is an experimental Racket to JavaScript (ES6) compiler. It allows programmers to use both JavaScript's and Racket's ecosystem and aims to make this interoperability as smooth as possible.
- Pycket is a Racket implementation generated using the RPython framework.

Applications and Practical use

Apart from having a basis in programming language theory, Racket was designed as a general-purpose language for production systems. Thus, the Racket distribution features an extensive library that covers systems and network

programming, web development, a uniform interface to the underlying operating system, a dynamic foreign function interface, several flavours of regular expressions, lexer/parser generators,[58] logic programming, and a complete GUI framework. Racket has several features useful for a commercial language, among them an ability to compile standalone executables under Windows, macOS, and Unix, a profiler and debugger included in the integrated development environment (IDE), and a unit testing framework.

Racket has been used for commercial projects and web applications. A notable example is the Hacker News website, which runs on Arc, which is developed in Racket. Naughty Dog has used it as a scripting language in several video games. Racket is used to teach students algebra through game design in the Bootstrap program.

M.Harini

I B.Sc. (Computer Technology)

INTRODUCTION TO NOSQL

NoSQL is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructured and semi-structured data. Unlike traditional relational databases that use tables with pre-defined schemas to store data, NoSQL databases use flexible data models that can adapt to changes in data structures and are capable of scaling horizontally to handle growing amounts of data.

The term NoSQL originally referred to "non-SQL" or "non-relational" databases, but the term has since evolved to mean "not only SQL," as NoSQL databases have expanded to include a wide range of different database architectures and data models.

NoSQL databases are generally classified into four main categories:

1. **Document databases:** These databases store data as semi-structured documents, such as JSON or XML, and can be queried using document-oriented query languages.
2. **Key-value stores:** These databases store data as key-value pairs, and are optimized for simple and fast read/write operations.
3. **Column-family stores:** These databases store data as column families, which are sets of columns that are treated as a single entity. They are optimized for fast and efficient querying of large amounts of data.
4. **Graph databases:** These databases store data as nodes and edges, and are designed to handle complex relationships between data.

NoSQL databases are often used in applications where there is a high volume of data that needs to be processed and analyzed in real-time, such as social media analytics, e-commerce, and gaming. They can also be used for other applications, such as content management systems, document management, and customer relationship management.

However, NoSQL databases may not be suitable for all applications, as they may not provide the same level of data consistency and transactional

guarantees as traditional relational databases. It is important to carefully evaluate the specific needs of an application when choosing a database management system.

NoSQL originally referring to non SQL or non relational is a database that provides a mechanism for storage and retrieval of data. This data is modeled in means other than the tabular relations used in relational databases. Such databases came into existence in the late 1960s**,** but did not obtain the NoSQL moniker until a surge of popularity in the early twenty-first century. NoSQL databases are used in real-time web applications and big data and their use are increasing over time.

- NoSQL systems are also sometimes called Not only SQL to emphasize the fact that they may support SQL-like query languages. A NoSQL database includes simplicity of design, simpler horizontal scaling to clusters of machines,**has** and finer control over availability. The data structures used by NoSQL databases are different from those used by default in relational databases which makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it should solve.
- NoSQL databases, also known as "not only SQL" databases, are a new type of database management system that has**,** gained popularity in recent years. Unlike traditional relational databases, NoSQL databases are designed to handle large amounts of unstructured or semi-structured data, and they can accommodate dynamic changes to the data model. This makes NoSQL databases a good fit for modern web applications, real-time analytics, and big data processing.
- Data structures used by NoSQL databases are sometimes also viewed as more flexible than relational database tables. Many NoSQL stores compromise consistency in favor of availability, speed**,,** and partition tolerance. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages, lack of standardized interfaces, and huge previous investments in existing relational databases.
- Most NoSQL stores lack true ACID (Atomicity, Consistency, Isolation, Durability) transactions but a few databases, such as MarkLogic, Aerospike, FairCom c-treeACE, Google Spanner (though technically a NewSQL database), Symas LMDB, and OrientDB have made them central to their designs.
- Most NoSQL databases offer a concept of eventual consistency in which database changes are propagated to all nodes so queries for data might not return updated data immediately or might result in reading data that is not accurate which is a problem known as stale reads. Also ha**s** some NoSQL systems may exhibit lost writes and other forms of data loss. Some NoSQL systems provide concepts such as write-ahead logging to avoid data loss.

- One simple example of a NoSQL database is a document database. In a document database, data is stored in documents rather than tables. Each document can contain a different set of fields, making it easy to accommodate changing data requirements
- For example, “Take, for instance, a database that holds data regarding employees.”. In a relational database, this information might be stored in tables, with one table for employee information and another table for department information. In a document database, each employee would be stored as a separate document, with all of their information contained within the document.
- NoSQL databases are a relatively new type of database management system that has**a** gained popularity in recent years due to their scalability and flexibility. They are designed to handle large amounts of unstructured or semi-structured data and can handle dynamic changes to the data model. This makes NoSQL databases a good fit for modern web applications, real-time analytics, and big data processing.

Key Features of NoSQL

1. **Dynamic schema:** NoSQL databases do not have a fixed schema and can accommodate changing data structures without the need for migrations or schema alterations.
2. **Horizontal scalability:** NoSQL databases are designed to scale out by adding more nodes to a database cluster, making them well-suited for handling large amounts of data and high levels of traffic.
3. **Document-based:** Some NoSQL databases, such as MongoDB, use a document-based data model, where data is stored in a **scales**semi-structured format, such as JSON or BSON.
4. **Key-value-based:** Other NoSQL databases, such as Redis, use a key-value data model, where data is stored as a collection of key-value pairs.
5. **Column-based:** Some NoSQL databases, such as Cassandra, use a column-based data model, where data is organized into columns instead of rows.
6. **Distributed and high availability:** NoSQL databases are often designed to be highly available and to automatically handle node failures and data replication across multiple nodes in a database cluster.
7. **Flexibility:** NoSQL databases allow developers to store and retrieve data in a flexible and dynamic manner, with support for multiple data types and changing data structures.
8. **Performance:** NoSQL databases are optimized for high performance and can handle a high volume of reads and writes, making them suitable for big data and real-time applications.

Advantages of NoSQL: There are many advantages of working with NoSQL databases such as MongoDB and Cassandra. The main advantages are high scalability and high availability.

1. High scalability: NoSQL databases use sharing for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved is sharing. Vertical scaling means adding more resources to the existing machine whereas horizontal scaling means adding more machines to handle the data. Vertical scaling is not that easy to implement but horizontal scaling is easy to implement. Examples of horizontal scaling databases are MongoDB, Cassandra, etc.

2. **Flexibility:** NoSQL databases are designed to handle unstructured or semi-structured data, which means that they can accommodate dynamic changes to the data model. This makes NoSQL databases a good fit for applications that need to handle changing data requirements.

3. High availability: The auto**,** replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.

4. **Scalability:** NoSQL databases are highly scalable, which means that they can handle large amounts of data and traffic with ease. This makes them a good fit for applications that need to handle large amounts of data or traffic

5. **Performance:** NoSQL databases are designed to handle large amounts of data and traffic, which means that they can offer improved performance compared to traditional relational databases.

6. **Cost-effectiveness:** NoSQL databases are often more cost-effective than traditional relational databases, as they are typically less complex and do not require expensive hardware or software.

7. **Agility:** Ideal for agile development.

Disadvantages of NoSQL: NoSQL has the following disadvantages.

1. **Lack of standardization:** There are many different types of NoSQL databases, each with its own unique strengths and weaknesses. This lack of standardization can make it difficult to choose the right database for a specific application

2. **Lack of ACID compliance:** NoSQL databases are not fully ACID-compliant, which means that they do not guarantee the consistency, integrity, and durability of data. This can be a drawback for applications that require strong data consistency guarantees.

3. **Narrow focus:** NoSQL databases have a very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the field of Transaction Management than NoSQL.

4. **Open-source:** NoSQL is an open-source database. There is no reliable standard for NoSQL yet. In other words, two database systems are likely to be unequal.

5. **Lack of support for complex queries:** NoSQL databases are not designed to handle complex queries, which means that they are not a good fit for applications that require complex data analysis or reporting.
6. **Lack of maturity:** NoSQL databases are relatively new and lack the maturity of traditional relational databases. This can make them less reliable and less secure than traditional databases.
7. **Management challenge:** The purpose of big data tools is to make the management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than in a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.
8. **GUI is not available:** GUI mode tools to access the database are not flexibly available in the market.
9. **Backup:** Backup is a great weak point for some NoSQL databases like MongoDB. MongoDB has no approach for the backup of data in a consistent manner.
10. **Large document size:** Some database systems like MongoDB and CouchDB store data in JSON format. This means that documents are quite large (BigData, network bandwidth, speed), and having descriptive key names actually hurts since they increase the document size.

Types of NoSQL database: Types of NoSQL databases and the name of the database system that falls in that category are:

1. **Graph Databases**:
   Examples: Amazon Neptune, Neo4j
2. **Key value store:**
   Examples: Memcached, Redis, Coherence
3. **Column:**
   Examples: Hbase, Big Table, Accumulo
4. **Document-based:**
   Examples: MongoDB, CouchDB, Cloudant

K.Sathya
III B.Sc. (Information Technology)

FUNCTIONAL SEMICONDUCTOR MADE FROM GRAPHENE

Researchers at the Georgia Institute of Technology have created the world's first functional semiconductor made from graphene, a single sheet of carbon atoms held together by the strongest bonds known. Semiconductors, which are materials that conduct electricity under specific conditions, are foundational components of electronic devices. The team's breakthrough throws open the door to a new way of doing electronics.

Their discovery comes at a time when silicon, the material from which nearly all modern electronics are made, is reaching its limit in the face of increasingly faster computing and smaller electronic devices. Walter de Heer, Regents' Professor of physics at Georgia Tech, led a team

of researchers based in Atlanta, Georgia, and Tianjin, China, to produce a graphene semiconductor that is compatible with conventional microelectronics processing methods a necessity for any viable alternative to silicon.

In this latest research, published in Nature, de Heer and his team overcame the paramount hurdle that has been plaguing graphene research for decades, and the reason why many thought graphene electronics would never work. Known as the "band gap," it is a crucial electronic property that allows semiconductors to switch on and off. Graphene didn't have a band gap until now.

"We now have an extremely robust graphene semiconductor with 10 times the mobility of silicon, and which also has unique properties not available in silicon," de Heer said. "But the story of our work for the past 10 years has been, 'Can we get this material to be good enough to work?'"

A New Type of Semiconductor

De Heer started to explore carbon-based materials as potential semiconductors early in his career, and then made the switch to exploring two-dimensional graphene in 2001. He knew then that graphene had potential for electronics.

"We were motivated by the hope of introducing three special properties of graphene into electronics," he said. "It's an extremely robust material, one that can handle very large currents, and can do so without heating up and falling apart."

De Heer achieved a breakthrough when he and his team figured out how to grow graphene on silicon carbide wafers using special furnaces. They produced epitaxial graphene, which is a single layer that grows on a crystal face of the silicon carbide. The team found that when it was made properly, the epitaxial graphene chemically bonded to the silicon carbide and started to show semiconducting properties.

Over the next decade, they persisted in perfecting the material at Georgia Tech and later in collaboration with colleagues at the Tianjin International Canter for Nanoparticles and Nano systems at Tianjin University in China. De Heer founded the centre in 2014 with Lei Ma, the canter’s director and a co-author of the paper.

How They Did It

In its natural form, graphene is neither a semiconductor nor a metal, but a semimetal. A band gap is a material that can be turned on and off when an electric field is applied to it, which is how all transistors and silicon electronics work. The major question in graphene electronics research was how to switch it on and off so it can work like silicon.

But to make a functional transistor, a semiconducting material must be greatly manipulated, which can damage its properties. To prove that their platform could function as a viable semiconductor, the team needed to measure its electronic properties without damaging it.

They put atoms on the graphene that "donate" electrons to the system -- a technique

called doping, used to see whether the material was a good conductor. It worked without damaging the material or its properties.

The team's measurements showed that their graphene semiconductor has 10 times greater mobility than silicon. In other words, the electrons move with very low resistance, which, in electronics, translates to faster computing. "It's like driving on a gravel road versus driving on a freeway," de Heer said. "It's more efficient, it doesn't heat up as much, and it allows for higher speeds so that the electrons can move faster."

The team's product is currently the only two-dimensional semiconductor that has all the necessary properties to be used in nanoelectronics, and its electrical properties are far superior to any other 2D semiconductors currently in development.

"A long-standing problem in graphene electronics is that graphene didn't have the right band gap and couldn't switch on and off at the correct ratio," said Ma. "Over the years, many have tried to address this with a variety of methods. Our technology achieves the band gap, and is a crucial step in realizing graphene-based electronics."

Moving Forward

Epitaxial graphene could cause a paradigm shift in the field of electronics and allow for completely new technologies that take advantage of its unique properties. The material allows the quantum mechanical wave properties of electrons to be utilized, which is a requirement for quantum computing.

"Our motivation for doing graphene electronics has been there for a long time, and the rest was just making it happen," de Heer said. "We had to learn how to treat the material, how to make it better and better, and finally how to measure the properties. That took a very, very long time."

According to de Heer, it is not unusual to see yet another generation of electronics on its way. Before silicon, there were vacuum tubes, and before that, there were wires and telegraphs. Silicon is one of many steps in the history of electronics, and the next step could be graphene.

"To me, this is like a Wright brother’s moment," de Heer said. "They built a plane that could fly 300 feet through the air. But the skeptics asked why the world would need flight when it already had fast trains and boats.

In 1936, the British mathematician Alan Turing came up with an idea for a universal computer. It was a simple device: an infinite strip of tape covered in zeros and ones, together with a machine that could move back and forth along the tape, changing zeros to ones and vice versa according to some set of rules. He showed that such a device could be used to perform any computation.

Turing did not intend for his idea to be practical for solving problems. Rather, it offered an invaluable way to explore the nature of

computation and its limits. In the decades since that seminal idea, mathematicians have racked up a list of even less practical computing schemes. Games like Minesweeper or Magic: The Gathering could, in principle, be used as general-purpose computers. So could so-called cellular automata like John Conway's Game of Life, a set of rules for evolving black and white squares on a two-dimensional grid.

S.Dinesh

II B.Sc. (Computer Technology)

ORIGAMI COMPUTER

In September 2023, Inna Zakharevich of Cornell University and Thomas Hull of Franklin & Marshall College showed that anything that can be computed can be computed by folding paper.

Zakharevich, a lifelong origami enthusiast, started thinking about this problem in 2021 after stumbling on a video that explained the Turing completeness of the Game of Life. "I was like, origami is a lot more complicated than the Game of Life," Zakharevich said. "If the Game of Life is Turing complete, origami should be Turing complete too."

But this wasn't her area of expertise. Although she'd been folding origami since she was young "if you want to give me a super complex thing that requires a 24-inch sheet of paper and has 400 steps, I'm all over that thing," she said her mathematical research dealt with the much more abstract realms of algebraic topology and category theory. So she emailed Hull, who studied the math of origami full time.

"She just emailed me out of the blue, and I was like, why is an algebraic topologist asking me about this?" Hull said. But he realized he'd never actually thought about whether origami might be Turing complete. So Zakharevich set out to prove that you can make a computer out of origami. First they had to encode computational inputs and outputs as well as basic logical operations like AND and OR as folds of paper. If they could then show that their scheme could simulate some other computational model already known to be Turing complete, they would accomplish their goal.

A logical operation takes in one or more inputs (each one written as TRUE or FALSE) and spits out an output (TRUE or FALSE) based on a given rule. To make an operation out of paper, the mathematicians designed a diagram of lines, called a crease pattern, that specifies where to fold

the paper. A pleat in the paper represents an input. If you fold along one line in the crease pattern, the pleat flips to one side, indicating an input value of TRUE. But if you fold the paper along a different (nearby) line, the pleat flips onto its opposite side, indicating FALSE.

Two of these input pleats feed into a complicated snarl of folds called a gadget. The gadget encodes the logical operation. In order to make all these folds and still get the paper to fold flat a requirement that Hull and Zakharevich impose they included a third pleat that's forced to fold in a particular way. If the pleat flips one way, it means the output is TRUE. If it flips the other way, the output is FALSE.

The mathematicians designed different gadgets that turn inputs into outputs according to various logical operations. "It was a lot of playing around with paper and sending pictures to each other … and then writing rigorous proofs that these things worked the way we said they did," Hull said.

It's been known since the late 1990s that a simpler one-dimensional analogue of Conway's Game of Life is Turing complete. Hull and Zakharevich figured out how to write this version of Life in terms of logical operations. "We ended up only needing to use four gates: AND, OR, NAND and NOR," Zakharevich said, referring to two additional simple gates. But to combine these different gates, they had to build new gadgets that absorbed extraneous signals and allowed other signals to turn and intersect without interfering with each other. "That was the hardest part," Zakharevich said, "figuring out how to make everything line up properly." After she and Hull managed to fit their gadgets together, they could encode everything they needed in paper folds, thereby showing that origami is Turing complete.

An origami computer would be massively inefficient and impractical. But in principle, if you had a very large piece of paper and lots of time on your hands, you could use origami to calculate arbitrarily many digits of π, determine the optimal way to route every delivery driver in the world, or run a program to predict the weather. "In the end, the crease pattern is gargantuan," Hull said. "It's hard to fold, but it gets the job done."

For decades, mathematicians were drawn to origami because "it seemed fun and useless," said Erik Demaine, a computer scientist at the Massachusetts Institute of Technology who has contributed extensively to the mathematics of origami. But recently it has also caught the eye of engineers.

The math of origami has been used to design massive solar panels that can be folded up and transported into space, robots that swim through water to collect environmental data, stents that travel through tiny blood vessels, and more. "Now there's hundreds if not thousands of people using all the origami math and algorithms that

we've developed in the design of new mechanical structures," Demaine said.

V.B Krishna Prabu

I B.Sc. (Computer Technology)

SCALA PROGRAMMING

Scala is a general-purpose, high-level, multi-paradigm programming language. It is a pure object-oriented programming language which also provides the support to the functional programming approach. There is no concept of primitive data as everything is an object in Scala. It is designed to express the general programming patterns in a refined, succinct, and type-safe way. Scala programs can convert to bytecodes and can run on the JVM(Java Virtual Machine). Scala stands for Scalable language. It also provides the Javascript runtimes. Scala is highly influenced by Java and some other programming languages like Lisp, Haskell, Pizza, etc.

Evolution of Scala:

Scala was designed by the Martin Odersky, professor of programming methods at École Polytechnique Fédérale de Lausanne (EPFL) in Switzerland and a German computer scientist. Martin Odersky is also the co-creator of javac (Java Compiler), Generic Java, and EPFL's Funnel programming language. He started to design the Scala in 2001. Scala was first released publicly in 2004 on the Java platform as its first version. In June 2004, Scala was modified for the .Net Framework. Soon it was followed by second version i.e. (v2.0) in 2006. At JavaOne conference in 2012, Scala was awarded as the winner of the ScriptBowl contest. From June 2012, Scala doesn't provide any support for .Net Framework. The latest version of scala is 2.12.6 which released on 27-Apr-2018.

Why Scala?

Scala has many reasons for being popular among programmers. Few of the reasons are :

- **Easy to Start:** Scala is a high level language so it is closer to other popular programming languages like Java, C, C++. Thus it becomes very easy to learn Scala for anyone. For Java programmers, Scala is more easy to learn.
- **Contains best Features:** Scala contains the features of different languages like C, C++, Java, etc. which makes the it more useful, scalable and productive.
- **Close integration with Java:** The source code of the Scala is designed in such a way that its compiler can interpret the Java classes. Also, Its compiler can utilize the frameworks, Java Libraries, and tools etc.

After compilation, Scala programs can run on JVM.

- **Web – Based & Desktop Application Development:** For the web applications it provides the support by compiling to JavaScript. Similarly for desktop applications, it can be compiled to JVM bytecode.
- **Used by Big Companies:** Most of the popular companies like Apple, Twitter, Walmart, Google etc. move their most of codes to Scala from some other languages. reason being it is highly scalable and can be used in backend operations.

M.S.K Manassha

I B.Sc. (Information Technology)

◆••••••••••••••••••••••••••••••••••◆

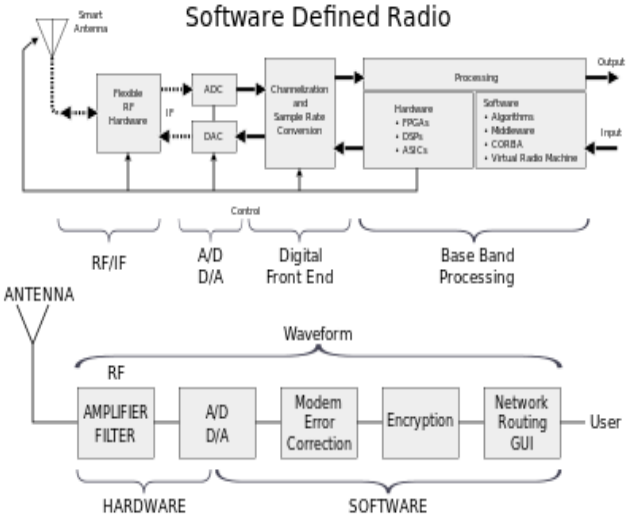
SOFTWARE DEFINED RADIO

Software-defined radio (SDR) is a radio communication system where components that conventionally have been implemented in analog hardware

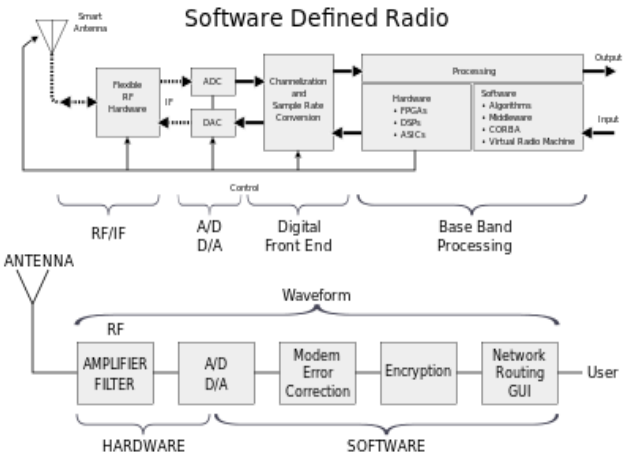
(e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded system.[1] While the concept of SDR is not new, the rapidly evolving capabilities of digital electronics render practical many processes which were once only theoretically possible.

A basic SDR system may consist of a personal computer equipped with a sound card, or other analog-to-digital converter, preceded by some form of RF front end. Significant amounts of signal processing are handed over to the general-purpose processor, rather than being done in special-purpose hardware (electronic circuits). Such a design produces a radio which can receive and transmit widely different radio protocols (sometimes referred to as waveforms) based solely on the software used.

Software radios have significant utility for the military and cell phone services, both of which must serve a wide variety of changing radio protocols in real time. In the long term, software-defined radios are expected by proponents like the Wireless Innovation Forum to become the dominant technology in radio communications. SDRs, along with software defined antennas are the enablers of cognitive radio.



Operating principles



Software defined radio concept

Superheterodyne receivers use a VFO (variable-frequency oscillator), mixer, and filter to tune the desired signal to a common IF (intermediate frequency) or baseband. Typically, in SDR, this signal is then sampled by the analog-to-digital converter. However, in some applications it is not necessary to tune the signal to an intermediate frequency and the radio frequency signal is directly sampled by the analog-to-digital converter (after amplification).

Real analog-to-digital converters lack the dynamic range to pick up sub-microvolt, nanowatt-power radio signals produced by an antenna. Therefore, a low-noise amplifier must precede the conversion step and this device introduces its own problems. For example, if spurious signals are present (which is typical), these compete with the desired signals within the amplifier's dynamic range. They may introduce distortion in the desired signals, or may block them completely. The standard solution is to put band-pass filters between the antenna and the amplifier, but these reduce the radio's flexibility. Real software radios often have two or three analog channel filters with different bandwidths that are switched in and out.

The flexibility of SDR allows for dynamic spectrum usage, alleviating the need to statically assign the scarce spectral resources to a single fixed service.

G.Aakash

III B.Sc. (Information Technology)

THE TECHNOLOGY YOU USE IMPRESSES NO ONE. THE EXPERIENCE YOU CREATE WITH IT IS EVERYTHING

-SEAN GERETY-