UNIT 13 SOFTWARE PROCESS IMPROVEMENT

Structure

13.0 INTRODUCTION

The process of Software engineering is undergoing rapid changes. Cloud platforms, Tools, open source frameworks, AI-driven automation are acting as catalysts to the software engineering changes. As there is a high risk in the software projects, it is imperative that we should adopt the best practices during software engineering and continuously improve the software engineering processes.
In this unit we discuss the improvements and optimizations for the software engineering processes.

We have depicted the continuous improvement process in Figure 13.1.

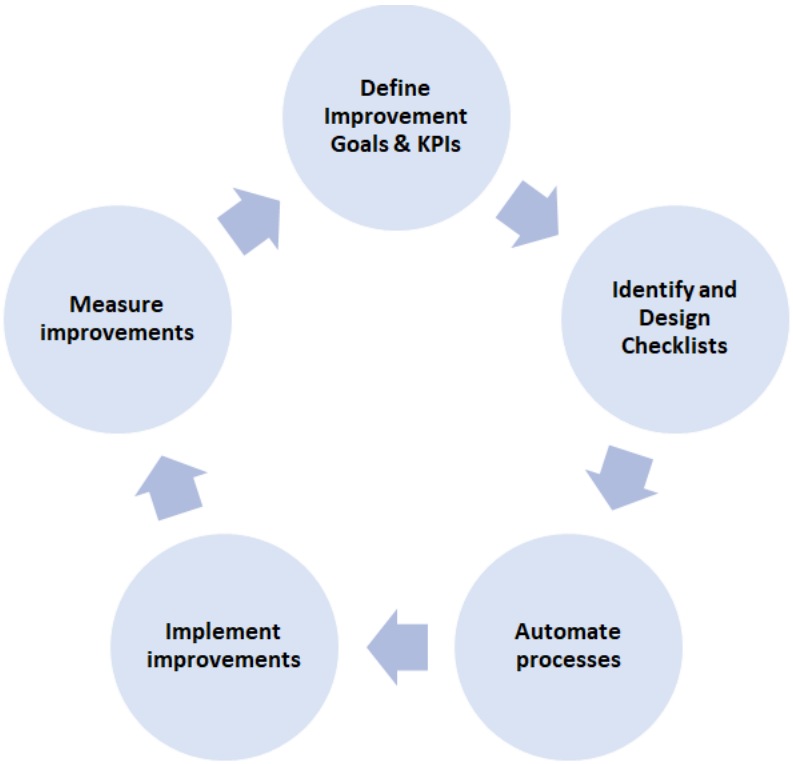

Figure 13.1 : Continuous Improvement Process

As part of the continuous improvement process, we define the main improvement goals and KPIs. The goals and KPIs should be aligned with the overall business vision. As a next step we identify the design checklists, naming conventions, tools, frameworks, methods and others. We then automate the processes and implement the planned improvements. We then measure the improvements and continuously fine tune it iteratively.

13.1 OBJECTIVES

After going through this unit, you should be able to

- understand key concepts of software process improvement,
- understand details of SEI CMMi process levels,
- understand details of software process optimization through first time right framework,
- software process optimization for validation process
- Understand SEI CMMi methods and best practices for requirements development, technical solution, requirement, requirements management, product integration and project planning.

13.2 SOFTWARE ENGINEERING INSTITUTE CAPABILITY MATURITY MODEL INTEGRATED (SEI – CMMI)

The CMMi is a process improvement model developed by Software Engineering Institute (SEI) at Carnegie Mellon University. The SEI-CMMi integrates the software

engineering processes, assesses the maturity levels, assesses the process readiness and provides the coverage for technical, physical and functional aspects of the system.

SEI institute was setup in 1982 at Carnegie Mellon University as an effort to resolve the software process challenges at the US Department of Defense (DoD). The first version of SEI-CMMi was published in 1991.

SEI-CMMi provides a phase-wise steps for each of the levels. Each level defines the capabilities, focus areas, process area and maturity needs for the organization. We can identify a process improvement area and apply the SEI -CMMi process for it to improve the process.

Given below are the key advantages of SEI- CMMi model:

- Improve the software engineering processes using the best practices suggested by SEI-CMMi model.
- Apply the proven processes and methods for software engineering
- Standardize and benchmark software engineering processes.
- Assures the quality of the deliverable.
- Assess the current maturity level of the organization and the process capability readiness.
- Helps us to compare the organizations using the maturity levels.
- Mitigates the risks associated with processes by following the proven methods

We have depicted the various levels of the SEI CMMi in the Figure 13.2.

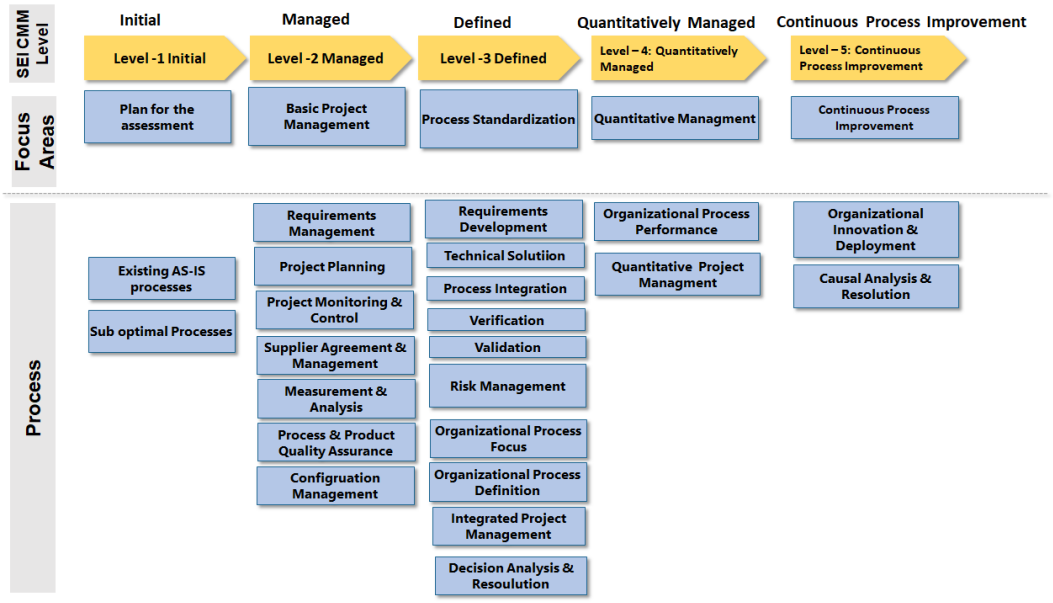

Figure 13.2 SEI CMM Levels

The figure 13.2 elaborates all the SEI CMM levels and the key focus areas and processes at each of the levels. The initial stage is the as-is stage of the organization. Level 1 is called "*Managed*" wherein the key focus area is basic project management. At this stage, we optimize and improve the processes related to requirements management, project planning, project monitoring, supplier agreement, and measurement, process and product quality assurance and configuration management.

Level 3 is "*Defined*" stage wherein we standardize the processes. We standardize the processes related to requirements development, technical solution, process integration, verification, validation, risk management, organizational process focus, organizational process definition, and integrated project management and decision analysis.

Level 4 is called "*Quantitatively managed*" stage where quantitative management is the main focus area. We aim to optimize the processes such as organizational process performance and quantitative project management.

The final level 5 is called "*Continuous process improvement*" wherein we focus on iteratively improving the organizational processes such as organizational innovation and deployment and causal analysis and resolution.

We look at examples for various SEI CMM levels in later section.

13.3 SOFTWARE PROCESS OPTIMIZATION THROUGH FIRST TIME RIGHT FRAMEWORK.

We have detailed the first time right (FTR) framework that incorporates the best practices at various phases of the software engineering. As part of the FTR framework we have explained the key best practices, methods, tools and improvement techniques during SDLC phases including requirements gathering, architecture and design, testing, DevOps, Infrastructure, project management and governance.

Figure 13.3 depicts the FTR framework.

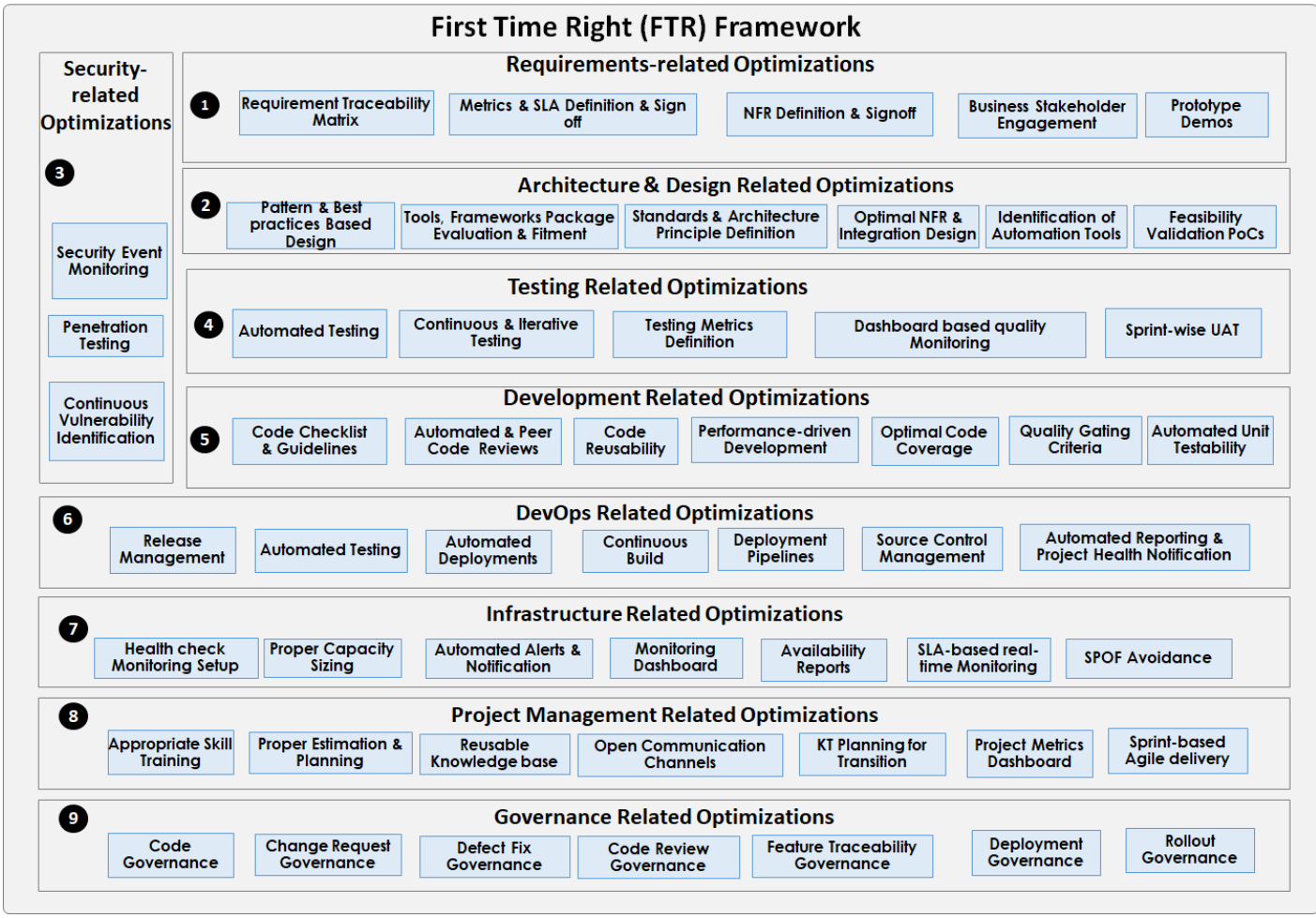

Figure 13.3 : First Time Right (FTR) Framework

Essentially the FTR framework encompasses various SDLC phases and comprehensive program concerns. We have categorized the key components of FTR framework into 9 logical categories:

13.3.1 Requirements Related Optimizations

In many scenarios the requirement gaps snowball into production issues. Hence it is crucial to plug all the gaps in the requirements stage. We have proposed below given optimizations during requirements elaboration stage:

- **Requirement traceability matrix**: The business analysts and the project manager should jointly own the requirement traceability matrix that maps each use case/Jira story to the corresponding test case, code artefact and the release details to ensure that there are no gaps in the delivery.
- **Metrics and SLA definition and sign-off**: We need to quantify the functional and non-functional requirements with accurate metrics and SLAs. For instance, a requirement like "*performance should be good*" is vague and ambiguous; we should formally quantity it as "*the total page load time for the home page should be less than 2 second with the maximum concurrent user load of 100 users in the North American geography*". We need to define the metrics related to application response time, availability, scalability, security and get a formal signoff from business ness.
- **NFR definition and sign-off**: All the non-functional requirements such as *security, performance, scalability, accessibility, multi-lingual capability* and such should be properly defined and signed off by the business.
- **Business stakeholder engagement**: The business stakeholders should be actively engaged throughout the requirement elaboration phase. Without active stakeholder engagement we would miss the requirements and we would face challenges in getting the sign-off.
- **Prototype demos**: We should prepare the mockups/prototypes iteratively and should do frequent demos to showcase the design, user journeys and multi-device experience. This helps us to pro-actively solicit the feedback comments from all the stakeholders and incorporate them.
- **Design Thinking Approach**: Leverage the design thinking approach to empathize with the user and iteratively define the optimal solution exploring the alternatives. Use working ideation and prototype sessions to test the solutions.

13.3.2 Architecture and Design Related Optimizations

During the architecture and design phase, we have has to ensure the following for delivering the first time right design and architecture:

- **Patterns and best practices based design**: The architect has to identify all the application architecture patterns, Industry best practices and design patterns applicable for the solution. As part of this exercise, we has to identify various layers, components and the responsibilities.
- **Tools, frameworks, package evaluation and fitment**: The architect has to evaluate the market leading products, frameworks, open source libraries that are best fit for the requirements. Leveraging proven frameworks, tools and open source libraries greatly contribute to the improved productivity, turnaround time and improved quality. For example as part of the architecture phase, the architect has to critically evaluate technologies such as Angular vs React vs Vue for UI; Spring Boot vs Serverless services, SQL vs NoSQL, mobile web vs hybrid app vs native app, PaaS vs IaaS vs SaaS etc.
- **Standards and architecture principles definition**: The architect has to define the applicable standards for the solution and for the business domain. At this stage the architect has to define the architecture principles such as headless design, stateless integration, token based security etc. Few application domains need to

implement few standards for regulatory compliance (such as HIPAA for health care domain)

- **Optimal NFR and Integration design**: The architect has to design the application to satisfy all the specified NFR (performance, scalability, availability and such) and the integration SLAs.
- **Identification of automation tools:** The architect has to identify all the automation tools that can be used for the project. This includes automation tools for code review, IDE, functional testing and such.
- **Feasibility validation PoCs:** For complex requirements, we need to carry out the feasibility assessment through proof-of-concepts (PoCs). This helps us to finalize the tool, technology, integration method, performance and assess the scalability and performance of the method.

As part of process improvement and optimization, we need to track the trends along various dimensions such as architecture shift, technology shift, integration shift and others and marked the recommended tenets for the solution. We have depicted the sample architecture trends in Figure 13.4.

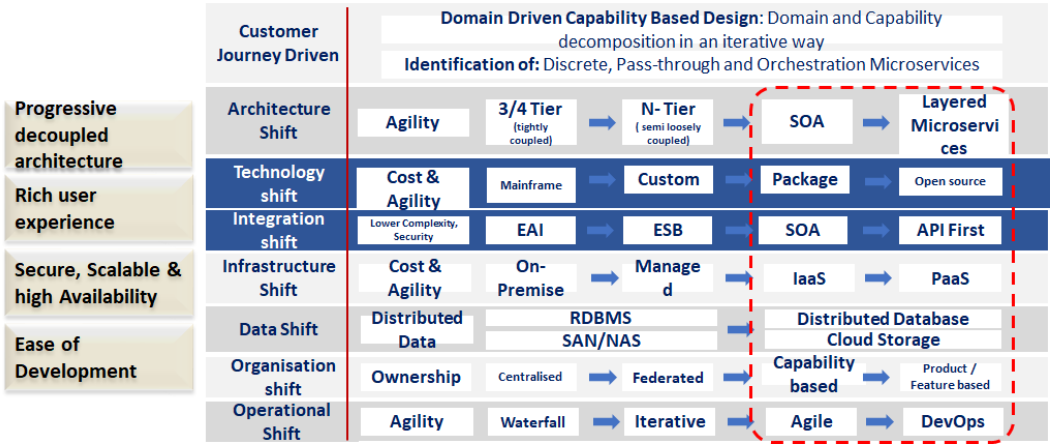

Figure 13.3 : Architecture Trends

13.3.3 Security Related Optimizations

On the security front, we need to continuously monitor and audit for critical security events (such as failed login attempts, password change events, impersonation events, role change events etc.) and setup a notification infrastructure for the same.
We should also carry out the penetration testing and vulnerability testing iteratively to discover the security vulnerabilities early.

13.3.4 Testing Related Optimizations

A comprehensive validation is one of the key success factors for delivering the first time right deliverable. The validation team needs to follow the below-given optimizations:

- **Automated testing**: The testing team has to use the automated tools such as Apache JMeter or Selenium to automate the regression scenarios and other functional test scenarios to improve the overall productivity.
- **Continuous iterative testing**: The testing has to be a continuous and iterative process across all sprints. The helps us to discover the defects in the early stage.
- **Testing metrics definition**: The testing team has to define the quality metrics such as defect rate, defect slippage rate, defect density and track the metrics with each sprint.

- **Dashboard based quality monitoring**: The test lead should pro-actively monitor all the testing metrics in a dashboard and notify the project manager in case of any critical violations.
- **Sprint-wise early UAT**: We have noticed that involving the business stakeholders with each sprint delivery in the UAT phase helps us to uncover any business related gaps early in the game. Additionally the team can incorporate the feedback in the next sprint.

13.3.5 Development Related Optimizations

The big chunk of responsibility for first time right deliverable lies with the development team. I will design and implement the below given optimizations for the development team:

- **Code checklist and guidelines**: Developers should use the code guidelines, naming conventions and coding best practices in a checklist format. Before the start of the project, the architect along with project manager has to create the checklist, naming conventions, best practices and such. Some of the most common code checklist are as follows:
    - Language specific coding best practices
    - Performance checklist
    - Security coding checklist
    - Code naming conventions
    - Design checklist
- **Automated and Peer Review process**: Developers should use automated static code analyzers such as PMD/SonarQube to ensure the code quality. The developers should also get their code reviewed by their peers and leads on a frequent basis.
- **Code reusability:** Developers should actively explore the code reusability opportunities in the following order:
    - Look for reusable libraries, code modules, components offered by the underlying platform/product/framework/accelerators.
    - Explore the availability of reusable libraries, code modules, components at the organization level.
    - Look for approved open source libraries that satisfies the functionality.
    - If nothing is available, develop the code in modular way so that it can be reused.
- **Performance driven development**: Performance should not be an afterthought, but it must be implemented from the very early stages. Developers need to pro-actively carry out the performance testing of the integrated code to ensure that their code is free of memory leaks, integration bottlenecks and others. We have detailed the performance based design and web optimization framework in the NFR section.
- **Optimal code coverage**: Developers should ensure that their unit test cases provide more than 90% code coverage. This ensures high code quality with minimal defects. Developers should also try to use an automated tool for generating unit test cases.
- **Quality gating criteria:** We need to define multi-level code quality gating criteria as follows (this could also go in as code check-in checklist):
    - **Developer-level code quality**: The developer has to use the defined coding checklist and naming conventions to adhere to those guidelines. The developer can also use the IDE for the same.
    - **Automated Local code quality analyzer**: The developer has to use the static code analyzers such as PMD, SonarQube to analyze all coding issues and fix the major and critical issues.
    - **Manual code review**: The developers can request for peer review and lead review of the code. Once all of the above is completed, the developers can check in the code to the source control system.

- **Integrated code review**: We could setup a Jenkins job with SonarQube to continuously review the integrated code and generate the report. The Jenkins job can notify the developers and project managers for major and critical violations.

- **Automated unit testability**: Developers should use automated unit test generators (such as EvoSuite, Veracode) to improve the quality and productivity of the developer.

13.3.6 DevOps Related Optimizations

DevOps defines a set of tools and processes to ensure the proper delivery and release of the application. Given below are the key optimizations from DevOps standpoint:

- **Release management**: DevOps setup provides an opportunity for setting up automated release management system
- **Automated testing**: We can configure unit test jobs and functional testing jobs (such as Selenium) to test the code from source control system
- **Automated deployment**: We can use Jenkins pipelines and deployment jobs to automate the deployment.
- **Continuous build**: We can enable the continuous build to catch any errors with integrated code.
- **Deployment pipelines**: The pipelines enable and automate the code deployment.
- **Source control management**: As part of DevOps process we also define the source control management processes related to pull request, approval, code check-in, code merge and such.
- **Automated reporting and project health notification:** We could configure the project health notification, reporting features to trigger the notification in case of any SLA violation.

We have depicted the process improvement and optimizations for Azure DevOps in figure 13.5.

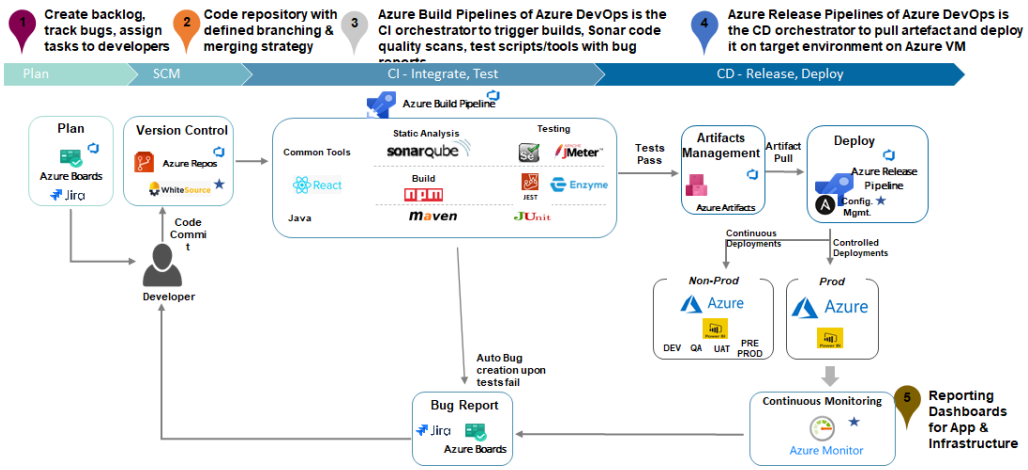

Figure 13.5 : DevOps Pipeline in Azure DevOps

13.3.7 Infrastructure Related Optimizations

A right sized infrastructure is critical for the optimal application delivery. In order to deliver the first time right solution, given below are the infrastructure related optimizations:

- **Health check monitoring setup**: We need to setup the pro-active healthcheck/heartbeat monitoring infrastructure to continuously monitor the availability of the servers.
- **Proper capacity sizing**: All the server and network capacity should be sized based on the user load and related non-functional requirements (such as scalability, availability, performance). Based on the availability requirements, we should also setup the disaster recovery (DR) and the corresponding sync jobs.
- **Automated alerts and notification**: We should be able to configure notification triggers and the monitoring setup should notify the administrators in case of SLA violation.
- **Monitoring dashboard**: The monitoring dashboard should provide a snapshot of all key parameters such as availability percentage, performance, CPU/memory/network utilization, request rate and such.
- **Availability reports**: The monitoring setup should be able to generate on-demand monitoring reports for various infra-related parameters.
- **SLA based real time monitoring**: We should also setup the real-time application monitoring infrastructure across various geographies to understand the performance and availability issues.
- **Single Point of Failure (SPOF) avoidance**: Ensure that none of the systems in the end to end request processing pipeline form the bottleneck leading to single point of failure (SPOF). We can ensure high availability through cluster setup, redundancy setup, DR setup, regular backup and by other means.

Additionally we should setup the automatic elastic scalability, optimal load balancing, CDN based caching based on the application requirements.

13.3.8 Project Management Related Optimizations

Project managers shoulder greater responsibility in defining and enforcing the processes to implement first time right deliverable. Given below are the key optimizations from project management standpoint:

- **Appropriate skill training**: The project manager has to train the project team members to equip them with suitable technology skills. It is recommended to have right mix of skillset (lead to developer ratio) in the team to ensure timely delivery and high quality deliverable.
- **Proper estimation and staffing:** The project manager has to do the proper effort estimates and staff resources accordingly. The project manager has to define proper risk management plan for all known contingencies.
- **Reusable knowledge base:** The project manager has to maintain the best practices, Standard Operating Procedures (SOP), key design decisions (KDD), How-to documents, troubleshooting documents, learnings, process documents, KT documents and others in a centralized knowledge base. This helps the team members to build upon the collective knowledge and leverage it for faster and better deliverable.
- **Open communication channels**: In order to avoid information silos, the project manager has to establish open communication channels across all tracks. Various tools such as Slack, MS Teams can be leveraged for this.
- **KT Planning for transition:** If the team takes over any new engagement from an incumbent, we need to plan for proper KT from the incumbent.
- **Project metrics dashboard**: The project manager has to track the overall project health in the metrics dashboard. Typically a project dashboard consists of burn rate, code quality score, open defect count, average defect fix time, sprint run rate and such. This helps the project manager to get a holistic view of all open issues and prioritize them accordingly.
- **Sprint based Agile Delivery:** Modern digital projects are complex and are end-user focused. Hence the project manager has to plan for sprint based agile delivery so that we can mitigate the risk and get the early user feedback.

- **Continuous Improvement model:** The project manager has to constantly seek to improve the overall productivity and quality using the learnings, metrics from earlier sprints. The project manager has to continuously look out for automation opportunities.

The project manager also has to pro-actively understand the regulatory and compliance requirements (such as accessibility standards), browser and device compatibility (browsers and mobile devices that need to be supported), multi-lingual requirements (list of left-to-right and right-to-left languages) that need to be supported and accordingly plan for the same.

13.3.9 Governance Related Optimizations

Project governance broadly details the well-defined processes to streamline the complex project activities. Given below are the governance-related optimizations:

- **Code Governance** defines the code management related processes such as code merging, code check in, code versioning and such.
- **Change request governance** broadly defines the scope creep management process. It defines the criteria for change request management, its prioritization, and impact management and implementation plan.
- **Defect fix governance** defines the processes related to defect prioritization, SLAs and such.
- **Review governance** defines the processes related to code review and approval.
- **Feature traceability governance** defines the process to trace the requirement to the final release.
- **Deployment governance** provides the deployment plan for code artefacts.
- **Rollout governance** defines the rollout plan across various geographies, features, languages, rollback plan and such.

We have depicted the key elements of solution governance in Figure 13.6.

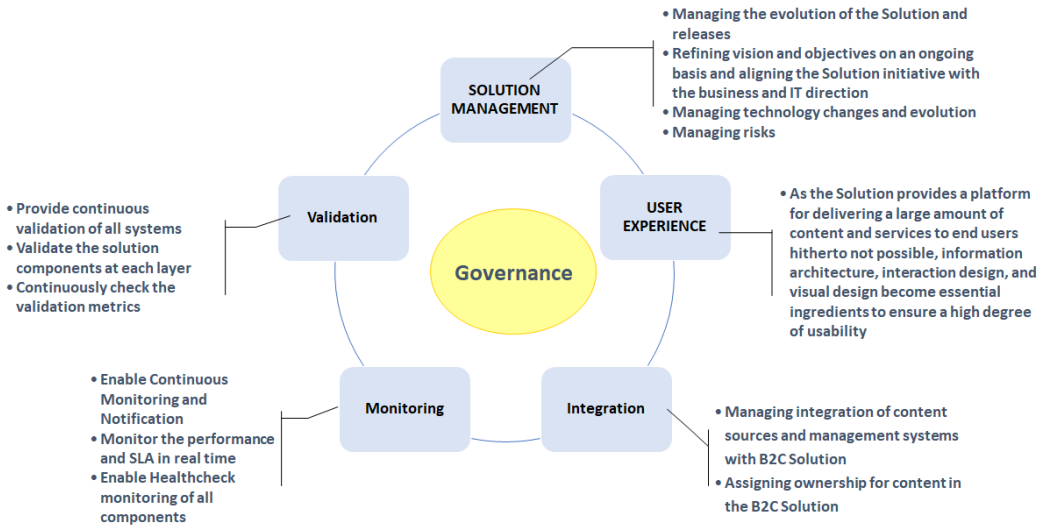

Figure 13.6 : Solution Governance

13.3.10 Tools

We have given various tools that can be used for implementing the First time right framework in Table 13.1.

Table 13.1: FTR Tools

Tool Category	Open Source/ Commercial Tool(s)
Web Page Analysis tools (HTML analysis, performance benchmarking, improvement guidelines)	Yahoo YSlow , Google PageSpeed , HTTPWatch , Dynatrace AJAX Edition
Page development tools (analysis of page load times, asset size, asset load times and such.)	Firebug, Google Chrome Developer toolbar, Fiddler , HTTP Archive WEB PAGEiddle, CSSLint , JSLint , W3 CSS Validator , W3 HTML validator
Asset merging and minification tools (JS/CSS minification)	Yahoo UI (YUI) minifier, JSMini , JSCompress
Page Performance Testing tools (load simulation)	JMeter, LoadUI, Grinder, Selenium
Image compression tools	PNGCrush, Smush It , Img min , JPEG Mini
Web Server Plugins (for automatic compression, minification, merging, placement, caching etc.)	Mod_pageSpeed , mod_cache, mod_SPDY mod_expiry , mod_gzip
Website Performance Testing	GTMetrix , Pingdom
Synthetic monitoring (transactions simulation & performance statistics)	Web Page test , DynaTrace Synthetic monitoring
CDN	Akamai, CloudFlare, KeyCDN,
Web Analytics (track user behavior, performance reporting)	Google Web Analytics, Omniture, Piwik
CSS Optimization tools	CSS Sprites , SpriteMe , SpritePad
Bottleneck Analysis (dependency & bottleneck analysis)	WebProphet , WProf
Real User Monitoring (RUM) (monitoring & bottleneck analysis)	New Relic , Dynatrace , Gomez
Network analysis (network traffic, HTTP headers, request/responses, protocol analysis)	Wireshark , Charles Proxy
Application Performance Monitoring (APM) (Layer wise monitoring of application code)	New Relic , Dyna Trace Monitoring, Nagios

☞ **Check Your Progress 1**

1. The focus of SEI CMMi level 2 is _____
2. Requirements development is included in level _____
3. The focus areas of level 3 of SEI CMMi is _____
4. _____ maps each use case/Jira story to the corresponding test case
5. _____approach to empathize with the user and iteratively define the optimal solution exploring the alternatives
6. We assess the feasibility of the tool/method through _____

13.4 SOFTWARE PROCESS OPTIMIZATION FOR VALIDATION PHASE

In this section we discuss the software process optimization for the performance validation phase.

A robust web performance test methodology defines the sequence and activities for various performance testing phases. We have detailed the performance test methodology in Figure 13.7.

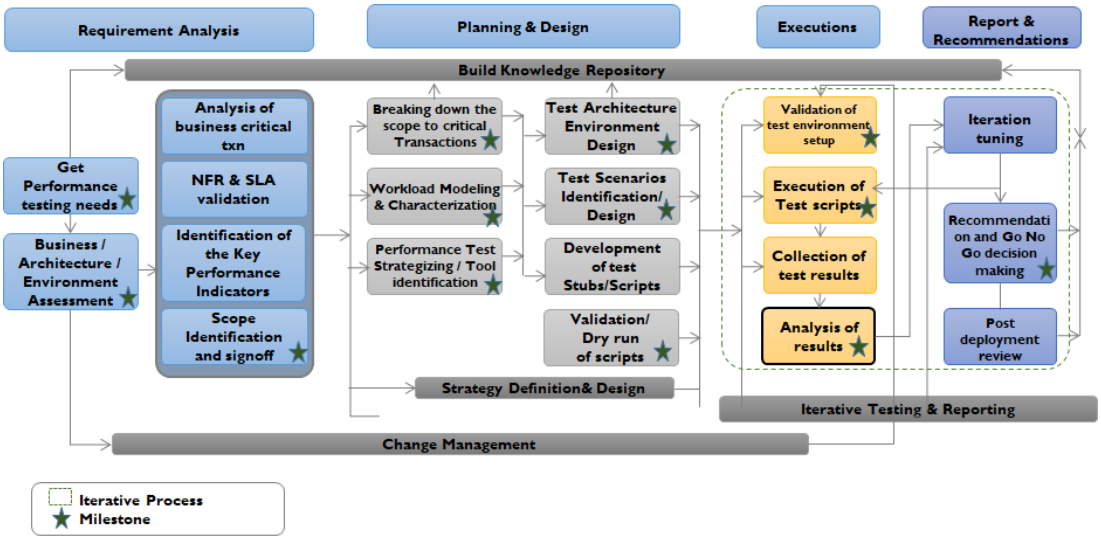

Figure 13.7 : Web Performance Test Methodology

During ***requirement analysis phase***, the performance testing team will get the complete performance test needs and document the requirements. During this phase, the testing team assesses the architecture and design documents. The team would also analyze the business critical and performance critical transactions. The performance validation team gets sign-off on performance-related NFRs (such as peak user load, peak content volume and such) and performance-related SLAs (timing metrics for the defined NFRs) during this phase. The team identifies the key performance indicators (KPI) such as average page response time across all geographies, availability metrics, performance metrics, performance SLA adherence rate and such. Some of the common performance related questions are given below:

- Which pages receive highest traffic?
- How many pages have acceptable performance above configured performance thresholds?
- Which pages have bad performance?
- What is the caching strategy used?

The ***planning and design phase,*** the performance testing team will break down the scope into business transactions that need separate performance scripts. We also characterize the workload model based on the load numbers we have gathered. The performance testing team creates the overall performance test plan. The performance environment will be setup after all the performance-testing tools are finalized. Performance testing team develops the performance test scripts.
During the ***execution phase,*** the performance test team executes all the scripts and record various timing metrics that are defined in the requirements analysis phase. Various performance tests such as load testing, stress testing, endurance testing will be carried out in this phase. Automation scripts will be used to automatically execute the performance tests after each major release. The results will be analyzed and document.
Reports will be published in the **reporting and recommendation** phase. Performance testing teams recommends a “go” or a “no-go” decision based on the overall performance and adherence of the performance results to the defined SLAs. Performance testing and reporting will be done iteratively for various sprints of the release.

13.4.1 Performance Testing Tools

Given below are the key testing tools that can be used for web performance testing in Table 13.2

Table 13.2 : Performance Testing Tools

Category	Performance testing tools
Web Page Testing	Google Chrome Lighthouse Google PageSpeed Insights
Application Profiling	Open Source: JProbe, Eclipse Profiler Commercial: Jprofiler, OptimizeIt,
Load testing	Open Source: Apache JMeter, Grinder, Apache Bench, HTTPPerf Commercial: HP LoadRunner, NeoLoad, BlazeMeter
Service testing	LoadUI, SOAPUI
Client side performance testing	https://www.webpagetest.org/
Real time web performance monitoring	Open Source: Nagios, Hyperic – HQ, Perfmon, NMon Commercial - HP SiteScope, Wily Introscope
Application Performance Monitoring (APM)	Dynatrace, AppDynamics, New Relic
Mobile App testing	Appium, UI Automator
Analysis tools	GC Analyzer, JVM Analyzer

13.4.2 Performance Monitoring and Notification

A robust monitoring and alerting setup should be able to capture the system metrics (CPU, memory), log metrics (system logs, application logs), errors, performance and such. The monitoring setup should also be flexible to monitor various systems such as Linux OS, database server, stand-alone server, performance and such. The alert and notification setup should be flexible enough to send notifications to various channels such as email, pager, incident management system and such. We should be able to configure the performance thresholds and resource utilization thresholds that can trigger the notification.
A comprehensive monitoring tool should support these features:

- Core Monitoring capabilities
  - Resource (CPU, Memory) monitoring
  - Network (Router, Switch, and Firewall etc.) Monitoring
  - Server Monitoring
  - Windows Event Log Monitoring
  - Applications Monitoring
  - Virtual instances
- Application Server monitoring capabilities
  - Database Monitoring
  - Web Page, Web Server/ Web Services Monitoring
    - Middle Ware Monitoring
    - Custom Application Monitoring
- Reporting Dashboard
  - Business service management views

- Comprehensive dashboard
- Real Time trends and availability of devices
- Events and Correlated Alarms

- Reporting
  - Standard daily, weekly, monthly, quarterly, and yearly reports

13.4.3 Performance Monitoring Tools

We could leverage many open source and commercial tools for performance monitoring. Table 13.3 lists the popular performance monitoring tools.

Table 13.3 : Performance Testing Tools

Monitoring needs	Monitoring Tool
System Monitoring (CPU, Memory, and disk)	• Windows Performance monitor (perfmon) • NMon for AIX servers • System Activity Report (SAR) for UNIX systems • Node exporter for container pods
File monitoring	Filebeat
Network monitoring, System monitoring and infrastructure monitoring	• Nagios • ELK (Elastic search, Logstash and Kibana) • Grinder • AppDynamics (Commercial) • New Relic (Commercial) • Zabbix (https://www.zabbix.com/)
Statistics dashboard and visualizations **Alert and monitoring dashboard**	• Kibana • Grafana
Real time event monitoring **Visualizations** **And data queries**	Prometheus & Grafana
Search engine	Elasticsearch
Synthetic monitoring tool	• DynaTrace (Commercial) • Selenium • Lighthouse • Webpagetest.org
Database monitoring	• Automatic Workload Repository (AWR) • Fluentd
Log monitoring	• Splunk • Fluentd
Message streaming	Apache Kafka
Notification	Alert Manager
Container Monitoring	• Node Exporter • Docker Stats • cAdvisor • Prometheus
Web Page Monitoring (page size, page response time, number of requests, asset load time etc.)	• Webpagetest.org • Site Speed (https://www.sitespeed.io/) • Google Page Speed Insights (https://developers.google.com/speed/pagespeed/insights/)

	• Pingdom (commercial) • Silk performance Manager (commercial) • Uptrends (commercial) • https://web.dev/measure/
Development tools/Page Auditing	• Google Chrome Developer Tools • Test My site (https://www.thinkwithgoogle.com/feature/testmysite/) • Google Chrome lighthouse • HTTP Watch • https://speedrank.app/en • Fiddler • Firebug • Web Tracing Framework http://google.github.io/tracing-framework/ • Timeline tool https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/timeline-tool#profile-painting
Multi-geo web performance testing	• https://performance.sucuri.net/
Cloud Monitoring	• AWS CloudWatch
Website speed test	• https://tools.keycdn.com/speed
Load testing	• BlazeMeter • Apache JMeter
Web site latency test	• Ping Test (https://tools.keycdn.com/ping)
Real user monitoring (RUM)	• New Relic • SpeedCurve (https://speedcurve.com/) • DynaTrace • Akamai mPulse (Commercial) • AppDynamics (Commercial)
Analyze network timings	• Resource Timing API (https://developer.mozilla.org/en-US/docs/Web/API/Resource_Timing_API/Using_the_Resource_Timing_API) • Network Information https://developer.mozilla.org/en-US/docs/Web/API/NetworkInformation

13.5 SEI CMMi PROCESS EXAMPLES

In this section we shall look at improving the processes at various SEI CMMi levels. We have taken the focus areas and the best practices for each of the processes.

13.5.1 Requirements Development

This is part of level 3 "Defined" stage wherein the expectation is to thoroughly understand the requirements and develop the solution for that.

Key Focus Areas
Given below are the key focus areas:

- Understand the requirements from various perspectives (business, operations, performance) and document the requirements.

- Thoroughly understand the integration requirements and the related nonfunctional SLAs related to security, performance, scalability and availability.
- Understand the exception scenarios for each of the requirements.

Methods for process standardization
Given below are the main methods and best practices for requirements development process standardization:

- Design a comprehensive requirements gathering template to cover the requirements from various dimensions.
- Elaborate the user stories and document the exception flows
- Define bi-directional requirement traceability matrix to trace the requirements to its corresponding development artefacts.
- Validate the requirements with the stakeholders and obtain a formal sign-off from the stakeholders.
- Build quick prototypes to validate the user experience related requirements.

13.5.2 Technical Solution

This is part of level 3 "Defined" stage wherein the expectation is to build the solution based on best practices and best of breed technologies.

Key Focus Areas
Given below are the key focus areas:

- Evaluation of alternate solution options and tools
- Evaluate build, reuse or buy decisions
- Design and develop a scalable and high performing solution

Methods for process standardization
Given below are the main methods and best practices for Technical solution process standardization:

- Evaluate the alternate solution options against the define criteria (such as cost, effort, performance)
- Elaborate the detailed design and leverage the best practices, recommended naming conventions and use patterns.
- Use the automated static code analyzers to ensure code quality.
- Develop unit test cases to ensure adequate code coverage
- Document the code and provide a user guide.

13.5.3 Requirements Management

This is part of level 2 "Managed" stage wherein the expectation is to manage the requirements and resolve any inconsistencies across various requirements.

Key Focus Areas
Given below are the key focus areas:

- Change management for the requirements
- Identify the inconsistency among the requirements
- Manage the requirement change and its impact

Methods for basic project management
Given below are the main methods and best practices for requirements management:

- Leverage various methods such as prototyping, user journey mapping, interviews, focus group discussions, story boarding for gathering requirements.
- Define the change management process.
- Develop bi-directional requirement traceability matrix to ensure the proper implementation of requirements.

13.5.4 Product Integration

This is part of level 3 "Defined" stage wherein the expectation is to provide an optimal integration across various interfaces.

Key Focus Areas
Given below are the key focus areas:
- Identify all the needed interfaces
- Optimal assembly of the product components.
- Define the integration SLAs

Methods for process standardization
Given below are the main methods and best practices for product integration process standardization:

- Design and implement integration specification document to specify all the APIs, contracts and exception scenarios
- Test the interfaces at various loads to understand the performance and scalability
- Handle the exception scenarios such as timeouts, system down issues.

13.5.5 Project Planning

This is part of level 2 "Managed" stage wherein the expectation is to define the accurate project plan for managing the project

Key Focus Areas
Given below are the key focus areas:
- Accurate scope, effort and time estimation
- Accurate risk management.
- Appropriate stakeholder management.

Methods for basic project management
Given below are the main methods and best practices for project planning:

- Use accurate estimates for proper project planning.
- Create a comprehensive project plan to cover the risks, milestones, project metrics, dependencies, resources, acceptance criteria, training plan, escalation matrix, knowledge management plan and others.
- Identify main business stakeholders for each of the departments.
- Develop comprehensive stakeholder management plan covering the following as active engagement of all concerned stakeholders is a key to sell the solution and for the long term success of the program:
    - Map their roles, responsibilities and ownership in the matrix

- Communicate the key decisions and articulate impact for each of the stakeholders and highlight the impact for their respective departments.
- Address their concerns in separate spin-off meetings.
- Take the key stakeholders into confidence in the overall decision making.
- Organize the demos, PoCs, prototypes, journey map walkthrough, product evaluation sessions based on the roles and responsibilities for each of the stakeholders.

Check Your Progress 2

1. _____ captures the system metrics (CPU, memory), log metrics (system logs, application logs), errors and performance.
2. Splunk is one of the _____ tools
3. _____ traces the requirements to its corresponding development artifacts
4. _____ specifies all the APIs, contracts and exception scenarios
5. One of the tools to check the web site latency is _____

13.6 SUMMARY

In this unit, we started discussing main features of the SEI CMMi. We looked at various 5 levels of the SEI CMMi methodology. The level 1 is called initial. The level 2 is termed as managed where the main focus area is basic project management. Process standardization is the focus area of level 3 defined. The focus area of level 4 is quantitatively managed and the continuous process improvement is the focus area of level 5. We then discuss the first time right (FTR) framework that discusses the optimizations in the areas of requirements, security, testing, development, DevOps, infrastructure, project management and governance. We also looked at the tools used in the FTR framework. We then had a deep dive about the software process optimization for performance validation framework. We then looked at five examples of SEI CMMi process optimization examples related to requirements development, technical solution, production integration, and project planning and requirements management.

13.7 SOLUTIONS/ANSWERS

Check Your Progress 1

1. Basic project management.
2. Level 3
3. continuous process improvement
4. Requirement traceability matrix
5. Design thinking
6. Proof of concept (PoC)

Check Your Progress 2

1. Monitoring and Notification.
2. log monitoring

3. bi-directional requirement traceability matrix
4. integration specification document
5. Ping Test

13.8 FURTHER READINGS

References

Shivakumar, S. K., Shivakumar (2018). *Complete Guide to Digital Project Management*. Apress.

Team, C. P. (2002). CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.1, Continuous Representation. *CMU/SEI*.

Conradi, H., & Fuggetta, A. (2002). Improving software process improvement. *IEEE software*, *19*(4), 92-99.

https://en.wikipedia.org/wiki/Capability_Maturity_Model

https://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration

UNIT 14 EMERGING TRENDS IN SOFTWARE ENGINEERING

Structure

14.0 INTRODUCTION

The field of Software Engineering is rapidly evolving. Fast paced changes, rapid innovations, market demands and ever increasing user demands result in continuous changes and improvements in the software engineering field. Innovations and changes happen in the software engineering tools, software engineering process, programming languages, hosting/delivery platforms and other areas. It is imperative of software engineers to track the industry changes and keep oneself up to-date with the changes. The key drivers behind the changes are productivity improvement, quality improvement, and quick time to market, cost optimization, high performance, high availability and high scalability.
In this unit we shall examine the key emerging trends in the field of software engineering

14.1 OBJECTIVES

After going through this unit, you should be able to

- understand key emerging trends of software engineering,
- know the high level details of low-code and no-code development,
- know the high level details of Artificial Intelligence (AI),
- know the high level details of Blockchain,
- know the high level details of Internet of Things (IoT),
- know the high level details of Augmented Reality (AR), Virtual Reality (VR),
- know the high level details of continuous delivery,

- know the high level details of containerization,
- know the high level details of Cloud Delivery Model, and
- look at various delivery approaches such as mobile-first, cloud-first, and offline first approach.

14.2 DRIVERS FOR INNOVATION

The key drivers for the software engineering innovations that lead to the emerging trends are given below:

- **Productivity Improvement**: Software engineering methods (such as low-code/no code, containerization) and delivery models (such as cloud delivery model) aim to improve the overall productivity of the developers thereby reducing the time to market.
- **Quality Improvement**: Process innovations such as continuous improvement, agile delivery enhances the overall delivery quality reducing the risk.
- **User Experience Improvement and User engagement**: Innovations such as AR, VR, AI, blockchain and machine learning improve the end user experience and provide the relevant, contextual information in quick time.
- **Automation**: AI and Machine learning are the key enablers for driving the automation.

We will look at the each of the trends in next sections

14.3 KEY EMERGING TRENDS IN SOFTWARE ENGINEERING

The key emerging trends in software engineering are as follows:

- Low Code/No Code Development wherein the tools and platforms enable developers and business stakeholders to develop the solutions with minimal code.
- Artificial Intelligence (AI) and machine learning methods enable developers to analyze huge quantity of data, detect patterns, predict trends and forecast values.
- Blockchain technologies a secured distributed record handling capabilities
- Internet of Things (IoT) platform provide methods, tools and technologies to track and monitor the devices
- Continuous delivery is a set of processes, tools and technologies that provide iterative and agile delivery of the solution with incremental features.
- Cloud Platforms has redefined the way software is built, deployed and delivered.
- Containerization provide self-contained and portable deployment model
- BigData technologies handle massive volume of data with high performance and high availability
- Augmented Reality enhances the real world objects by adding digital objects.
- Mobile-first, cloud-first, offline-first are some of the emerging delivery methods.

- Other trends include serverless functions, conversational applications, analytics, computer vision, service orchestration, Software as Service (SaaS), cross platform development

Figure 14.1 provides the key trends in software engineering.

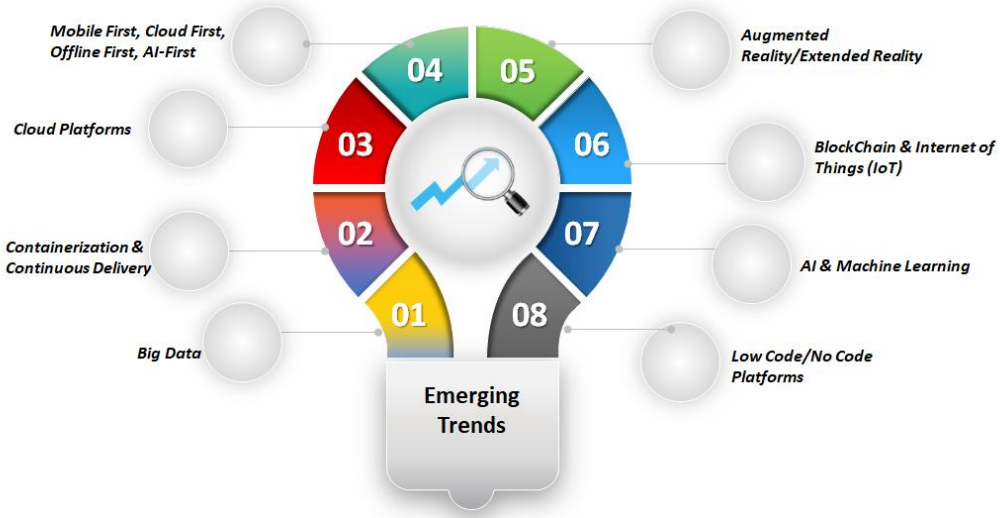

Figure 14.1 : Emerging Trends in Software Engineering

14.4 ARCHITECTURE BRIEF DETAILS OF EMERGING TRENDS

In this section we look at the details of emerging trends. For each of the trends, we shall look at the motivation, key concepts, tools, technologies involved as a part of the trend.

14.4.1 Low Code and No Code Platforms

The low code and no code platforms (LCNC) allow the users to rapidly build the user interfaces (UI), solution components, integrations with zero or minimal code. While enterprises demand rapid application development, on the other hand there is a shortage of the skilled developers; LCNC platforms are designed to fill this gap. LCNC platforms provide visual interfaces, wizards, drag and drop features that users can leverage to rapidly develop the solution components. LCNC is the new avatar of the Rapid Application Development (RAD) tools. LCNC goes a step ahead and provide pre-built integrators, reusable UI templates, solution components and configurable modules to ease the development. Due to the minimal custom coding involved, the testing effort, maintenance effort is also reduced.
LCNC platforms reduce the hand coding effort, automate the code generation process by abstracting the finer details of the software development. LCNC platforms heavily use automated code generation, visual programming paradigm and model based design.

The salient features of LCNC platforms are as follows:

- Visual programming model through which citizen developers can develop the solution
- Pre-built integrators to retrieve and persist data faster
- Drag and drop features to rapidly build the application.
- Model driven approach for building solutions
- Reusable Pre-built components, vertical solution accelerators to speed up development

- Configurable solution components that can be extended for the business needs.

The key advantages of LCNC platforms are as follows:

- Faster time to market
- Improved developer productivity
- Enablement of business team with minimal coding experience
- Rapid development and deployment
- Increased quality of the solution
- Scalable application with reusable components
- Enable quicker integration with enterprise systems.
- Leverage reusable components and in built user experience.
- Provides business self-service model
- Enable enterprises to become more agile and responsive to market dynamics
- Innovate fast with increased productivity.

Given below are some of the examples of the LCNC Platforms:

- Pega software provide the visual tools for modeling the enterprise business process and to build the corresponding screens.
- Salesforce Lightning Platform provides automation tools for modeling and developing flows through process builder. The platform also provides tools to build mobile apps with high security with minimal code.
- Microsoft Power Platform provides features to build app by using pre-defined templates, easy to use connections and cross-platform applications
- Appian platforms provide drag-and-drop features to model the complex business process.

14.4.2 Artificial Intelligence (AI)

We are living in an era where most of the tasks, processes and day-to-day activities are getting increasingly automated by machines. Automation brings in productivity, convenience, time savings, cost savings that allow humans to invest their time and energy in more valuable and complex tasks. Machine-led automation is revolutionizing the way we live, communicate, work and do business. AI-led systems are becoming ubiquitous, impacting modern human civilization in numerous forms and re-defining the daily tasks in our lives. Due to the wide impact of AI technology, we have discussed it in detail in this section. We shall look at various disciplines used in AI, applications and trends within AI.

Artificial Intelligence (AI) is the guiding force behind the automation revolution. AI helps machines “learn”, “understand” like the way humans do. AI is an interdisciplinary science that uses various methods such as natural language processing (NLP), machine learning (ML), knowledge processing, reasoning, predicting and such.

In simple terms AI includes methods and systems to make the machine perform intelligent and cognitive tasks that humans are good at performing. Tasks such as game playing, language translation, recognition patterns, identifying images, coming up with a rationale decision, problem solving, reasoning and such.

AI and Machine learning are redefining the ways people interact with machines, the ways enterprises analyze data and offer many features such as learning, prediction, forecasting, trend identification and such.

AI is an interdisciplinary field that has borrowed various methods, concepts, theories from various fields. We have given the list of fields that has contributed to the AI. We have depicted the fields of AI in figure 14.2.

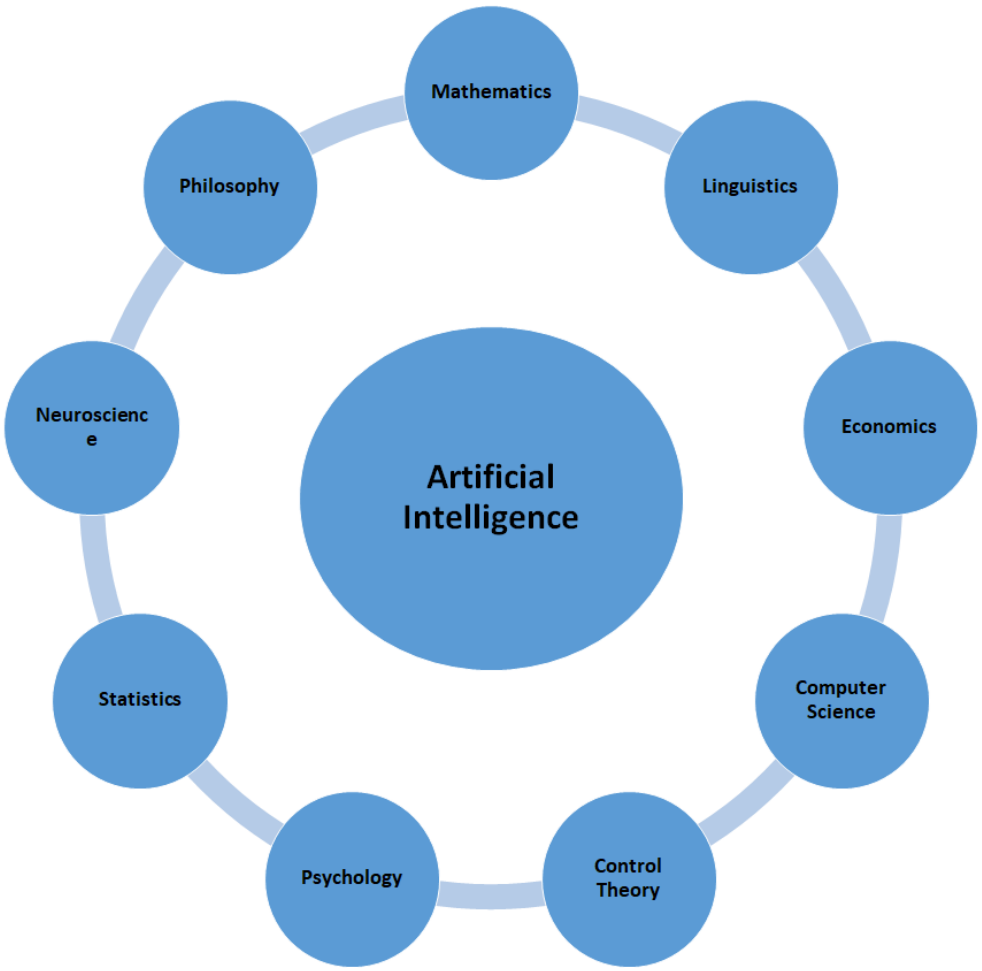

Figure 14.1 : Fields of AI

Philosophy

Concepts such as logical reasoning and theory of rationality, theory of language, reasoning are part of the philosophy. Laws of rationality define the laws and rules that govern the mind and dualism (describing dual forms of reality material/physical and immaterial/spiritual and mind and body are separate from each other), materialism (mind and body and other entities in the world are adhering to physical laws), principle of Induction (rules are based on association sensory exposure to the elements) and others. The key ideas of philosophy that has influenced the Artificial intelligence are as follows:

- Logic and methods of reasoning
- Mind as physical system
- Foundations of learning
- Language
- Rationality

Mathematics

Concepts such as Probability, Logic (such as propositional logic and predicate logic), algorithms, decidability and Formal representation and proof algorithms, automated reasoning, Formal Logic and knowledge representation. Historically the concepts such as intractability (time for solving the problem exponentially increases with the input size such as NP complete problems), reduction (transformation of one class of problems to another class with known solution), probability theory (Probability contributes the "degree of belief" to handle uncertainty in AI), *Decision theory* combines *probability theory* and *utility theory*

Economics

Concepts such as the following influence the field of AI:

- Game theory
- Formal theory of rational decisions
- Combination of decision theory and probability theory for decision making under uncertainty
- Markov decision processes

Linguistics

Concepts such as grammar, natural language processing, and knowledge representation are key influences to the AI field along with the below given topics:

- Understanding natural languages through different approaches.
- Formal languages
- Syntactic and semantic analysis
- Knowledge representation
- Relationship between language and thought process

Statistics

Topics such as regression models, learning from data, modeling uncertainty are used in artificial intelligence.

Neuroscience

The key ideas influenced by neuroscience are:

- Neurons as information processing units used in artificial neural networks.
- Study of brain functioning

Psychology

Concepts related to behaviorism and cognitive psychology (the study of information processing by brain) are areas of interest in psychology. The key factors of psychology influencing the artificial intelligence are:

- Phenomena of perception and motor control
- Human adaptation analysis
- Human behavior analysis, (how people think and act)
- The study of human reasoning and acting
- How do humans learn and process knowledge
- Study of learning, memory and thinking
- Provides reasoning models for AI
- Human brain as an information processing machine.

Computer Science

Most of the core concepts of the computer science such as programming, algorithms, design patterns, parallel computing, data structure, large scale computing, Machine learning, pattern detection, grid computing and such.

Control Theory

The main concepts influenced by control theory are as follows:

- Stability of systems
- Simple optimal agent design (systems that maximize objective function over time)
- How can artifacts operate under their own control?
- Optimal agents receiving feedback from the environment.
- Adaptation of the artifacts and their actions to:
    - Do better for the environment over time
    - Based on an objective function and feedback from the environment

Given below are some of the key AI related applications

- Autonomous vehicles: Self-driving automobiles is a fast-emerging area that heavily involves AI methods to perform tasks such as obstacle sensing, automatic navigation, autonomous planning, real-time sensor data processing, reasoning and decision making and such.
- Conversational Interface: Chatbots, virtual assistants (such as Apple Siri, Amazon Alexa) converse with humans in natural language and can perform structured tasks (such as booking appointments, route navigation, search, recommendations, activity planning, reminders and such).
- Computer Vision: Machines can perceive the images through computer vision methods.
- Robotics: Commercial and industrial robots employ AI methods to perform well-defined and structured tasks such as car washing, floor cleaning, assembling parts, motion planning and such. Highly advanced robots are also used in complex surgeries.
- Natural Language Processing: The key methods in this area are entity extraction, part of speech (POS) identification, intent identification, language "understanding" and such. Voice bots, chat bots, voice search, virtual assistants heavily use natural language processing. Other tasks such as machine translation, sentiment, information retrieval, text mining analysis also rely upon NLP methods.
- Expert systems: These systems are used for specialized activities such as legal advisory, medical advisory, domain-specific problem diagnosis (such as financial analysis, legal analysis, medical diagnosis, financial forecasting, Spam controls, logistics planning, document summarization and fraud detection). Expert systems store, process and use the knowledge for performing the activities.
- Game playing: Machines use AI methods such as automated reasoning, decision making to identify the most optimal moves in games such as Chess, Go, Checkers, Jeopardy!. As games have well-defined and structure rules, they can be formally defined which computers can use and learn/adapt from previous mistakes.
- Theorem proving: Theorem provers are mainly used to prove the correctness of mathematical theorems.

- Vision/Perception: Computer vision agents help in object recognition, face recognition, navigation, and information extraction.
- Search engines and Question and answer systems

AI is rapidly advancing into all fields of technology. Given below are the key emerging trends:

- Deep Learning: This areas of AI uses artificial neural networks to do the complex tasks such as automatic navigation, pattern recognition, computer vision, autonomous vehicles and such.
- Autonomous vehicles: Consumer vehicles such as cars and transportation vehicles such as trucks are gaining
- Facial Recognition: Recognizing and using the face as a biometric authentication is gaining popularity.
- Edge AI: AI chips are used for faster execution of facial recognition, NLP methods, speech recognition, computer vision, autonomous vehicle and such.
- Conversational AI: Many of the user facing actions such as incident management, text search, voice search, recommendation can be automated by conversational interfaces such as chatbots.
- Analytics: AI can be used along with analytics to achieve more accuracy in predictive analytics.
- Personalized search: AI algorithms are increasingly used to learn the user interests, behavior and provide personalized and effective search results.
- Explainable AI: The AI algorithms should be able to explain the results and reasons for providing the output. This helps in auditing, governance and help in complying with any regulations.
- Cognitive computing: The term refers to emerging set of technologies led by AI to provide more contextual, interactive and personalized information.
- Reinforcement learning: Unlike supervised learning and unsupervised learning, reinforcement learning "learn" through experience and mistakes without any explicit rules and training.
- Convergence with IoT, Blockchain and other emerging technologies: The convergence is key to some of the popular applications such as autonomous vehicles.

☞ Check Your Progress 1

1. _____ is mainly used in low code no code applications.
2. Machines can perceive the images through _____
3. Conversational Interface interact with humans through _____.

14.4.3 Blockchain

Blockchain technology provides a decentralized, distributed database that provides a non-tamper able record keeping facility. Blockchain is built using peer-to-peer system and provides cryptographic security. Blockchain can be used as a ledger for transactions that provide immutable and publicly viewable record of the transactions. The blockchain technology provides a way to record the legitimate transactions in "blocks" that are chained to previous blocks. Bitcoin cryptocurrency uses blockchain technology.

Given below are the key features of blockchain:

- Blockchain provides a permanent and distributed record of all genuine transactions
- Blockchain provides a decentralized governance
- The data/record in the blockchain is immutable (that cannot be altered) and irreversible due to the chained and distributed nature of the blocks.
- The blockchains provide transparent way to track the records and transactions.
- Blockchains also provide secure way for performing the transactions. As the copies are distributed across the entire chain, any attempt to alter the block's content can be easily identified by the remaining chain of blocks.
- The data in the block can be updated only upon consensus among the participants.

Given below are the key applications of blockchain:

- Blockchain can be used to store legal contracts that cannot be tampered.
- A company's Product inventory information can be stored in Blockchain.
- The supply chain information can be stored in blockchain to track a product throughout its lifecycle.
- Cryptocurrencies can be managed using blockchains
- Medical records can be secured stored using blockchains.
- Property records can be accurately stored and managed using blockchains.

14.4.4 Internet of Things

Internet of Things (IoT) represent a network of objects (known as "things") that have embedded entities (such as sensors, software, devices etc.) so that the objects can track, monitor, collect and exchange data. By embedded a tracking sensor we can collect the information of the object such as the temperature, position (latitude/longitude), speed and such. The collected information can then be analyzed to predict, forecast and conduct preventive maintenance activities.

With the emergence of 5G technology, it would be possible for all devices to be interconnected in the IoT world.

Given below are the key components of IoT:

- Sensors: The sensors are essential to collect the information about the object. Sensors could collect the information such as light, temperature etc. from the object.
- Communication: Sensors communicate the collected information to the cloud platform
- Data Analysis: The incoming data stream is analyzed to understand the trends, patterns, insights. The analysis is further used for prediction, recommendation, forecasting, analytical reporting and such. We use various data analysis methods such as machine learning methods, big data analysis methods to make sense out of sensor data.

The figure 14.3 provides a sample IoT ecosystem wherein the sensors feed the data to the analytics software running on IoT gateway. The analyzed data is then fed to the enterprise ecosystem for further analysis and reporting.

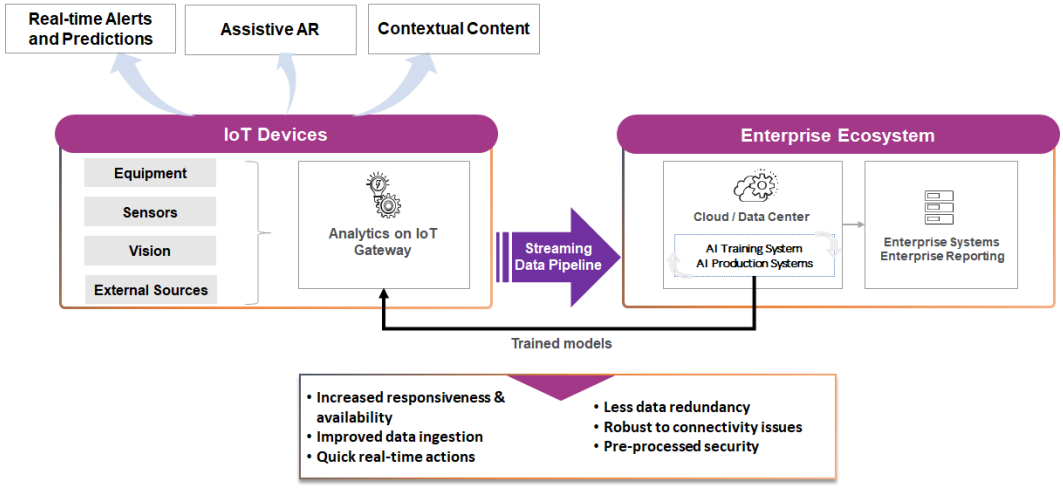

Figure 14.2 : IoT Ecosystem

Given below are the key advantages of IoT:

- Increased user experience: Through real-time tracking of the objects/things, we can provide a more accurate prediction/estimation to the end users thereby improving user experience.
- Cost Optimization: Regular monitoring of the machinery enables us to carry out preventive maintenance thereby reducing the machine downtime.
- Remote monitoring and control: IoT technologies enable us to monitor and control the devices remotely in real time.
- Advanced analytics: IoT provides us with advanced analytical information so that we can pro-actively act on the information to reduce the cost.

14.4.5 Augmented Reality/Extended Reality

Augmented Reality (AR) enhances/augments natural world objects by overlaying computer-generated information. AR uses multiple technologies such as AI, computer vision, analytics and such. AR aims to make the physical world objects more interactive by superimposing the image, video, text, and sounds onto the real world objects.

One popular AR application is that of a mobile app wherein the mobile app could overlay a new set of clothes for the user to understand how the new clothes fit. Few other AR-enabled mobile apps could virtually paint the house with various colors so that the user could choose the best suited color. Pokemon GO was one of the popular mobile games that heavily used AR wherein the Pokemon characters are overlaid onto the real world objects.

Figure 14.4 provides the key AR modules used in mobile app for adding an image to the real world. OpenCV, Google Lens, ARCore and Google ML toolkit are some of the popular tools used in the AR.

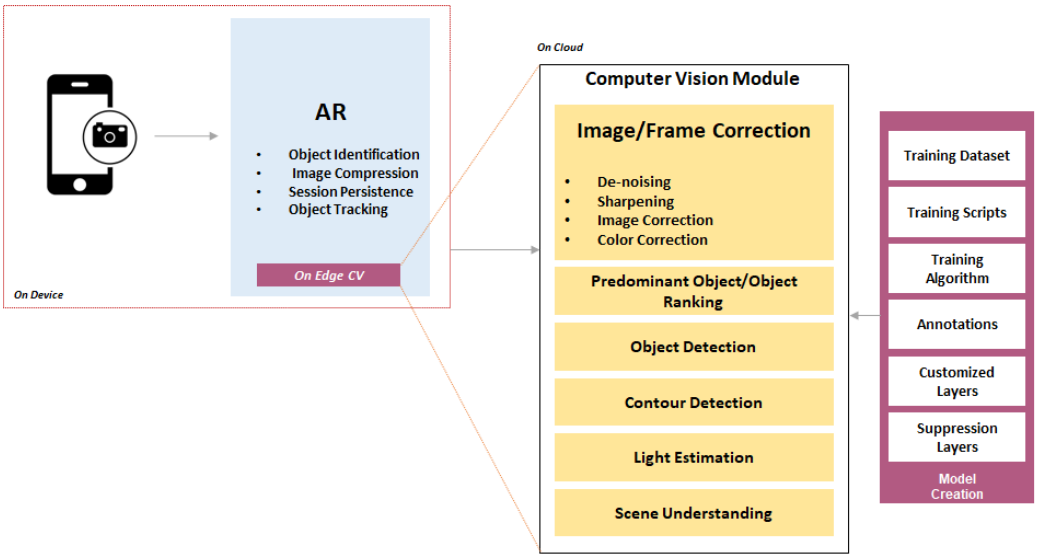

Figure 14.3 : Sample AR modules on Mobile and on Cloud

As depicted in the figure 14.4, AR Module takes environmental/mask information from CV & performs semantic interpretation. The computer vision Module performs all necessary preprocessing on the individual frames required for the AR Module to work on. This is the stage in the pipeline where data cleaning and refinement takes place. This is done on-device as well as at cloud. The ML Model Training basically feeds in the initial trained model to the CV Module. Additional layers/Suppression layers have been added to make the ML model less cumbersome for mobile devices which would perform the real-time visualization on edge.

14.4.6 Containerization

Containers are virtualized environments that provide independent, portable computing environments. With containers we could implement "write once, run anywhere" concept. Containers consist of all software needed to execute the applications including the libraries, tools, programming language, runtimes, dependencies, configurations and others. Containers abstract the application from underlying OS. Docker is one of the most popular container technologies. A docker image can be created using all the necessary binaries and dependencies needed for the application; these dependencies could include runtimes, database, application server and such. For development related activities we could also create Docker image consisting of IDE, debugging tools, browsers, SQL developer tools and others needed for development activities. Due to its flexibility, containers are the preferred choice for deploying microservices and provide elastic scalability.

Given below are key features of containerization:

- Portability: We could easily port the containers from one environment to another. Irrespective of the underlying OS (such as windows or Linux), various infrastructure methods (on-premise or cloud or virtual machine), we could easily deploy to the containers.
- Independence and self-contained: Containers are self-contained and independent platforms that
- Management and security: We could manage the computing resources such as CPU, memory and DNS name, load balancing, and sensitive confidential information easily with containers.

Given below are key advantages of containerization:

- Performance: As containers are light weight virtualization engines, they offer better performance and less overhead when compared to traditional virtual machines.
- DevOps Friendly: Container technology fits well into the modern DevOps methodology wherein we can deploy the artefacts to the Docker containers.
- Flexibility and Scalability: Container orchestration platforms such as Kubernetes provide options for elastic scalability (that is to automatically spin up new Kubernetes pods to serve the increased load based on configured memory and CPU thresholds).
- Fault isolation: If any of the containers goes down, it does not impact other containers sharing the OS. Container orchestration engines can isolate the faulty container and restart or spin up a new container.
- Agility and speed: Container ecosystem provides tools and technologies to easily configure, start or spin up a new container on demand.
- Efficiency: Being lightweight, containers are lot more efficient than other virtualization platforms. Containers are also better at scaling, ease management, enable faster roll out of new updates, enable logging, enable monitoring, and provide security.

Figure 14.5 provides various components of container based solution.

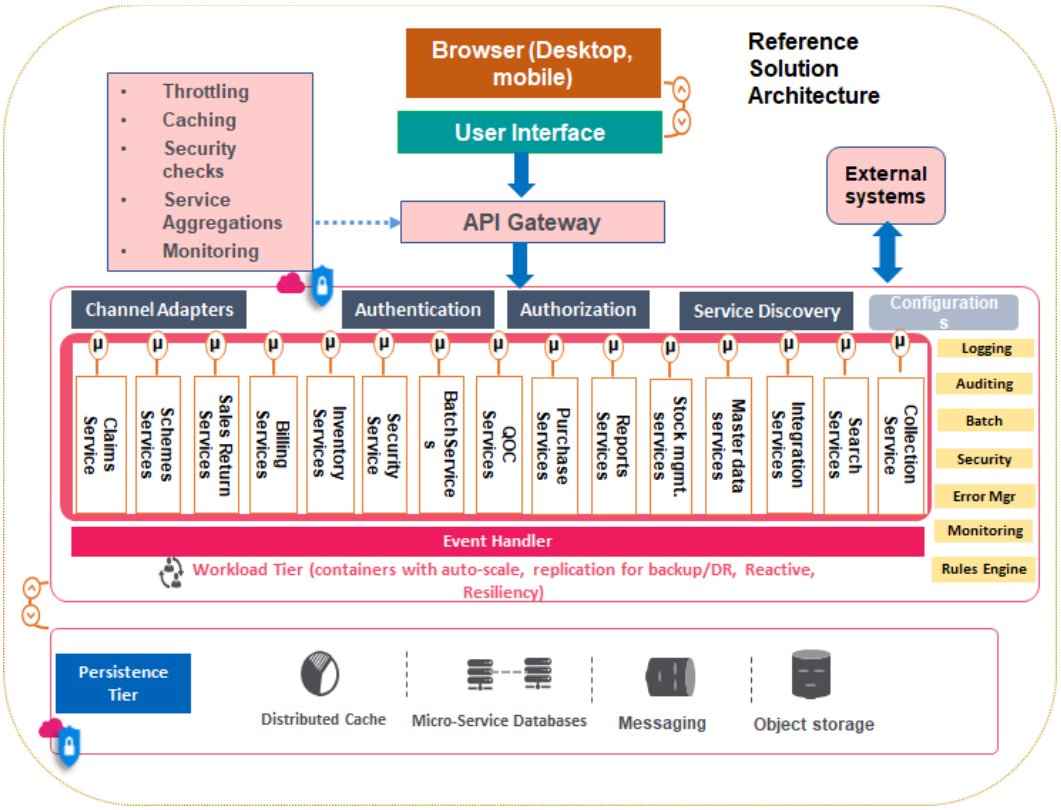

Figure 14.4 : Sample Container Based Solution

14.4.7 Continuous Delivery

The traditional waterfall development and deployment model needs higher time to market and is prone to delivery and quality risks. With the increasing user demand, market expectations and business changes, it is imperative to roll out the new release iteratively and frequently. Agile delivery implements the best practices of continuous delivery model that delivers the solution in a 2-week sprint cycles. With each delivery an incremental update is pushed to the production. Through continuous deployment, we deploy the updates automatically to the production on periodic basis. Continuous delivery model makes the enterprises responsive to changes, implements the feedback quickly, provides the features/fixes faster and reduces the delivery risk.

Continuous Integration (CI) is closely associated with continuous delivery (CD) wherein the application is integrated frequently and iteratively to reduce the integration risks. We can implement the CICD through popular tools such as Jenkins or Azure DevOps.

Figure 14.6 depicts a sample Jenkins based CICD pipeline. In the development stage, the GitHub is used for code merge and code review activities. During testing we could configure SonarQube based static code analysis and Junit based unit testing as part of Jenkins jobs. Fully tested artifacts are pushed to S3 for packaging. Maven can be used for building artifacts automatically. Finally during the deploy stage, the artifacts are deployed to the environment. We could configure AWS CloudWatch to monitor the application.

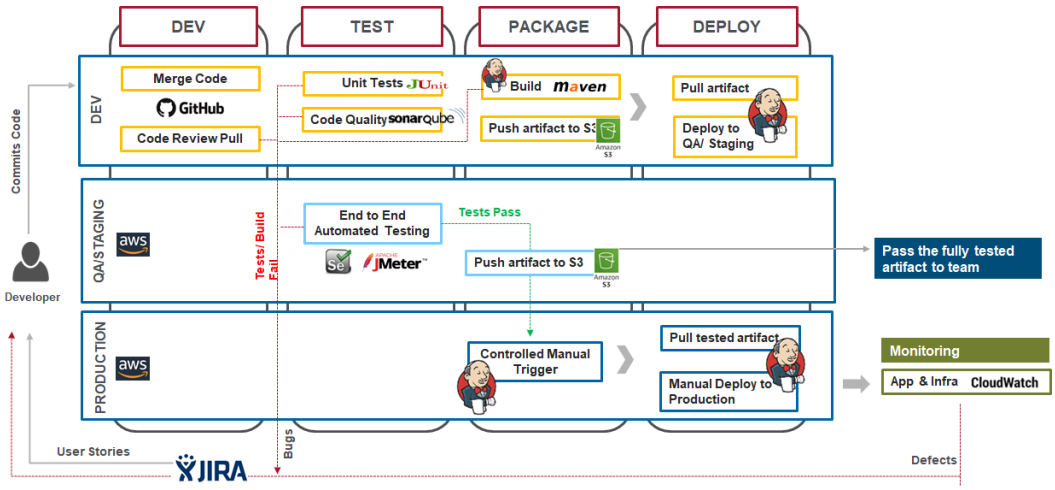

Figure 14.5 : Sample Jenkins based CICD Pipeline

The key metrics during agile delivery are given below:

- Defect Leakage: Defects that are found outside of a sprint through Acceptance, Integration, Performance, and Usability (AIPUS) testing
- Code Complexity: Typically, an automated test done with tools like SONAR that looks at coding standards and Cyclomatic complexity
- Burnup Charts: A look at project progress. The average delivery timeline is compared to the actual delivery progress
- Commit to Complete: The ratio of committed stories to completed stories
- Code Coverage: The percentage of unit test cases written
- Velocity: The number of story points that are completed in a Sprint

We have depicted various activities in a 2-week sprint in Figure 14.7.

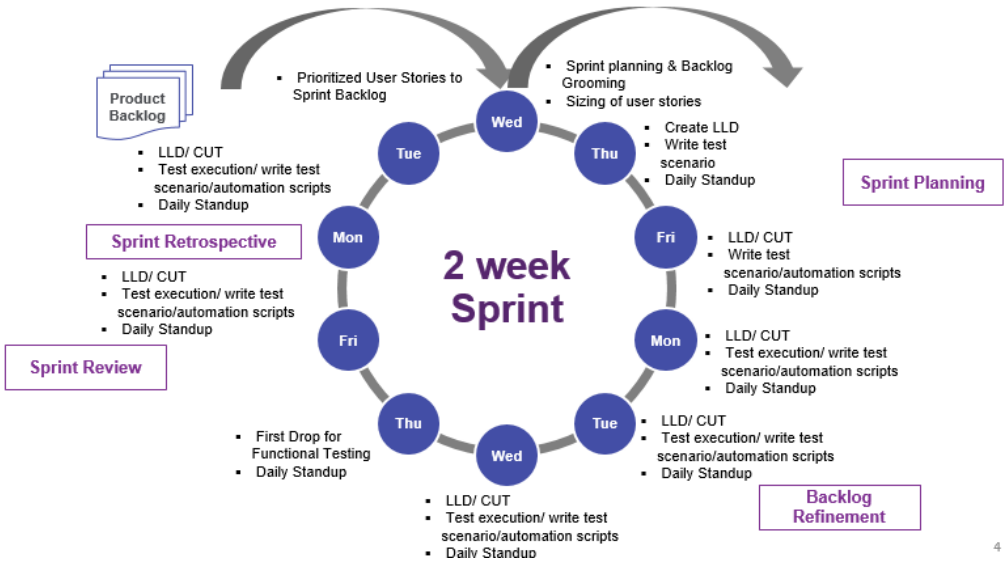

Figure 14.6 : Sample 2 Week Sprint

We have depicted a typical 2-week sprint flow and various activities, owners and metrics in Figure 14.8.

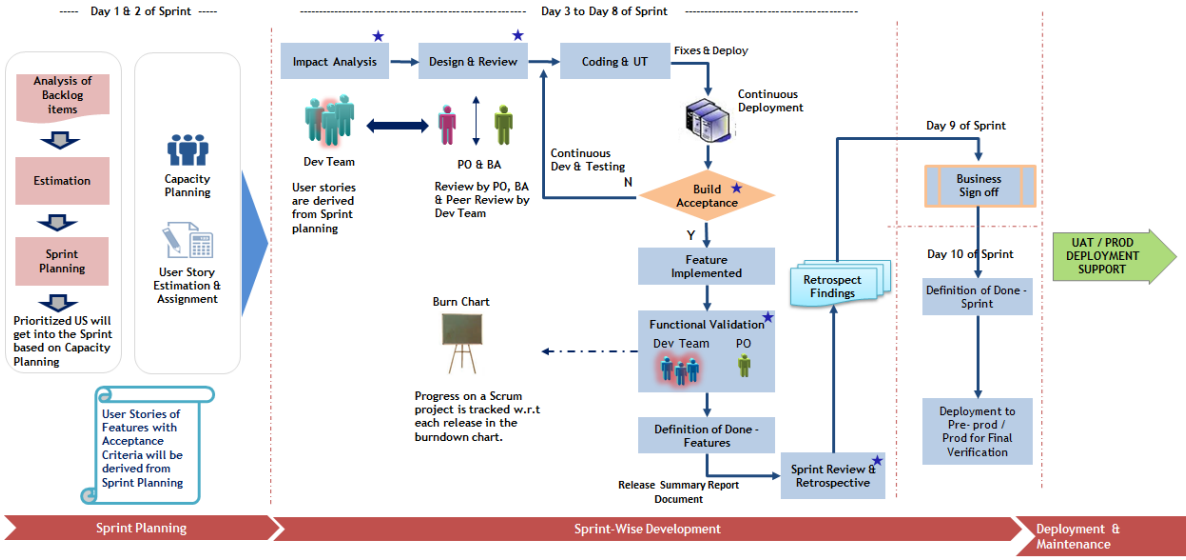

Figure 14.8 : Sprint in Action

14.4.8 Cloud Platforms

Cloud computing has redefined the way software applications are designed, built, deployed and monitored. Cloud platforms provide the computing resources such as servers, databases, network, storage on demand offering "pay-as-you-go" model. Enterprises can innovate faster, reduce the IT costs and build agile models using the cloud platforms. Microsoft Azure, Amazon Web Services, Google Cloud are the popular cloud platforms. Enterprises can use public cloud (owned by third party cloud providers), private cloud (enterprise-owned computing resources) or hybrid cloud (data shared between public cloud and private cloud)

Given below are the main types of cloud services:

- Infrastructure as Service (IaaS): Cloud platforms provide infrastructure components such as virtual machines, servers, database, operating systems and such.
- Platform as Service (PaaS): Cloud platforms provide an end to end environment for development, testing and deployment of software.

Developers can build their applications using the provided platform. PaaS platforms typically include operating systems, development tools along with the underlying infrastructure components.

- Software as Service (SaaS): In this model, the software applications are available over the cloud typically on subscription basis. The SaaS providers manage the security, infrastructure and maintenance of the software.

Given below are the key advantages of cloud services:

- Cost: Due to the metered billing of cloud platforms, enterprises will only pay for what is used and can avoid the upfront IT cost.
- Scalability: Cloud platforms provide elastic scalability based on the demand and load
- Performance: The cloud platforms offer state of the art computing infrastructure that are continuously patched and upgraded to ensure high performance.
- Availability: The cloud services offer very high availability due to the distributed nature of the computing resources and redundancy.
- Security: Most of the cloud platforms provide inbuilt methods, tools to safeguard against the security attacks.

We have depicted a sample cloud solution in the Figure 14.9.

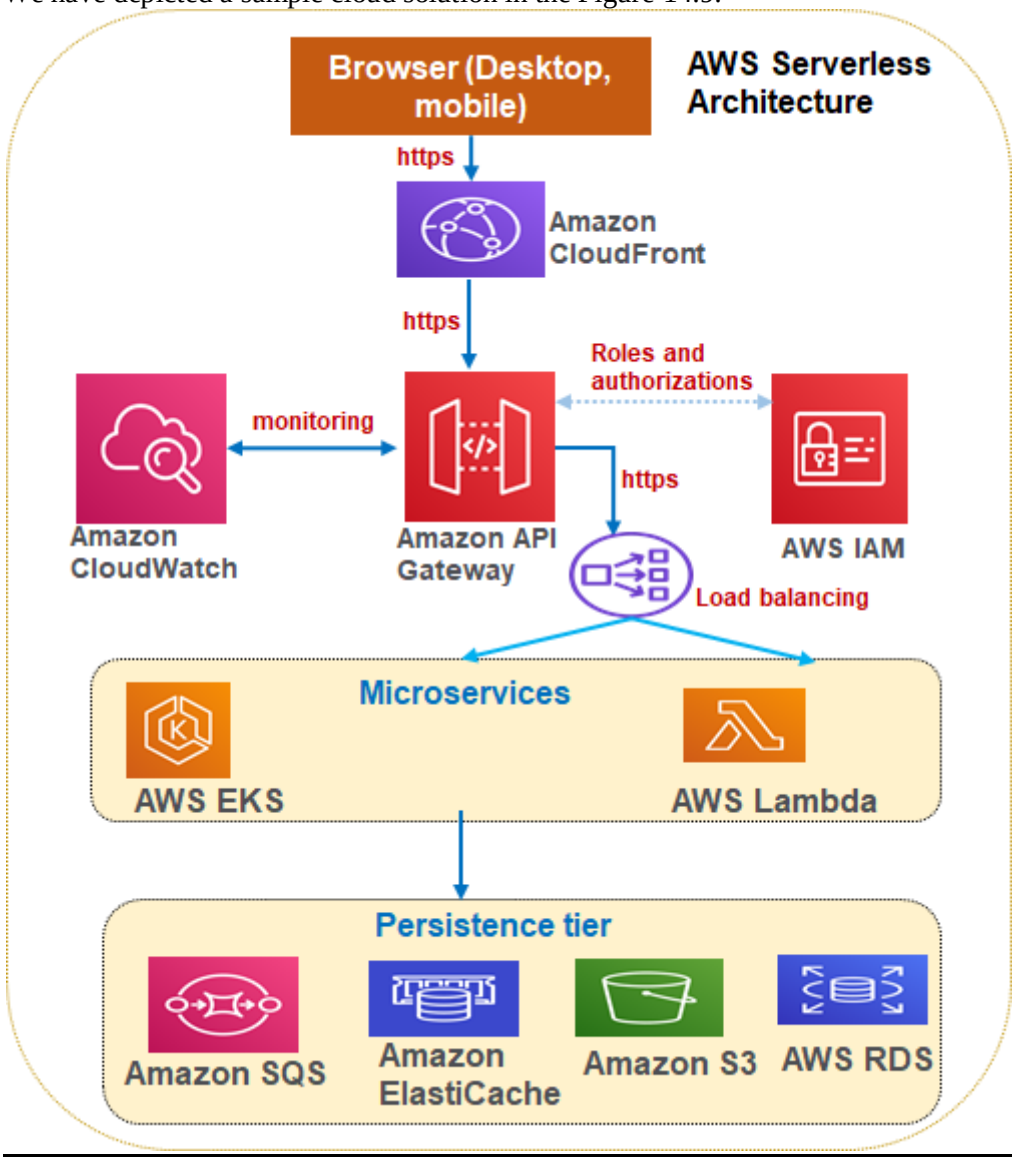

Figure 14.9 : Sample Cloud Solution

14.4.9 Big Data

The field of Big Data involves management, analysis and reporting of massive amount of data that cannot be handled by traditional data management software. With each growing data, we are generating and consuming huge volumes of data through social media posts, comments, pictures, videos and such. Storing, retrieving, managing and searching this data needs a different way of managing the data. Big Data systems are eventually consistency as per CAP (Consistency, Availability, Partition) theorem.

The sensor data and social media data are few examples of the big data. Big data technologies handle the ingestion of large volume of data, sharing the data, searching the data, updating the data, making the copies of data, reporting of data and such. Big data applications include product recommendation, consumer behavior analysis, sales forecasting, predictive analytics and such.

The three key attributes of Big Data are:

- Volume indicating the massive amount of data generated to qualify as Big Data. Though the definition of Big Data is changing with time, at this point we consider anything above 1 Terabyte as big data.
- Velocity indicates the speed at which the data gets generated
- Variety indicates the format of the data that gets generated. This includes structured data, semi-structured data, and unstructured data.

Due to semi-structured and unstructured nature, Big Data is managed in NoSQL databases such as DynamoDB, MongoDB. The NoSQL databases aim for eventual consistency instead of strict consistency.

Apache Hadoop, MongoDB, Apache Mahout, Apache Spark are some of the most popular technologies used for managing Big Data.

Given below are key scenarios for Big Data:

- Manage and process sensor data coming from IoT devices
- Store and search the social media data or user generated data
- Manage unstructured (such as text or images or video) data that needs flexible schema.
- Capture and manage customer behavioural data from various sources such as social media, CRM system and others.
- Capture and manage user transaction data for recommendation engines.
- Data needed for providing personalized and contextual web-scale search.

Besides these scenarios, there are numerous use cases that need Big Data systems such as data needed for predicting customer churn, fraud detection data, risk assessment data,

14.4.10 Software Development Methodologies

Various development methodologies are used to develop and deploy the modern applications. We have given the most popular ones below:

- Cloud first approach in which we develop the applications using cloud native technologies. The applications are developed, tested and deployed on cloud platforms.
- Mobile first approach in which the applications are built for mobile devices primarily. This includes the mobile apps or web applications using responsive web design (RWD) methodologies
- AI First approach in which the AI methods and machine learning is adopted to gain insights from data in all the service offerings.

- Offline first approach in which the applications are built for handling network unavailability issues using local storage mechanisms. Progressive web applications (PWA) provide offline capabilities.

Check Your Progress 2

1. Blockchain provides a _____ governance
2. The key components of IoT are sensors, communication and _____
3. _____ abstract the application from underlying OS
4. Normally Agile delivery involves _____ sprint cycles
5. The cloud service type that provides provide infrastructure components such as virtual machines, servers is called _____

14.4.11 Other Emerging Trends

We have compiled few other emerging trends below:

Serverless Computing

Developers can quickly develop and deploy the business functions without bothering about the underlying server, deployment, capacity and others. AWS Lambda, Azure Functions are some of the examples of serverless computing.

Cross Platform Development

In cross platform development, we could develop the code only once and then reuse the code for various mobile platforms such as Android, IoS, Windows, and Linux etc. Figure 14.10 provides a sample architecture of Xamarin based cross-platform solution. Many cross platforms provide inbuilt widgets, API integrations and portability features. Xamarin and Google Flutter are popular cross platform development technologies. We have depicted a sample Xamarin cross platform solution in figure 14.10

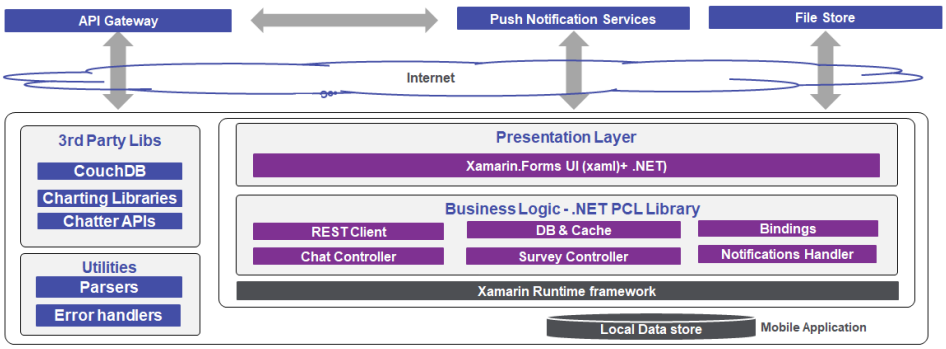

Figure 14.7 : Xamarin based Cross Platform Solution

Conversational Interfaces

AI-powered chatbots provide capabilities to understand and respond to natural language queries. Conversational interfaces can be customized to handle the functional domain specific data. Conversational interfaces are handling the first-level queries in many sectors such as banking, retail and e-commerce domains. We have depicted the high level components of a conversational interface in the figure 14.11.

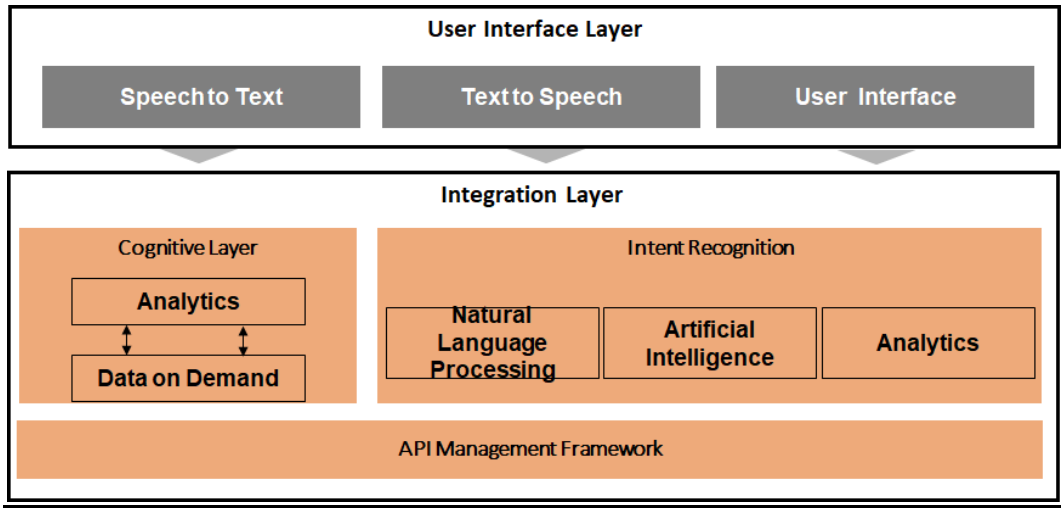

Figure 14.8 : Conversational Interface

14.5 SUMMARY

In this unit, we started discussing the main drivers of innovation. After briefly discussing the key trends in software engineering, we discussed the low code and no code platforms that provide visual tools for rapid application development. AI and machine learning methods are adopted in almost all areas of software development to provide automation and data analysis. We looked at blockchain technology for distributed data handling. IoT ecosystem provide real time continuous monitoring of the devices and make them smart. AR technology augments the real world with digital objects. Containerization provide portable technology for application development and deployment. Continuous Delivery provides an iterative and continuous development and deployment. Big data technologies are designed to handle huge volume of data.

14.6 SOLUTIONS/ANSWERS

Check Your Progress 1

1. visual programming.
2. computer vision.
3. natural language processing.

Check Your Progress 2

1. decentralized.
2. data analysis
3. Containers
4. 2 week
5. Infrastructure as Service

14.7 FURTHER READINGS

References

https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/#benefits

https://en.wikipedia.org/wiki/Big_data

UNIT 15 INTRODUCTION TO UML

Structure

15.0 INTRODUCTION

The UML diagrams are used for modeling the object oriented design. UML stands for Unified Modeling Language. UML provides a standards-base, general purpose modeling language that helps everyone to understand various aspects of the system. As UML is language independent, we can use UML for many different languages such as Java, Python, C sharp and others. UML provides the software blueprint in the visual representation.

15.1 OBJECTIVES

After going through this unit, you should be able to

- understand key concepts of UML,
- understand the object oriented design,
- understand the difference between structural UML diagrams and behavioral UML diagrams,
- learn to model the class diagram, sequence diagram, component diagram and deployment diagram
- understand state chart diagram, use case diagram, activity diagram

15.2 BACKGROUND OF OBJECT ORIENTED DESIGN

Function-oriented programming emphasize on algorithms. A function or procedure is the basic building block in function-oriented programming. This is mostly a top-down approach. The function-oriented programming suits well for relatively smaller and

less complex systems. Due to inherent coupling and behavioral relationship, usually there will be high complexity.

In the object-oriented programming, the data is exposed only through the interfaces. The object-oriented programming follows a bottom up approach. The object normally represents a logical unit or a real-world object. The object state contains all the object properties/attributes and the behavior of an object is expressed through its functions or methods.

The problems that are inherently complex are well suited for object oriented programming. The object oriented programming is designed for change, allows reuse, improves productivity, speeds up development time and improves maintainability. We can build scalable and modular applications with object oriented programming.

15.3 KEY CONCEPTS OF OBJECT ORIENTED DESIGN

In object oriented programming, the solution is composed of objects that are instances of classes interacting with each other via relationships. Objects represent real world entities. Objects encapsulate data and functions/behavior that control the data. While objects are the basic building blocks of object oriented design, classes are the blueprints of the objects.
The object oriented programming focuses on decomposition, encapsulation, abstraction, modularity and inheritance.

The abstraction hides the inner details of the entity and exposes the behavior through interface. It defines the clear boundaries of the object and provides the required characteristics and behavior of an object. Abstraction can be implemented through a well-defined interface. Abstraction plays a key role in the object modelling providing the integrations only with the exposed interfaces and hides the user from the inner details of the component.

The modularity partitions the individual components based on their responsibilities so that the modules can be easily reused and extended. The system is decomposed into loosely coupled cohesive modules.

In the **encapsulation based design**, the data is exposed only through the functions and the class provides methods for binding data. Encapsulation eliminates the direct dependencies between objects and hence avoids tight coupling. Encapsulation is also called data hiding or information hiding.

In Inheritance the child class obtains the characteristics and behavior of its parent class. It is possible to model the hierarchy through inheritance. You can also change the behavior of the subclass through overloading and overriding.

In **composition** one object are composed of other objects.

UML diagrams are effective way to visually model the object oriented design as the UML diagrams can depict inheritance, composition, modularity and abstraction.

15.4 INTRODUCTION TO UML DIAGRAMS

UML model was released in 1997 as a common design language building and modelling software applications. UML model provides the ways to depict the complex the interaction between components, the behavior, and sequence, static and dynamic nature of the software components. UML models are independent of the programming language.

UML diagrams visually model and depict the system from various perspectives such as use case, implementation, process and deployment. We can visually depict, model, document and build the system using UML diagrams.

Definition

A Unified Modeling Language is a structured language that provides the commonly agreed vocabulary to communicate, model and document the structural and behavioral aspects of the software system. UML is widely used as a blueprint to design, develop, document, analyze, and comprehend the software systems.

Importance of UML diagrams

When modeling the architecture of the overall software system, it is important to understand and analyze the system from various perspectives. A software architect has to understand the implementation aspects of the application, overall structure of the application, the system topology, flow of the messages and such. A model abstracts the system that helps in easier understanding. UML diagrams is a popular way to model the system. UML is also supported by popular Integrated Development Environments (IDE) such as Eclipse. Given below are the key advantages of the UML diagrams:

- UML diagrams model the big picture of the entire application and helps us to analyze the application from various dimensions.
- UML diagrams provide a common language and vocabulary to understand, model, document and discuss the system design.
- UML diagrams helps us to understand the conformance of the system to business and technical requirements.
- The team gets the clear understanding of the main classes, components, interfaces and their relationship through UML diagrams. This helps the developers to develop the low-level design faster and implement the system quicker.
- During impact analysis or during major bug fixes or during system enhancements, we can use the UML diagrams to analyze the key impact areas to come up with optimal design.
- We can implement the key architecture best practices such as loose coupling, layered architecture, and separation of concerns by modeling the system in UML.
- UML diagrams make the system easy to develop, maintain and change.
- We can document the system from various perspectives through the UML diagrams. UML diagrams also document the key architecture decisions.
- UML diagrams help us visualize the overall structure and behavior of the system and its various components.

Classification of UML Diagrams

We can broadly classify UML diagrams into two categories - static diagrams or structure diagrams and dynamic diagrams or behavior diagrams.
Static or structural diagrams represent the static physical elements of the system. Static diagram includes class diagrams, object diagrams, component diagrams and deployment diagrams. The dynamic or behavior diagrams depict the runtime and dynamic behavior of the software systems. The dynamic diagrams include sequence diagram, collaboration diagram, state diagram and activity diagram. We have depicted the main categories of UML diagrams in Figure 15.1.

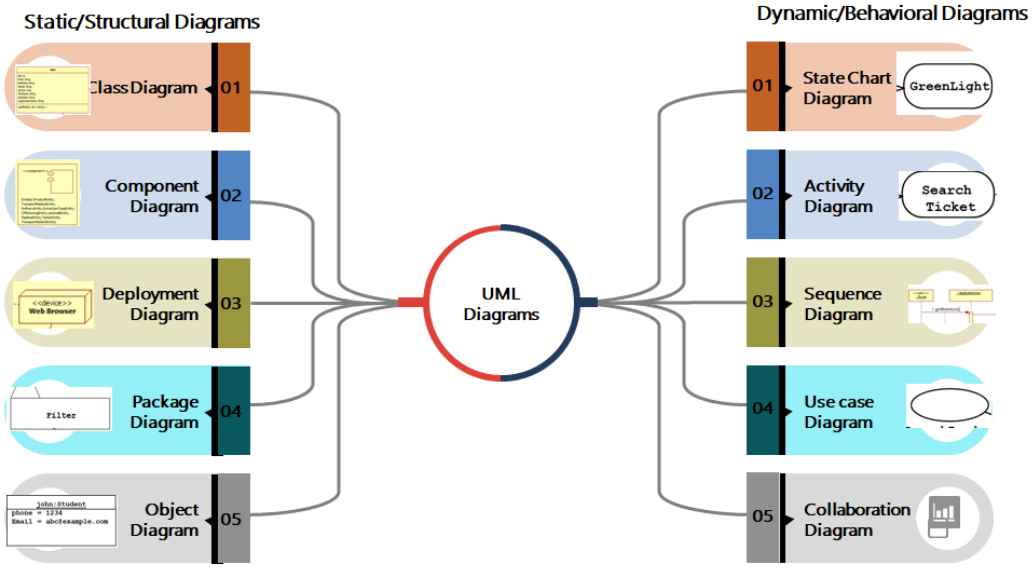

Figure 15.1 : Categories of UML diagrams

Dynamic or behavioral UML Diagrams

Dynamic diagrams or behavioral diagrams depict the dynamic and moving parts of the system. The main behavioral elements depicted in the dynamic UML diagram are object state, control flow, interaction, state machines and such. The main dynamic UML diagrams are sequence diagram, activity diagram, state chart diagram, use case and collaboration diagram.

The **sequence diagram** models the behavior of a single use case or a single scenario. The sequence diagram visually depicts how each objects interact with each other. In the sequence diagram we have a time ordered messages where vertical dimension represents the time axis and horizontal dimension represents the roles.

The **activity diagram** describes the business logic the business process and work flows. The activity diagram models the use case.

The **state diagrams** describe the systems behavior through state transitions. The state diagram depicts all possible states of an object when an event occurs.

Static or structural UML Diagrams

Static or structural UML diagrams depict the overall structure of the system that remains static. The main structural elements depicted in the UML diagram are classes, objects, nodes, interfaces, use cases, collaboration and components. The main static UML diagrams are package diagram, class diagram, component diagram, deployment diagram and object diagram.

A **class diagram** depicts the classes along with its attributes and operations. The **object diagram** depicts the instance of class encompassing the state.

The **package diagram** organizes the elements of system into logical groups to minimize the dependencies among them.

The **component diagram** represents the physical modules, known as components of the solution. The components are logical, independent autonomous and encapsulated units of a system and exposes one or more

interfaces. The component diagram depicts the relationship between various components of the system.

The **deployment diagram** defects the relationship between the software and hardware components of the system. The nodes in the deployment diagram represent servers or other computational elements. Deployment diagram depicts how the components are organized in the system

Views Covered by UML Diagrams

We can comprehend a complex system only by looking at various aspects (also known as views or perspectives) of the system.

The UML diagrams cover various views. We have depicted various views covered by the UML diagrams in Figure 15.2.

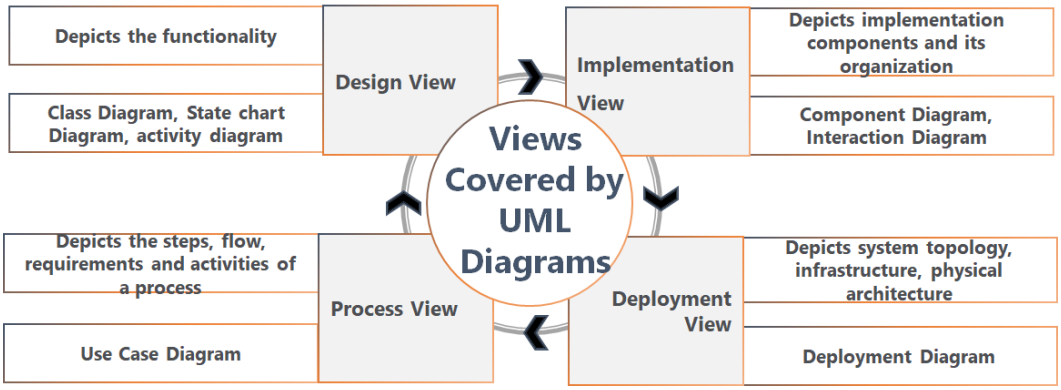

Figure 15.2 : Views covered by UML diagrams

As shown in figure 15.2, it is important to model the system from various views such as implementation, design, deployment and process. Each view provides the specific abstraction of the system. We can understand the system holistically by analyzing all the views.

The design view essentially captures the functionality of the system. The static part of the design is depicted through class diagram and the dynamic behaviour is captured in the activity diagram and state chart diagram.

The implementation view depicts the overall implementation components such as components (static aspects) and interactions (dynamic aspects) of the application. The implementation view specifies the physical view of the system.

In process view, the UML diagrams primarily depict the sequence of steps and flow and components involves in a use case. The Use case diagram models the requirement of the system. We can understand the throughput and scalability of the system through the use process view.

The deployment view covers the physical architecture of the system. We can understand the overall distribution and topology of the application components. We can assess the capacity and sizing of the overall solution in the deployment view.

Extensibility of UML Diagrams

We can use the below given features to extend the UML diagram:

- **Stereotypes** allow us to extend the UML vocabulary by adding the application specific elements. For example we can add the logging framework details for the application using stereotype
- **Constraints** allows us to add the business rules and conditions. For example we could specify the value of "date" attribute to be greater than 01-Jan-1970
- **Tagging** allows us to specify the additional metadata for the object. For instance we can specify the details such as author name, version using tagging.

15.5 STRUCTURAL UML DIAGRAMS

Structural elements depicted in the UML diagrams are the static parts of the system. Given below are the key structural elements depicted in the UML diagrams; normally they are "Nouns" of the system.

- **Class**: It is logical unit that encapsulate the state (through attributes), and behavior (through operations) of the objects and can optionally implement one or interfaces. A class can interact with other classes through relationships such as generalization, composition and such. A class is depicted in rectangle as shown in the Figure 15.3.

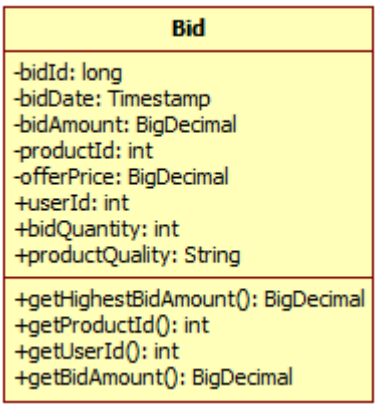

Figure 15.3 : Class

- **Interface**: An interface specifies the contract for the implementation through the operations. The implementation class has to implement the operations of the interface. We have depicted the interface in Figure 15.4.

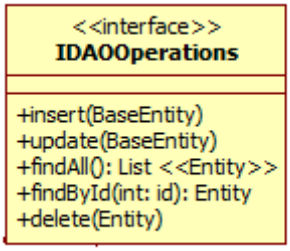

Figure 15.1 : Interface

- **Object**: An object is the instance of the class. Objects interact with each other by passing messages and expose the data through operations. We have depicted the Object in the figure 15.5.

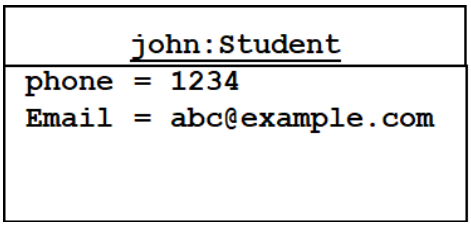

Figure 15.5 : Object

- **Component**: A component represents a logical module that exposes interfaces for interaction. The component is represented in Figure 15.6

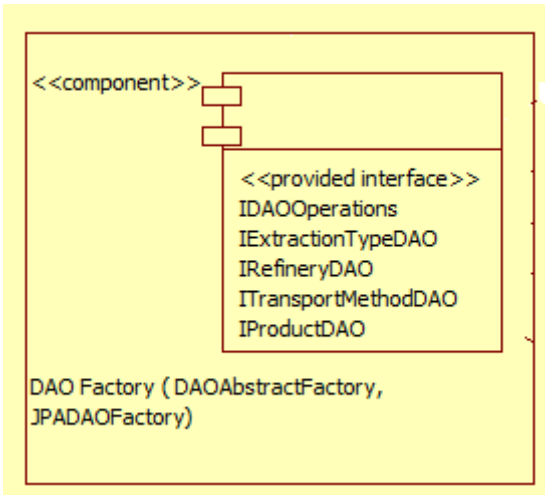

Figure 15.6 : Component

- **Use case:** A use case represents a specific functionality and represents the flow for implementing the use case. A use case is represented in Figure 15.7.

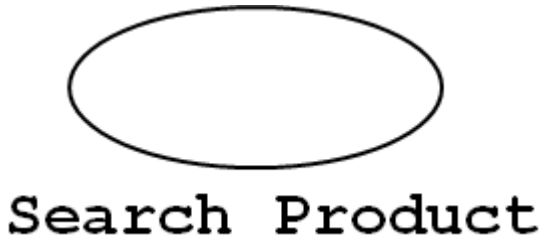

Figure 15.2 : Use case

15.5.1 Class diagram

The class diagram visually provides the object oriented design of the system and depicts the relationship between implementation classes. The class diagram visually provides the inheritance, composition, association relationship within classes.
In the class diagram we depict the class name, attributes, visibility (public, protected, private), associations and operations and optionally comment.

The relationships in the class diagram can be of the following types:

- **Association** relationship depicts the set of related links among the objects. For example a single "department" objects can be associated with multiple "employee" objects. Composition is type of association wherein a single object contains multiple dependent objects that cannot independently exist leading to whole-part kind of relationship. In aggregation relationship, a single object consist of multiple independent objects that can exist without the parent object. The association between "Department" class and "Employee" class is depicted in Figure 15.8. The association relationship indicates that each department has many employees. Aggregation is a weak form of association whereas composition is a strong form of association.

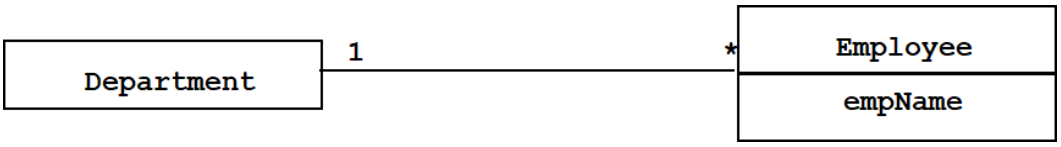

Figure 15.3 : Association Relationship

- **Dependency** relationship depicts the dependency between two elements (dependent and independent elements). When the independent element changes, the dependent element also changes.
- **Generalization** depicts the inheritance (parent-child relationship) in the class diagram. The child inherits the operations and attributes of the parent. The subclass is a specific kind of the super class.
- **Realization** depicts the contract between two entities. Normally we depict the realization relationship between interface (that specifies the contract) and the implementation class (that implements the contract).

The class diagram depicted in Figure 15.9 provides the details of three classes: User, Seller and Buyer.

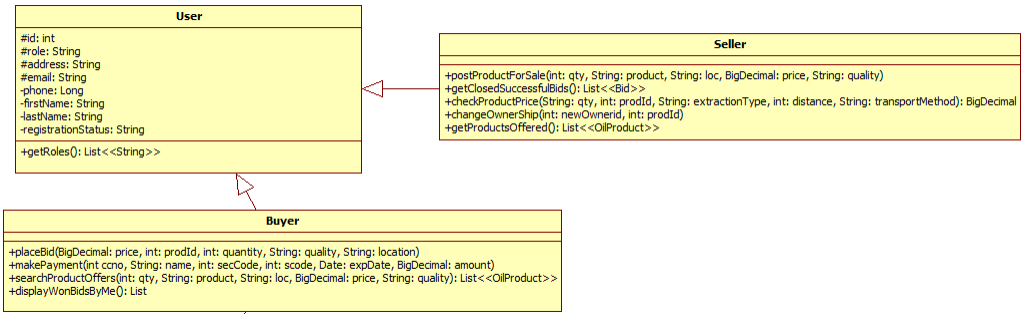

Figure 15.9 : Class diagram with Inheritance

The User class is the parent class that has attributes such as “id” and “role” are protected whereas the attributes such as “firstName”, “lastName” are private. The User class has a public method “getRoles ()” the return the list of String objects. The Figure 15.9 also depicts the generalization relation wherein the “Buyer” and “seller” classes are sub/child classes of “User” class.

The interface and its implementation is depicted in Figure 15.10.

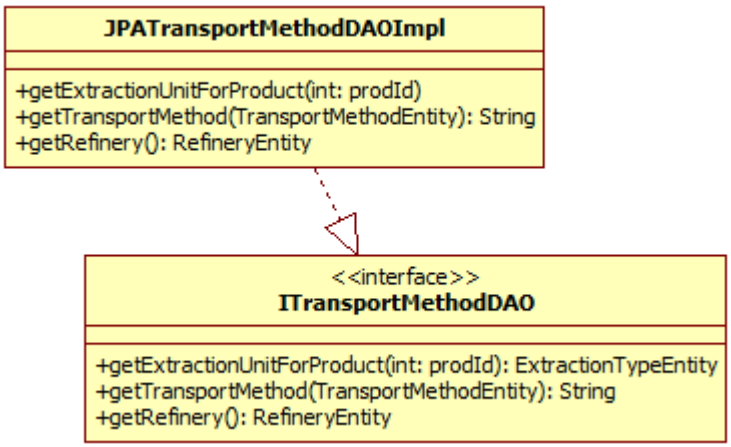

Figure 15.4 : Interface and its Implementation

The interface ITransportMethodDAO specifies three operations as part of contract and the implementation class JPATransportMethodDAOImpl provides the complete implementation of the three operations.

15.5.2 Object diagram

The object diagrams depict the instantiated objects and their state. The object diagram also represent the relationship between objects. We have depicted an object diagram in Figure 15.11.

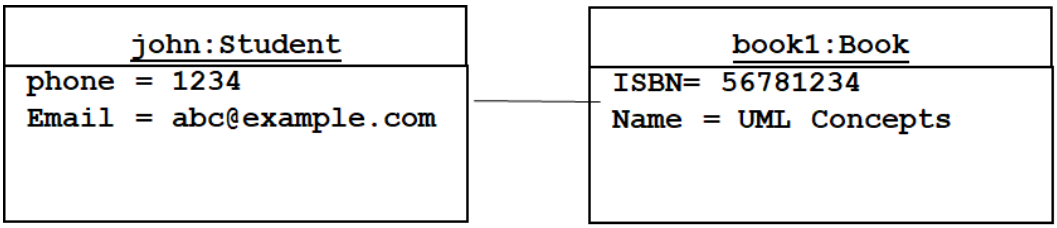

Figure 15.5 : Object Diagram

15.5.3 Component diagram

A component represent logically related entities such as classes and their dependents that interact with other components through well-defined interfaces.
The component diagram provides the physical view of the application wherein we model the software components and their dependencies on other components. The component diagram depicts the organization of physical entities of the application. We can depict the high order solution components or packages in the component diagram.

The component is depicted in Figure 15.12.

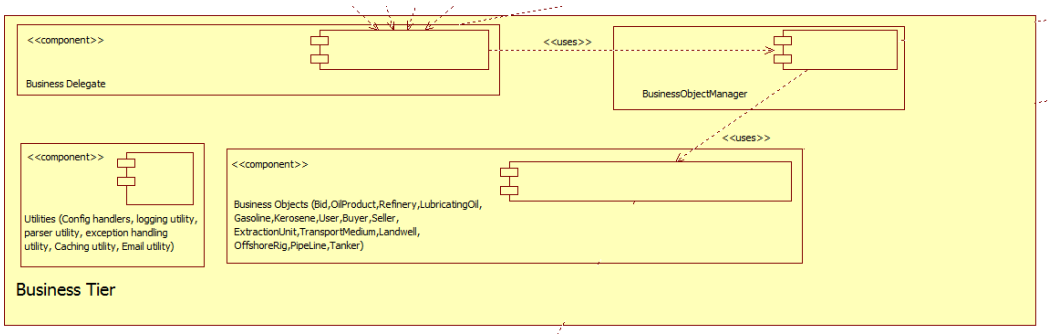

Figure 15.12 : Component Diagram

We have also depicted the interfaces exposed by the component and its interaction with other component in Figure 15.13.

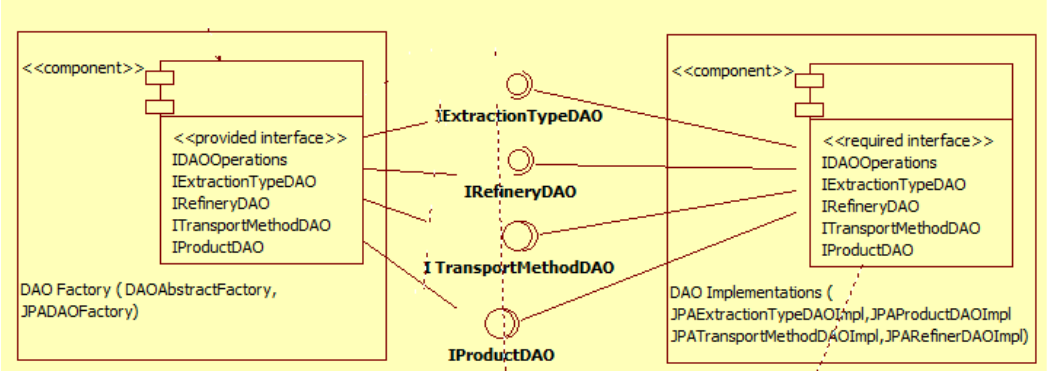

Figure 15.13 : Component Interfaces

15.5.4 Deployment diagram

We depict the details of solution components deployed on the infrastructure in the deployment diagram. Deployment diagram provide details of the hardware components (such as servers, load balancers, firewalls, virtual machines, routers, storage) and the software artefacts (such as files, Java Archives, libraries, executable etc.) that are deployed on them. Nodes (infrastructure elements) and their dependencies/relationships are modeled in the deployment diagram.
The systems support team and infrastructure team can understand the details of hardware and other physical components needed for the solution deployment. We could also understand the overall system topology, software used, the distribution model, machine configurations and such details. The infrastructure team can use this information to appropriately size the system and do the proper capacity planning. We have depicted the deployment diagram in Figure 15.14.

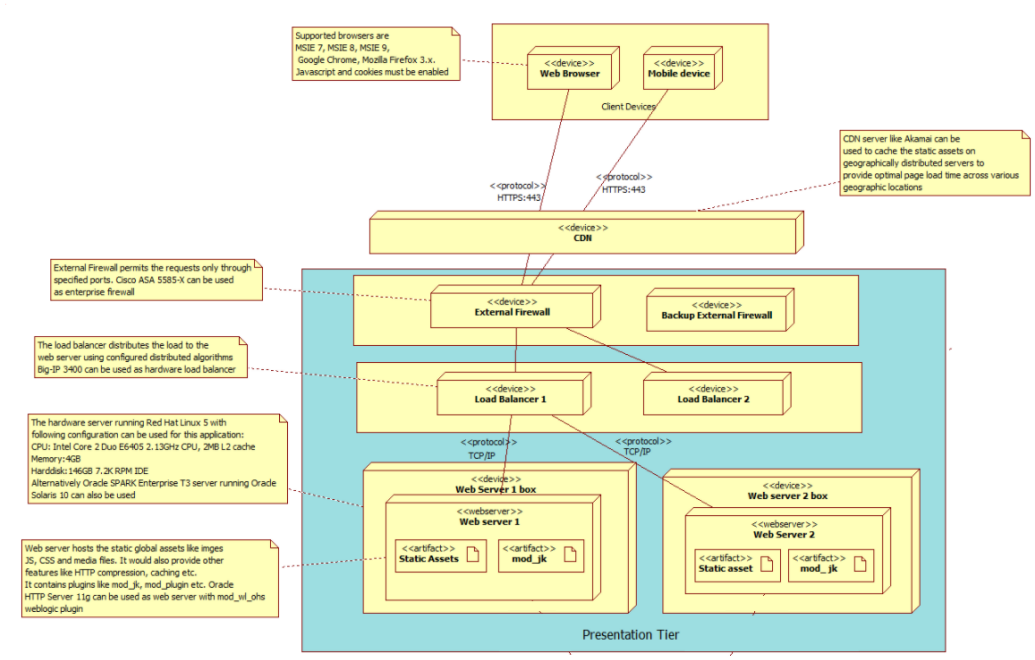

Figure 15.14 : Deployment Diagram

15.5.5 Package diagram

The package diagram depicts the packages and its constituent elements. We can organize the classes into logical groups through packages to enhance readability. The package diagram also depicts the relationship between packages. Package diagrams help us to logically group and organize the classes. By decomposing the system into subsystems and subsystems into packages we can organize the overall system better. We have depicted the package diagram in Figure 15.15.

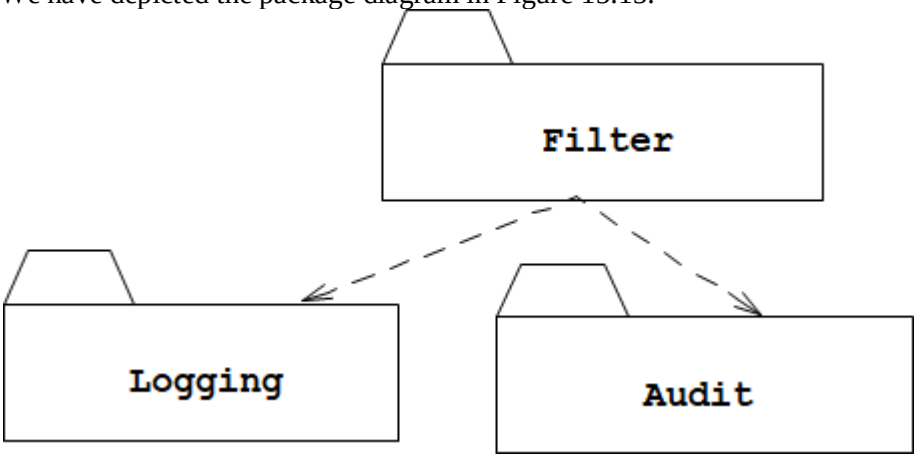

Figure 6.15 : Package Diagram

☞ **Check Your Progress 1**

1. _____ hides the inner details of the entity.
2. Encapsulation is also called _____
3. Two broad categories of UML diagrams are_____.
4. Four views covered by UML diagrams are_____.
5. _____ depicts the inheritance (parent-child relationship) in the class diagram
6. _____ Diagram depicts the software artefacts and the physical components of the infrastructure.

15.6 BEHAVIORAL UML DIAGRAMS

Behavioral elements depicted in the UML diagrams are the dynamic parts of the system. Given below are the key behavioral elements depicted in the UML diagrams; normally they are "verbs" of the system.

- **Interaction**: An interaction between two elements mainly happens through message passing and is represented as arrow as shown in Figure 15.16.
-

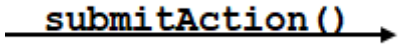

Figure 15.16 : Interaction

- **State machine**: The state machine represents various states of an object upon receiving an event. We have represented the state in Figure 15.17.

Figure 15.17 : State

15.6.1 State chart diagram

The state chart depicts the event-driven state change behavior through various states of a class and how the class can transition from one state to another for a specific event. The state chart diagrams depict the system behavior for external and internal events. The state chart has various element types – starting state or initial state which is the starting point of the process. When the object transitions to various States it is shown in the state chart diagram. The object in various states are connected by transition lines. We also depict the decision points in the state chart diagram. Statechart diagram helps us to understand the response of the system for the internal and external events. We have depicted the state chart diagram in Figure 15.18.

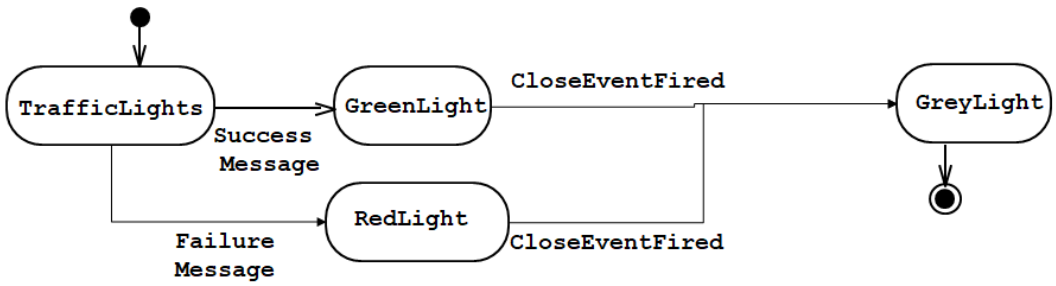

Figure 15.18 : State chart diagram

15.6.2 Activity diagram

The activity diagram depicts the control flow from one activity to another. We can model the system workflows, business rules, business process and understand the parallel and sequential activities through the activity diagram. Activity diagram also depicts the use case execution steps. Activity diagram is a type of state chart diagram in which states are depicted as activities. Activity diagrams are business-friendly. Similar to state chart diagram activity diagrams have initial stage with transition lines connecting various activities in the business process. The conditions can to be depicted in the activity diagram. The final activities that terminate the business process connected to the termination point.

We have depicted activity diagram in Figure 15.19.

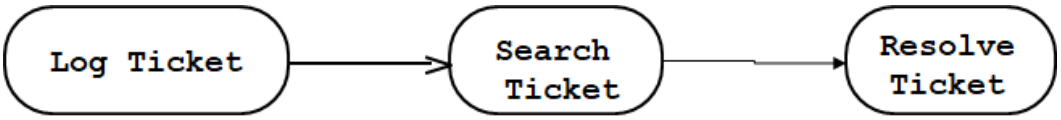

Figure 15.19 : Activity Diagram

15.6.3 Sequence diagram

The sequence diagram depicts the flow for a specific use case or a scenario. The objects perform the "call" by passing the messages. The sequence of messages are ordered by time order in which they occur. Sequence diagrams help us to understand the objects and the order in which they are invoked for the execution of a specific use case.
As depicted in Figure 15.20, arrows represent the messages, narrow vertical rectangle depict the activations and classes are in the top columns

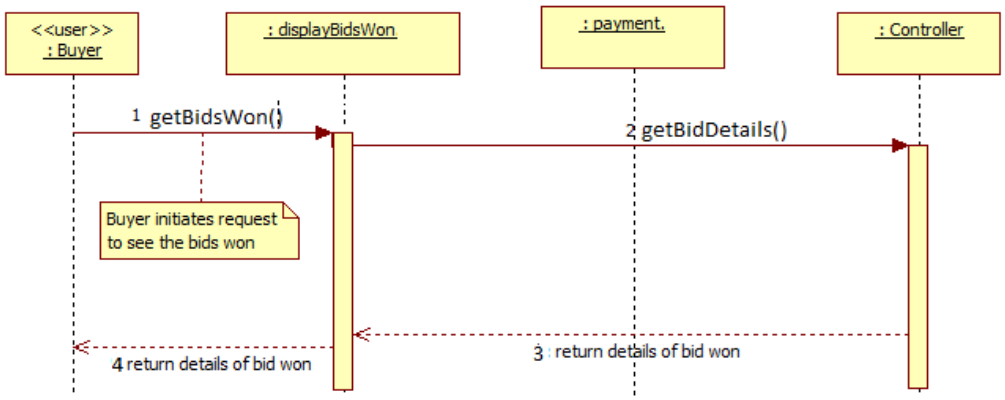

Figure 15.7 : Sequence Diagram

15.6.4 Use case diagram

Use case diagram depicts the functional behavior of the system in the visual format; use case diagram usually represent a single logical use case. The use case diagram provides visual depiction of relationship between the actors, use cases and various components for a specific functionality from the user's point of view. We can also depict the interactions of the system with external interfaces and actors.

We have depicted the use case diagram in Figure 15.21.

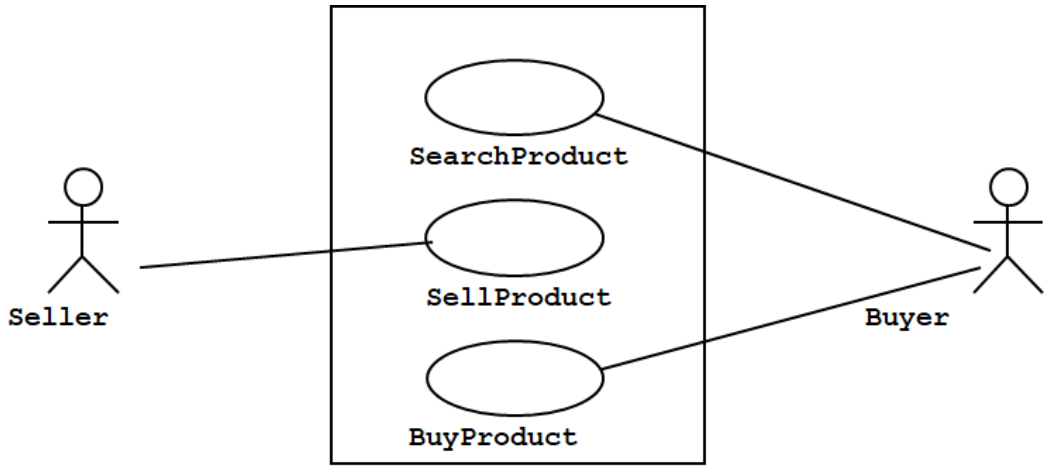

Figure 15.8 : Use case diagram

15.6.5 Collaboration diagram

A collaboration diagram consists of objects and links that represent the systems and the messages sent across the systems. A collaboration diagram is similar to a sequence diagram.

Check Your Progress 2

1. In sequence diagram _____ depicts the activations
2. _____ diagram depicts the control flow from one activity to another
3. The sequence of messages are ordered by _____
4. _____ diagram depicts relationship between the actors, use cases and various components
5. An interaction is normally represented by _____

15.7 SUMMARY

In this unit, we started discussing the key aspects of object oriented programming such as encapsulation, abstraction, inheritance and others. UML diagrams are industry standard for modeling the objects in the object oriented design. There are broadly two types of UML diagrams – structural diagrams that depict the static elements and the behavioral diagrams that depict the dynamic nature of the elements. Class diagrams depict the classes along with their relationship. Object diagrams depict the class instance along with its state. Component diagram depicts the components along with its interfaces. The deployment diagram depicts the software artefacts and the infrastructure elements. State chart diagram depicts the event-driven state transitions, Activity diagrams model the control flow and sequence diagrams provide time order sequence of objects. The use case diagram model a specific use case and the actors involved in the use case. Collaboration diagram depicts the objects and the links.

15.8 SOLUTIONS/ANSWERS

Check Your Progress 1

1. Abstraction.
2. Information hiding or data hiding.
3. process view, implementation view, deployment view and design view

4. Generalization
5. Deployment

Check Your Progress 2

1. Narrow vertical rectangle.
2. Activity
3. Time order
4. Use case
5. arrow

15.9 FURTHER READINGS

References

The Unified Modeling Language User Guide, [G. Booch, J. Rumbaugh, I. Jacobson, 2000]

UML Explained, [Kendall Scott, 2001]

Applying UML and Patterns 2nd Ed., [Craig Larman, 2002]

UML Distilled 2nd Ed., [Martin Fowler with K. Scott, 2000

Rumbaugh, James, Ivar Jacobson, Grady Booch, the Unified Modeling Language Reference Manual, Addison Wesley, 1999

Jacobson, Ivar, Grady Booch, James Rumbaugh, the Unified Software Development Process, Addison Wesley, 1999

http://en.wikipedia.org/wiki/Object-oriented_programming

UNIT 16 DATA SCIENCE FOR SOFTWARE ENGINEERS

Structure

16.0 INTRODUCTION

We are living in data-driven world where making sense out of raw data is the information super power. With each passing day, data is growing exponentially. With every tweet, Facebook post, Instagram picture, YouTube video users are generating massive amounts of data. In addition to the social media popularity, the data generated by Sensors of IoT enabled devices and wearable also generates data at high velocity. Data science is gaining popularity with the emergence of Big Data. Data analysis has been around from last many decades. The data analysis methods have evolved over the period of time. As data scientists can crunch massive amount of data to detect anomalies, patterns, trends, organizations want to leverage data science to reduce cost, explore cross-sell/upsell opportunities, new market opportunities, forecast, recommend for gaining competitive advantage. Data science is an interdisciplinary field consisting of statistics, computer science, Machine learning and others.

Data is considered as strategic asset for businesses. Organizations need to derive insights and gain value from the data they have to solve business problems and succeed in their business.

Data science helps businesses to make data driven decision making. Organizations can apply the data collection, data preparation and data analysis methods to mine massive volume of data to understand customers' behavior and explore the business opportunities to influence their customers.

16.1 OBJECTIVES

After going through this unit, you should be able to

- understand key concepts, background and definition of Data Science
- understand the relationship between Data science and Big Data
- understand various lifecycle stages of the Data science solution engineering
- understand the popular methods of data analysis
- deep dive into a case study for predicting the probability of product purchase

16.2 APPLICATIONS OF DATA SCIENCE

Given below are the main applications of data science:

- **Smart Recommendations:** Data science applications analyze massive historical data and provide effective recommendations.
- **Big Data Analytics:** Data science applications analyze the massive data such as sensor data, IoT data, social media data, log data and such to gain insights.
- **Web Search:** Analyze the web-scale data to provide accurate search results.
- **Data Analytics:** Through analysis and visualization of data we can gain insights that can help us to make data-driven predictions, forecasting and prescription.
- **Business Intelligence:** Data science applications are mainly used for business intelligence purposes. Organizations can leverage business analytics applications using data science to identify the challenges, identify cross-sell/upsell opportunities, evaluate sales offers, explore marketing opportunities, product pricing, cost optimization avenues, ROI (return on investment) analysis, revenue loss prediction, store planning, customer analytics, seasonality analysis and others.
- **Healthcare:** Data generated by wearable is used for active monitoring and to reduce the health complications.
- **Finance:** Data science methods can be used for many finance applications such as fraud detection, risk detection, data-driven insights, pattern recognition, predictive analytics and such.
- **Automobile:** Training the driverless vehicles using the training data and evaluating the performance of the models.
- **Supply Chain and Inventory Management:** Leverage data science for forecasting, demand planning and revenue forecasting.
- **Telecom:** Use Data science methods to understand the customer churn forecasting.

There are numerous other Data science applications such as election campaigning, chatbots, virtual assistants, automated cars, disaster prediction, product pricing, preventive maintenance, customer retention, pattern recognition, online advertisement, demand forecasting, and trend analysis, identifying campaign effectiveness, recommendations and such.

We have depicted the key business value derived from data science in Figure 16.1.

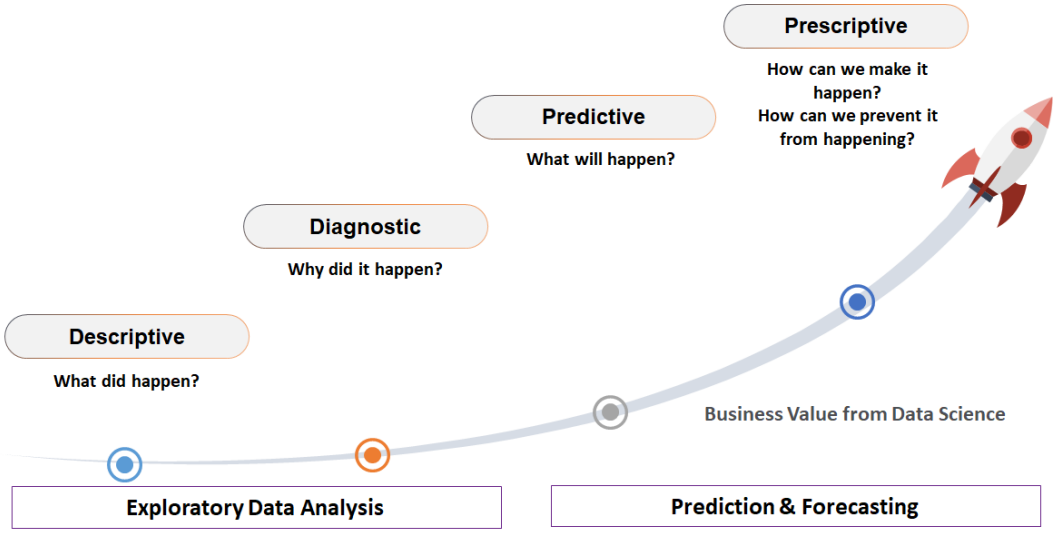

Figure 16.1 : Business Value from Data Science

As depicted in Figure 16.1, through exploratory data analysis, we can describe the events, phenomenon and scenarios. We can also look at the data dimensions and do the diagnosis to understand why the event happened. After data preparation, data analysis, model development we can predict what will happen in future and prescribe what can be done to make it happen or what can be done to prevent it from happening.

Given below are the popular applications of data science:

- Computing the most relevant online advertisement based on the user's interests, search history and showing the advertisement in real time.
- Finding the relevant product recommendations based on user's transaction history, browsing history, similar users' interest
- Flagging an email as spam through continuous learning of the features
- Identifying the obstructions by autonomous vehicles in real time.
- Understand the customer behavior in real time.
- Predicting customer churn probability using the signals.

16.3 BACKGROUND OF DATA SCIENCE

Data science is closely related to the field of statistics and mainly uses statistical methods for data analysis. As such data analysis and the analysis methods have been since 1960s. The term "Data science" can be traced back to 1974 when it was coined by Peter Naur and was later popularized by C.F.Jeff Wu in 1985.

Let us understand how the data analysis has evolved by looking at the recommendation use case. In the early 2010 we used the formal and rigid business rules for product recommendation; It was followed by linear regression model in 2011. In 2013 we achieved greater accuracy using logistical regression. Decision trees were widely used in 2014. Organizations like Amazon and Netflix leveraged collaborative filtering in 2015 for product recommendations. Bayesian network was used for recommendation in 2017. From 2019 onwards organizations are heavily using the machine learning and deep learning methods for product recommendation to achieve better accuracy.

16.4 DEFINITION OF DATA SCIENCE

Data science is an interdisciplinary field that deals with collecting, analyzing, visualizing data using statistical and machine learning methods to convert raw data into data products. Simply put, a data scientist is tasked with making sense and actionable insights out of raw data.
We have depicted various disciplines of data science in Figure 16. 2.

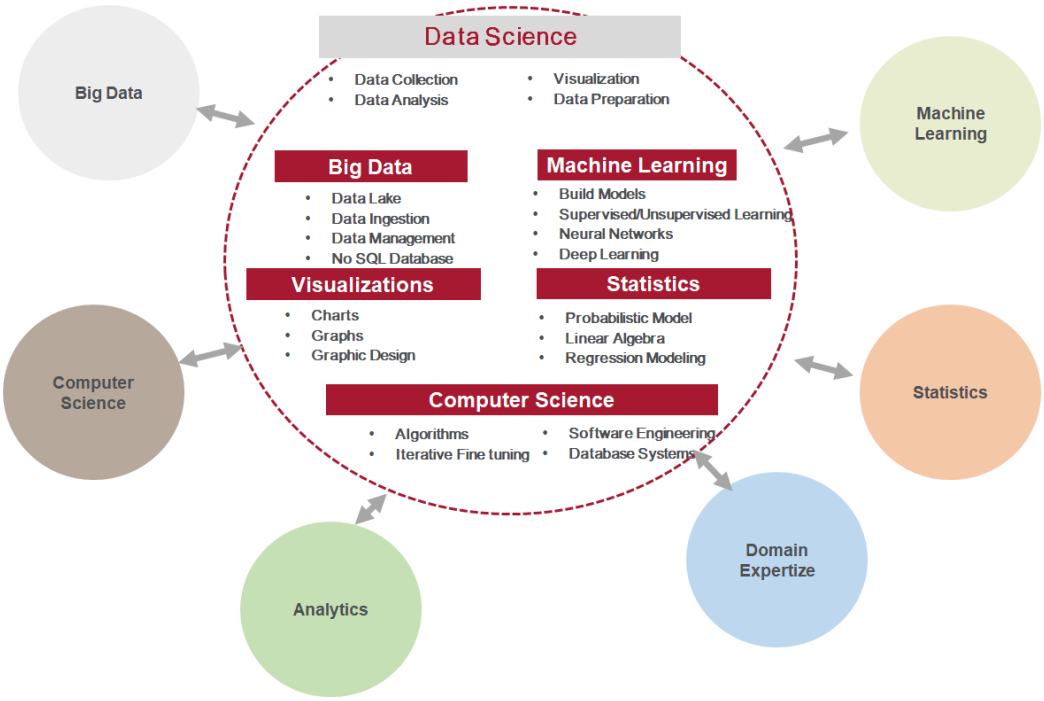

Figure 16.1 : Data Science Disciplines

Data science leverages various fields of knowledge such as machine learning, statistics, analytics, computer science and Big Data. Domain expertize is an essential skill needed for data science solutions.

16.5 DATA SCIENCE TOOLS

As data science is an interdisciplinary field, a data scientist needs to be familiar with multiple tools and technologies as given below:

- AI and Machine Learning: Deep learning methods, regression analysis, ML Models, Support vector machine (SVM), supervised and unsupervised learning, TensorFlow, Pytorch, Supervised learning includes methods such as classification, regression, logistical regression, and recommendation and decision trees. Unsupervised learning includes methods such as clustering, anomaly detection, visualization, association rule engine.
- Big Data: Apache Spark, Apache Hadoop, NoSQL databases (DynamoDB, MongoDB etc.)
- Statistics: Probability, Vector Algebra, Calculus, Linear Regression, Logistical regression,
- Data visualization: Qlik, PowerBI, Tableau
- Domain skills: Thorough understanding of the problem domain (finance, retail, commerce etc.)
- Data science platforms: Matlab, IBM Watson Studio, Anaconda, Numpy, Jupyter,

- Operations Research: Decision Tree, decision modeling (deterministic, probabilistic)
- Programming Languages: R, Python, Scala, Julia, Java, C
- Model Development: Matlab, Octave, MADLib

16.5 DATA SCIENCE AND BIG DATA

The proliferation of Big Data has propelled the increase in popularity of Data Science. The sources that generate Big Data increase in volume, velocity (massive rate of increase), veracity and variety (structured, semi-structure, and unstructured) posing challenges in organizing and analyzing the Big Data. Examples of Big Data are sensor data, user click data, tweet data and such. We cannot easily manage Big Data through traditional databases. Hence Big Data systems don't strictly adhere to ACID (Atomicity, Consistency, Integrity and Durability) properties but aim for eventual consistency as per CAP (Consistency, Availability, Partition) theorem. Some of the popular Big Data systems are NoSQL products such as MongoDB, DynamoDB, CouchDB and such.
As part of Data science development, we build, explore and fine tune various models to handle the massive data sets.

Big Data provide the massive data management, data processing technologies such as Apache Hadoop, and NoSQL databases for data science methods.

16.6 PHASES INVOLVED IN DATA SCIENCE PROCESS

We have depicted various phases involved in the data science process in the Figure 16.3.

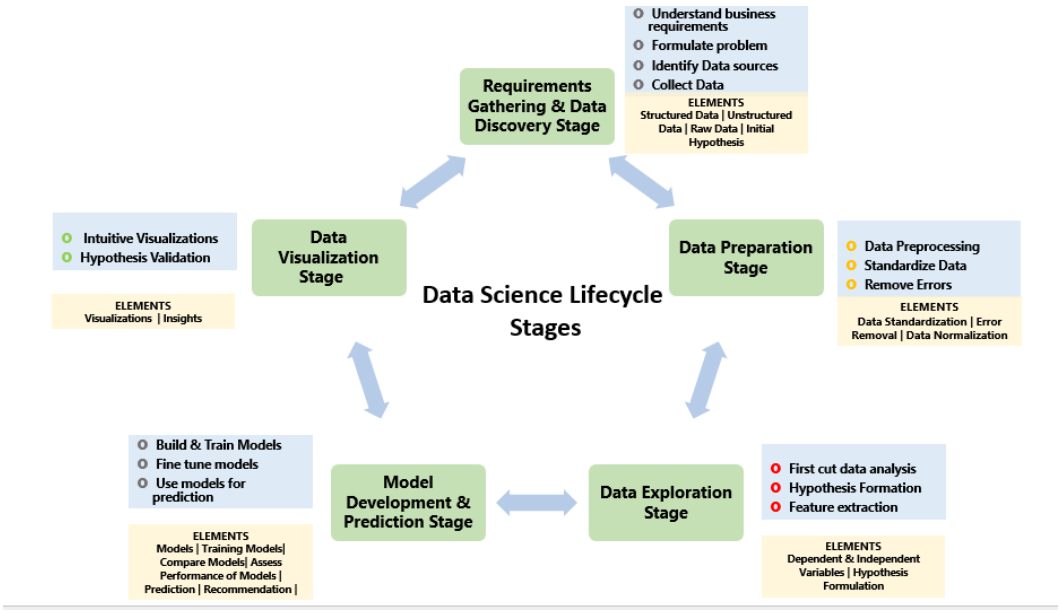

Figure 16. 2 : Lifecycle Stages of Data Processing

In data science the data is processed through various stages in the pipeline. We have depicted the stages in Figure 16.3. During each of the stages in the pipeline, the raw data undergoes series of analysis and changes. The lifecycle stages are cyclic as we continuously use the feedback from each stage to its previous stage.

We shall look at each of the stages in next sections.

16.6.1 Requirements Gathering and Data Discovery

We need to understand the business requirements, the goals, focus areas and challenges. We need to understand the key data sources, data inventories, existing data analysis tools and technologies, the subject matter experts (SMEs), project sponsor, main stakeholders, time and budget for the program. It is essential to also understand any existing data analysis initiatives undertaken to learn from the past experience.

Domain experts, business analysts for the business domain help us to provide the heuristics related to the business domain using which we can formulate the initial hypothesis. Once we thoroughly understand the business domain we can then formulate the problem for applying the data science methods.

Once we fully understand the requirements, we need to collect data from various sources such as data lakes, relational databases, ERP systems, internal and external social media systems, collaboration systems and such. For instance for the customer behavioral analysis, we need to gather data from various sources such as CRM system (user case details), user profile details, order history, product review system (product name, review details), email system, social media (user's social media handle, user details) and such.

16.6.2 Data Preparation stage

To improve the quality of the extracted data, we need to cleanse the data to standardize the format, structure and remove any inconsistencies (duplicates, inconsistencies, missing values, errors and such). Data preparation is a time consuming stage wherein a data scientist separates signal from noise. The data conditioning can be performed by ETL (Extract, Transform and Load) tools. As part of data conditioning, we do the data normalization, conversion to same case and such.

Given below are some of the main activities as part of the data conditioning stage:

- Removing duplicates
- Bring the data to a common or standard format
- Infering the missing values
- Smoothing the data by removing the noise
- Filtering and sampling the data
- Making the data types consistent across the entire data set
- Replace missing values with their mean or median values

The conditioned data will be used for exploration and model development.

16.6.3 Data exploration stage

During this stage, we extract the features from the data and process the data. Based on the extracted features we can perform first cut analysis and form the initial hypothesis. We also classify, cluster the data into the logical categories. We identify the relationship between variables during this stage. We then select the key predictors that are essential for the recommendation and for prediction.

For instance, in order to model the price of a used car, we need to examine massive amount of historical used car sales transactions consisting of various attributes. However key features such as car mileage, car age, kms driven, engine capacity, purchase price have direct correlation with the car's market price. These key features directly influence the final price for the used car. Data scientists look at the data and try to correlate the useful features and form the hypothesis. In the used car price

prediction use case, the hypothesis could be “The car age is inversely correlated to the final price (with the increase in car age, the final price decreases)”.

16.6.4 Model development and Prediction stage

We design and develop the machine learning models in this stage. We train and fine tune the model and test their performance on an iterative basis. Data scientists evaluate various models for the selected data and finally the model that provides the highest performance will be selected for prediction.

We use many statistical methods such as regression analysis, qualitative methods, decision tree models, deep learning models for data analysis during this stage. We evaluate the performance of the model using various data points such as the accuracy of the model prediction, model performance on various training and test data sets and such.

We then use the selected models for the prediction.

16.6.5 Data Visualization stage

In the final stage, we visually communicate the obtained insights using reports, business intelligence (BI) tools to help the audience make informed decisions. We also document the key findings, observations and the behavior of the model for various test cases. Often we use the visual representations such as charts, infographics to communicate the findings to the stakeholders.

Once the model is accepted by the stakeholders, we operationalize the model and use it for business prediction. We need to continuously fine tune the model based on the feedback and learning from the real world data.

A sample data visualization is shown in Figure 16.4.

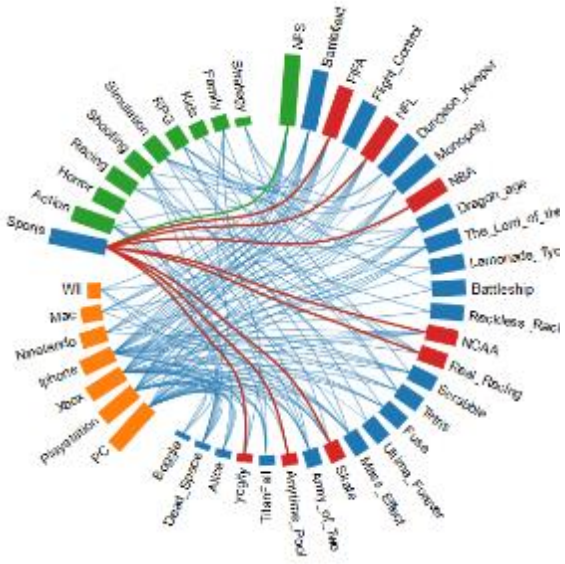

Figure 16.4 : Sample Data Visualization

16.6.6 A Sample Case Study

In this example we shall look at a sample case study to understand the various stages of data science pipeline events.
The use case we have considered below is for predicting the most popular (and moving) product at a store. Based on the product’s popularity, the organization can chose to replenish the stock of the product faster.

Requirement gathering and Data collection stage

In this phase, we collect all the data needed to predict the popular product at a given store. We collect the below given data at a product level for each store.

Table 16.1 : Product sales data in raw format

Top 5 in last month (Yes/No)	Top 5 in last year (Yes/No)	Brand campaign (Yes or No)	Buyer Age Group 10-18 yrs. (Yes/No)	Buyer Age Group 19-34 yrs. (Yes/No)	Buyer Age Group 30-50 yrs. (Yes/No)	Buyer Income (10K – 30K) Yes/ No	Product	Store	Sales
1	0	1	0	1	Nil	1	ABC	Store1	500
1	1	Yes	0	0	1	1	XYZ	Store1	1500
			0			1	ABC	Store1	500

In table 16.1, "1" denotes "yes" and "0" denotes "No"

Likewise we collect all the possible data from all stores for all combinations of identified features

Data Cleansing Stage

During data cleansing stage, we identify the following:

- Errors: If there are any errors in the data entry we will fix it
- Missing data: If the data for any features are missing, we can either take the median value to fill the data or drop that feature if it is missing in both training and test data set.

In the above example one of the values is "Yes" that needs to be coded to "1". Another value is "Nil" that will be converted to "0". Third row of data is empty for many features hence we will discard that row for the training purpose. The conditioned data is depicted in table 16.2.

Table 16.1 : Conditioned Product sales data

Top 5 in last month (Yes/No)	Top 5 in last year (Yes/No)	Brand campaign (Yes or No)	Buyer Age Group 10-18 yrs. (Yes/No)	Buyer Age Group 19-34 yrs. (Yes/No)	Buyer Age Group 30-50 yrs. (Yes/No)	Buyer Income (10K – 30K) Yes/ No	Product	Store	Sales
1	0	1	0	1	0	1	ABC	Store1	500
1	1	1	0	0	1	1	XYZ	Store1	1500

Data Exploration Stage

During this stage, we identify the prominent features that influence the overall product's popularity. In the above example, we identify that demographic data has greatest influence on the overall product sales.

We understand the features that have positive/linear correlation with the product sales, non-linear correlation with the product sales.

Prediction Stage

In this stage, we predict the product sales for each store. For this, we develop the models that use the dependent features to predict the overall product sales for each store. We can explore various models given below against the test data and compare the accuracy and performance.

- Gaussian Naive Bayes
- Logistic Regression
- Support Vector Machines
- Perceptron
- Decision Tree Classifier
- Random Forest Classifier
- KNN or k-Nearest Neighbors
- Stochastic Gradient Descent
- Gradient Boosting Classifier

Once we finalize the model we can train it further and fine tune to achieve better accuracy. We can then use the model to predict the top selling products.

Data visualization stage

In the data visualization stage, we provide the top predicted products for each store so that the store owner can replenish the inventory for the popular products as depicted in the Figure 16.5.

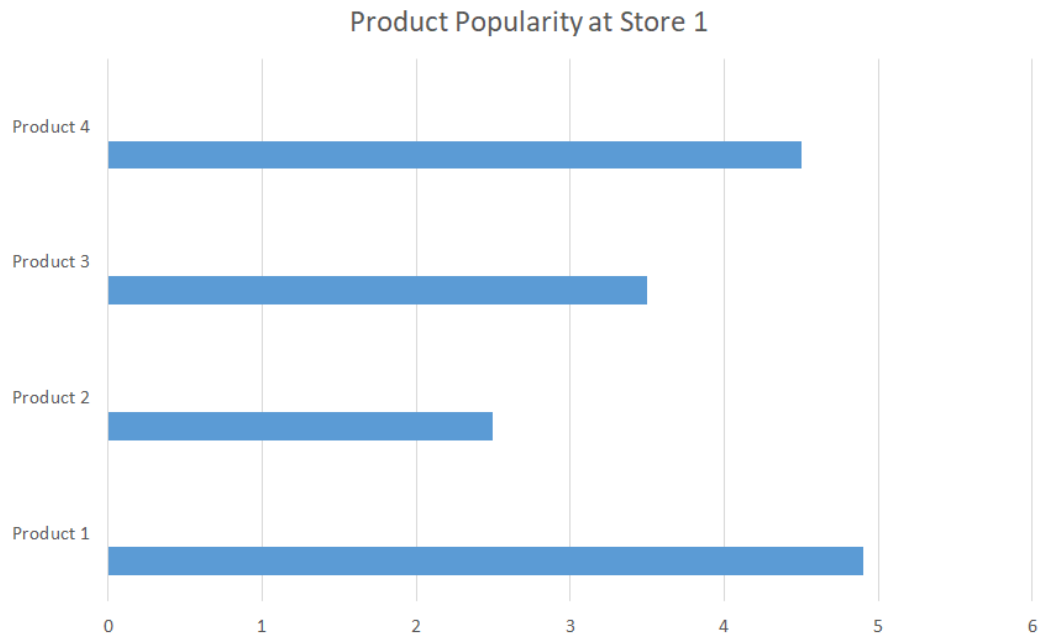

Figure 16.3 : Top producs at store 1

16.7 DATA SCIENCE METHODS

In this section we shall look at the most popular data analysis methods we use in the data science projects.

16.7.1 Clustering Method

As part of clustering method, we group the objects into a logical cluster/group based on the object attribute. As part of unsupervised learning, the model groups the data based on the labels. For instance, as part of demographic data, we can define groups as follows:

- Users less than 20 years
- Users between 20-40 years
- Users more than 40 years

During exploratory analysis, we can form clusters and create rules and predictions based on the clusters. The main clustering methods are hierarchical clustering, fuzzy clustering, K-means clustering, nearest neighboring

Use cases
In Data science we heavily use the clustering methods for group-based personalization, customer segmentation, image processing

16.7.2 Collaborative Filtering or Similarity Matching Method

In collaborative filtering the system identifies the users with similar interests (based on likes or purchases of similar products). The information about the set the users with similar interests will then be used for recommending the new products for the users. Collaboration filtering uses set of rules to understand the similarity between users, products and information sources.

For examples if user A has same interest as that of user B while purchasing a book, then we can recommend user A's books to user B which user B has not yet purchased.

A variation of collaboration filtering is to understand the association of products (which products are often purchased together) known as co-occurrence grouping. Based on this insight, we can recommend the closely related products together to the customer.

Use cases
Collaboration filtering is used in recommendation systems for cross-sell and up-sell opportunities such as the following:

"Customers who bought this item also bought these items"

"Based on your purchase history, we recommend these products"

"You may also like these products"

Collaborative filtering is also used for campaigns, product promotions and product offers.

☞ **Check Your Progress 1**

1. In ____ phase we do the data pre-processing.
2. The key features of the data set are extracted in _____ phase
3. we group the objects into a logical cluster/group based on the object attribute in _____ method
4. _____ uses set of rules to understand the similarity between users, products and information
5. Understanding the association of products happens in _____

16.7.3 Regression Methods

Regression methods such as linear regression and logistical regression methods identify the relationship (as a function) between dependent variables and independent variable. Based on the learnt relationship the regression methods can predict the independent variable using the key predictor dependent variables.

For instance, we can get the historical prices of house sales from past 5 years. Based on the historical data we can use logistical regression between the independent variable (house price) with the dependent variables (such as number of beds, location, carpet area, amenities, nearby point of interests) to predict the likely price for houses.

Use cases
Regression methods are mainly used for prediction and forecasting scenarios such as demand forecasting, price prediction. Logistical regression is also used for predicting the probability of an event (such as customer churn event or loan default event) given the key dependent variables.

16.7.4 Classification Methods

As part of supervised learning, classifiers are given labelled data to understand the data. Once the classifier is fully trained, it can be used to label the new data.

The main classification methods are decision tree method, naïve Bayes classification and Confusion matrix. We will be using decision tree classification for a product purchase use case using the attributes in the next section.

In the decision tree method we structure the decisions in a tree format. The test points (known as nodes) with the Boolean values (yes or no) acting as branches. Once we fully traverse a path in the tree, we can reach the predicted value.

Use cases
Classification methods are mainly used for prediction and recommendation scenarios.

16.7.5 Other Methods

We have given other common methods for Data Science based solutions:

- Time series prediction methods to analyze the events occurring at regular time intervals
- Text analysis and sentiment analysis using methods such as bag of words, TFIDF, topic modelling to gain insights from the text.
- Profiling involves characterizing the event or user or an object.
- Reduction involves reducing the large data set into a smaller data set that represents the key samples from the large data set.

16.8 DATA SCIENCE PROCESS FOR PREDICTING THE PROBABILITY OF PRODUCT PURCHASE

In this sample case study let us look at various steps involved in the predicting the probability of product purchase and understanding of the key features.

Requirement gathering and Data collection stage

The sample case study is aimed at understanding the key features that influence the probability of product purchase and develop a prediction model given the key feature values.

We have provided a sample data collected for eleven historical transactions in table 16.3 for reference.

Table 16.3 : Product purchase data

Age_LT30	Age_GT30	In_LT50	In_GT50	Loyal_CT	onSale	Prod_buy
0	1	0	1	1	1	1
0	1	0	1	0	1	1
1	0	0	1	1	1	1
0	1	0	1	1	1	1
1	0	Yes	0	1	1	0
0	1	Correct	0	1	1	1
1	0	Y	0	0	1	0
1	0	Less than 50	0	0	0	0
0	1	Confirmed	0	0	1	1
1	0	One	0	1	1	0
1	0	NA	0	1	1	0

We have collected the key features that influence a product purchase in an e-commerce portal:

- Age_LT30 stands for age less than 30, indicates if the buyer is aged below 30 years. 1 indicates Yes and 0 indicates no
- Age_GT30 stands for age greater than 30, indicates if the buyer is aged below 30 years. 1 indicates Yes and 0 indicates no
- In_LT50 stands for income less than 50, indicates if the buyer's monthly income is less than 50K. 1 indicates Yes and 0 indicates no
- In_GT50 stands for greater less than 50, indicates if the buyer's monthly income is greater than 50K. 1 indicates Yes and 0 indicates no
- Loyal_CT stands for loyal customer who has enrolled into the loyalty program. 1 indicates Yes and 0 indicates no
- onSale stands for product on sale. 1 indicates Yes and 0 indicates no
- Prod_buy stands for product buy. 1 indicates Yes and 0 indicates no

We have collected eleven historical values for the illustration of the case study.

Data Cleansing Stage

In this phase, we fix the data errors and condition the data so that it can be easily modeled using one of the existing models. The highlighted rows in the table need to be properly coded. For accurate model training we need to use the correct code. Table 16.4 provides the correctly coded values.

Table 16.2 : Product purchase data coded

Age_LT30	Age_GT30	In_LT50	In_GT50	Loyal_CT	onSale	Prod_buy
0	1	0	1	1	1	1
0	1	0	1	0	1	1
1	0	0	1	1	1	1
0	1	0	1	1	1	1
1	0	1	0	1	1	0
0	1	1	0	1	1	1
1	0	1	0	0	1	0
1	0	1	0	0	0	0

0	1	1	0	0	1	1
1	0	1	0	1	1	0
1	0	1	0	1	1	0

Data Exploration Stage

In this stage, we identify the key features and its relation to the product purchase decision. We apply the statistical methods to understand the data distribution and the feature relationship. We can infer the following based on our analysis

- Feature Age_GT30 is positively and linearly correlated to prod_buy. Conversely Age_LT30 is inversely related to prod_buy.
- Feature IN_GT50 is positively and linearly correlated to prod_buy. Conversely In_LT50 is inversely related to prod_buy.
- Loyal_CT and onSale are positively correlated to prod_buy

We can use these features and insights for building and training models.

Prediction Stage

In this stage we build and evaluate various models. We can evaluate various models such as linear regression, logistic regression and others and understand their performance. We have selected the decision tree model as it factors all the key features. We can visualize the combination and impact of various features with the decision tree model.

We validate the decision tree model with the test data to ensure that the decision tree model does the accurate prediction.

Data visualization stage

Figure 16.6 provides sample decision tree for our product purchase prediction.

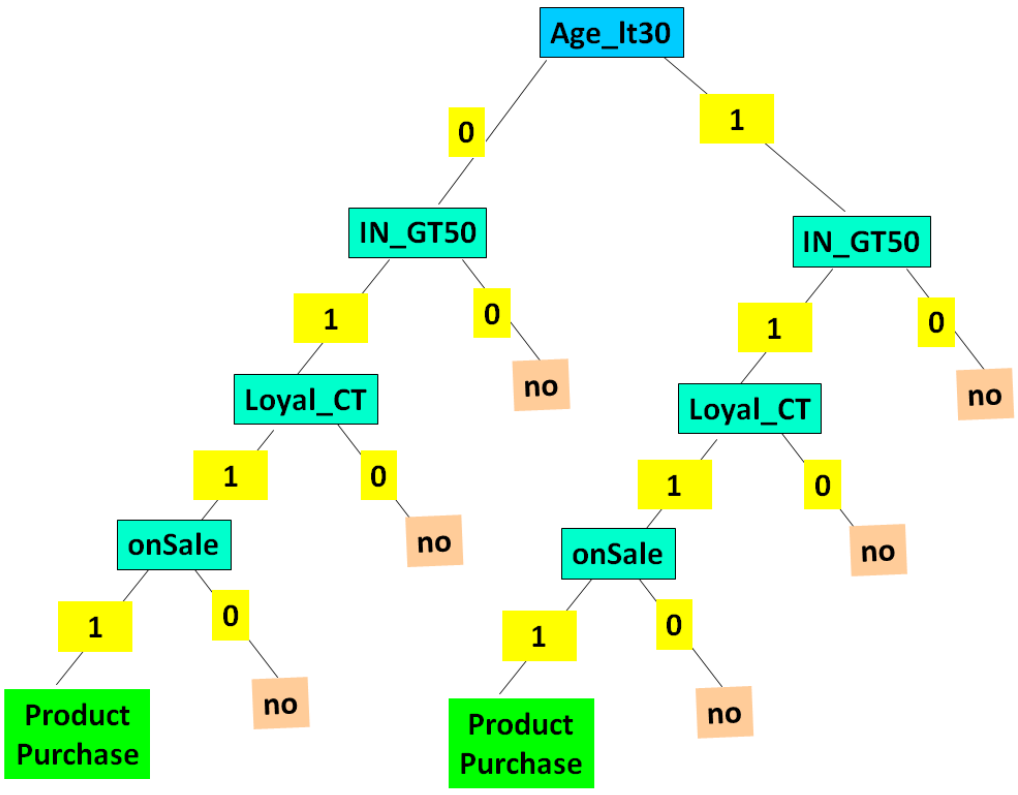

Figure 16.4 : Decision Tree for Product Purchase

☞ **Check Your Progress 2**

1. _____methods identify the relationship (as a function) between dependent variables and independent variable.
2. In _____ method, labelled data is used for understanding of the data.
3. _____ involves characterizing the event or user or an object
4. _____ involves reducing the large data set into a smaller data set

16.9 SUMMARY

In this unit, we started discussing main applications of the data science. We discussed the interdisciplinary nature of the data science as it involves methods from various fields such as computer science, big data, statistics, analytics, domain knowledge, machine learning and others. We also looked the key tools of data science such as R, Python, Matlab others. We discussed the relationship between data science and Big data. In next section we had detailed discussion on various phases of the data science lifecycle stages. The requirements gathering and data collection stage involves gathering all the data sources, data preparation stage conditions the data and fixes the errors. Data exploration stage involves close examination of the key features and building the hypothesis. In Prediction stage we build models to predict the event. Finally in the data visualization stage we document and showcase our findings. We also had a deep dive discussion on various data science methods related to clustering, collaborative filtering, regression and classification. Finally we look at a detailed case study for using data science method for predicting a product purchase probability.

16.10 SOLUTIONS/ANSWERS

Check Your Progress 1

1. preparation.
2. data exploration
3. clustering
4. collaborative filtering
5. co-occurrence grouping

Check Your Progress 2

1. Regression.
2. Classification
3. Profiling
4. Reduction

16.11 FURTHER READINGS

References

Provost, F., & Fawcett, T. (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking.* " O'Reilly Media, Inc.".

Van Der Aalst, W. (2016). Data science in action. In *Process mining* (pp. 3-23). Springer, Berlin, Heidelberg.

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... & Zimmermann, T. (2019, May). Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (pp. 291-300). IEEE.

Agarwal, R., & Dhar, V. (2014). Big data, data science, and analytics: The opportunity and challenge for IS research.