ARUN COMPUTER & IT

PDF Study Material

For PGDCA & DCA Examination

IT Trends

Unit-5

विषय सूची:

- Introduction to MIS
- Phases,of system development
- Data Analysis, decision tree and form design

हमारी वेबसाइट से डाउनलोड करें

Latest version available only at: www.hamararewa.in

Prepared & Updated by Arun Computer © 2026 All Rights

एमआईएस (MIS) –MIS (Management Information System)

MIS (प्रबंधन सूचना प्रणाली) एक कंप्यूटर-आधारित प्रणाली है, जो प्रबंधकों को निर्णय लेने के लिए आवश्यक जानकारी एकत्रित, प्रोसेस, स्टोर और वितरित करती है। यह संगठन के विभिन्न स्तरों (निचले, मध्यम, उच्च) पर नियोजन, नियंत्रण और निर्णयन में सहायता करती है। MIS में हार्डवेयर, सॉफ्टवेयर, डेटा, प्रक्रियाएँ और मानव संसाधन शामिल होते हैं। उदाहरण: सेल्स MIS (बिक्री डेटा), इन्वेंटरी MIS (स्टॉक डेटा), फाइनेंशियल MIS (लाभ-हानि रिपोर्ट)। MIS का मुख्य उद्देश्य सही समय पर सही प्रारूप में सही व्यक्ति तक सही जानकारी पहुँचाना है।

एमआईएस के प्रमुख घटक और कार्य

MIS के चार मुख्य घटक होते हैं:
(1) डेटा संग्रहण (Data Collection) – आंतरिक और बाह्य स्रोतों से कच्चा डेटा इकट्ठा करना।
(2) डेटा प्रोसेसिंग (Processing) – डेटा को छाँटना, गणना करना, सारांश बनाना, और विश्लेषण करना।
(3) डेटा भंडारण (Storage) – प्रोसेस्ड डेटा को डेटाबेस में सुरक्षित रखना।
(4) सूचना वितरण (Distribution) – रिपोर्ट, डैशबोर्ड, ग्राफ के माध्यम से प्रबंधकों तक जानकारी पहुँचाना। MIS के कार्यों में शामिल हैं: नियमित रिपोर्ट तैयार करना, अपवाद रिपोर्ट (exception reporting) देना, पूर्वानुमान (forecasting) लगाना, और निर्णय समर्थन (decision support) प्रदान करना। उदाहरण: मासिक बिक्री रिपोर्ट, इन्वेंटरी री-ऑर्डर अलर्ट।

सिस्टम डेवलपमेंट लाइफ साइकिल (SDLC) –

SDLC (सिस्टम डेवलपमेंट लाइफ साइकिल) एक संरचित और चरणबद्ध प्रक्रिया है, जिसका उपयोग सूचना प्रणाली या सॉफ्टवेयर को विकसित करने, बनाए रखने और बदलने के लिए किया जाता है। यह एक परियोजना प्रबंधन मॉडल है जो सिस्टम विकास के विभिन्न चरणों को परिभाषित करता है, जिससे समय, लागत और गुणवत्ता पर नियंत्रण रहता है। SDLC का उद्देश्य उच्च-गुणवत्ता वाला सिस्टम बनाना है जो ग्राहक की आवश्यकताओं को पूरा करे, बजट और समय सीमा के भीतर तैयार हो। SDLC के प्रसिद्ध मॉडलों में वॉटरफॉल, एजाइल, स्पाइरल, V-Model, और रैपिड एप्लिकेशन डेवलपमेंट (RAD) शामिल हैं।

SDLC के विभिन्न चरण (Various Phases of SDLC)

SDLC में सामान्यतः निम्नलिखित सात चरण होते हैं:
(1) योजना (Planning) – परियोजना की व्यवहार्यता, लागत, जोखिम और समय-सीमा का निर्धारण।
(2) आवश्यकता विश्लेषण (Requirement Analysis) – उपयोगकर्ताओं से आवश्यकताएँ एकत्रित और दस्तावेजित करना (SRS - Software Requirement Specification)।
(3) डिजाइन (Design) – सिस्टम आर्किटेक्चर, डेटाबेस, यूजर इंटरफेस और कंपोनेंट्स का डिजाइन तैयार करना।
(4) कोडिंग / कार्यान्वयन (Coding/Implementation) – डिजाइन के अनुसार प्रोग्राम लिखना और यूनिट टेस्टिंग करना।
(5) टेस्टिंग (Testing) – बग्स और त्रुटियों को खोजने के लिए इंटीग्रेशन, सिस्टम और यूजर एक्सेप्टेंस टेस्टिंग (UAT) करना।
(6) तैनाती (Deployment) – सिस्टम को प्रोडक्शन एनवायरनमेंट में लॉन्च करना और उपयोगकर्ताओं को प्रशिक्षण देना।
(7) रखरखाव (Maintenance) – सिस्टम में बग फिक्स करना, अपडेट देना और आवश्यकतानुसार संशोधन करना।

चरण 1: योजना (Planning / Feasibility Study)

योजना चरण SDLC का पहला और महत्वपूर्ण चरण है, जिसमें यह निर्धारित किया जाता है कि प्रस्तावित सिस्टम व्यवहार्य है या नहीं। इस चरण में चार प्रकार की व्यवहार्यता की जाँच की जाती है: आर्थिक व्यवहार्यता (Economic Feasibility) – क्या लाभ लागत से अधिक है?, तकनीकी व्यवहार्यता (Technical Feasibility) – आवश्यक तकनीक और कुशलता उपलब्ध है?, संचालनात्मक व्यवहार्यता (Operational Feasibility) – क्या उपयोगकर्ता नए सिस्टम को अपनाएंगे?, और समय व्यवहार्यता (Schedule Feasibility) – क्या परियोजना निर्धारित समय में पूरी हो सकती है?। इस चरण में एक परियोजना योजना (Project Plan) बनाई जाती है, जिसमें संसाधन, बजट, जोखिम और मील के पत्थर (milestones) शामिल होते हैं। इस चरण का आउटपुट "फीजिबिलिटी रिपोर्ट" और "प्रोजेक्ट चार्टर" होता है।

चरण 2: आवश्यकता विश्लेषण (Requirement Analysis)

इस चरण में, सिस्टम एनालिस्ट उपयोगकर्ताओं, स्टेकहोल्डर्स और ग्राहकों से मिलकर उनकी आवश्यकताओं, अपेक्षाओं और समस्याओं को समझता है। आवश्यकताएँ दो प्रकार की होती हैं: कार्यात्मक आवश्यकताएँ (Functional Requirements) – सिस्टम क्या करेगा (जैसे

"लॉगिन करना", "रिपोर्ट जनरेट करना") और गैर-कार्यात्मक आवश्यकताएँ (Non-Functional Requirements) – सिस्टम कैसे करेगा (जैसे प्रदर्शन, सुरक्षा, उपलब्धता, स्केलेबिलिटी)। आवश्यकताओं को एकत्रित करने के तरीकों में इंटरव्यू, सर्वेक्षण, प्रश्नावली, अवलोकन और कार्यशालाएँ शामिल हैं। इस चरण का अंतिम आउटपुट SRS (Software Requirement Specification) दस्तावेज़ होता है, जो एक कानूनी अनुबंध की तरह काम करता है। SRS को स्पष्ट, मापने योग्य और अस्पष्टता से मुक्त होना चाहिए।

चरण 3: डिजाइन (Design)

डिजाइन चरण में, SRS दस्तावेज़ को तकनीकी रूपरेखा (blueprint) में बदला जाता है जिसे प्रोग्रामर कोडिंग के लिए उपयोग कर सकें। यह दो उप-चरणों में विभाजित है: उच्च-स्तरीय डिजाइन (HLD - High-Level Design) – सिस्टम की समग्र संरचना, मॉड्यूल, डेटाबेस आर्किटेक्चर, और कंपोनेंट्स के बीच संबंध तय करता है। विस्तृत डिजाइन (DD - Detailed Design) – प्रत्येक मॉड्यूल का आंतरिक तर्क, एल्गोरिदम, डेटा संरचनाएँ, और यूजर इंटरफेस (स्क्रीन डिजाइन) तैयार करता है। इस चरण के आउटपुट में डेटा फ्लो डायग्राम (DFD), एंटिटी-रिलेशनशिप डायग्राम (ERD), स्ट्रक्चर चार्ट, और प्रोटोटाइप स्क्रीन शामिल होते हैं। एक अच्छा डिजाइन सिस्टम को स्केलेबल, मेंटेन करने योग्य और उपयोगकर्ता के अनुकूल बनाता है।

चरण 4: कोडिंग / कार्यान्वयन (Coding / Implementation)

कोडिंग चरण में, डिजाइन दस्तावेज़ के अनुसार प्रोग्रामर उचित प्रोग्रामिंग भाषा (जैसे Java, Python, C#, PHP) का उपयोग करके स्रोत कोड लिखते हैं। यह चरण सबसे लंबा होता है और इसमें कोडिंग मानकों (coding standards) का पालन किया जाता है, जैसे इंडेंटेशन, कमेंटिंग, नेमिंग कन्वेंशन्स। कोड लिखने के बाद प्रोग्रामर यूनिट टेस्टिंग (Unit Testing) करता है, जिसमें प्रत्येक मॉड्यूल या फंक्शन को अलग से चेक किया जाता है कि वह अपेक्षित आउटपुट दे रहा है या नहीं। इस चरण में सोर्स कोड मैनेजमेंट (Git, SVN), डिबगिंग टूल्स, और डेवलपमेंट एनवायरनमेंट (IDE) का उपयोग किया जाता है। इस चरण का आउटपुट "कार्यशील कोड" और "यूनिट टेस्ट रिपोर्ट" होता है। अच्छे कोडिंग प्रथाओं से बग्स कम होते हैं और रखरखाव आसान होता है।

चरण 5: टेस्टिंग (Testing)

टेस्टिंग चरण का उद्देश्य सॉफ्टवेयर में त्रुटियों (bugs), गैप्स, या आवश्यकताओं से विचलन को खोजना और ठीक करना है। इस चरण में विभिन्न स्तरों पर परीक्षण किए जाते हैं:
(1) इंटीग्रेशन टेस्टिंग – विभिन्न मॉड्यूल के बीच इंटरफेस की जाँच।
(2) सिस्टम टेस्टिंग – पूरे सिस्टम का समग्र परीक्षण।

(3) यूजर एक्सेप्टेंस टेस्टिंग (UAT) – अंतिम उपयोगकर्ताओं द्वारा यह सुनिश्चित करना कि सिस्टम उनकी आवश्यकताओं को पूरा करता है। इसके अलावा
प्रदर्शन टेस्टिंग (Performance Testing) – लोड और स्ट्रेस के तहत सिस्टम की प्रतिक्रिया,
सुरक्षा टेस्टिंग (Security Testing) – कमजोरियों की पहचान, और
रीग्रेशन टेस्टिंग (Regression Testing) – नए बदलावों से पुराने फीचर्स प्रभावित तो नहीं हुए,
की जाती है। टेस्टिंग मैन्युअल (manual) या स्वचालित (automated) उपकरणों (जैसे Selenium, JUnit) से की जा सकती है। इस चरण का आउटपुट "टेस्ट रिपोर्ट" और "बग रिपोर्ट" होता है।

चरण 6: तैनाती (Deployment / Delivery)

तैनाती चरण में, सफलतापूर्वक परीक्षण किए गए सॉफ्टवेयर को प्रोडक्शन एनवायरनमेंट (वास्तविक उपयोगकर्ताओं के लिए) में लॉन्च किया जाता है। तैनाती के विभिन्न तरीके हैं: डायरेक्ट कटओवर (Direct Cutover) – पुराने सिस्टम को अचानक बंद कर नया शुरू करना, फेज्ड तैनाती (Phased Deployment) – एक-एक मॉड्यूल धीरे-धीरे लॉन्च करना, पायलट तैनाती (Pilot Deployment) – पहले कुछ चुनिंदा उपयोगकर्ताओं के लिए लॉन्च करना, और समानांतर तैनाती (Parallel Deployment) – कुछ समय तक पुराने और नए सिस्टम को साथ चलाना। इस चरण में उपयोगकर्ताओं को प्रशिक्षण (Training) दिया जाता है और प्रलेखन (Documentation) – उपयोगकर्ता मैनुअल, सिस्टम एडमिन गाइड – सौंपा जाता है। तैनाती के बाद एक "गो-लाइव" घोषणा की जाती है और सिस्टम को मॉनिटर किया जाता है। इस चरण का आउटपुट "लाइव सिस्टम" और "प्रशिक्षित उपयोगकर्ता" होता है।

चरण 7: रखरखाव (Maintenance)

रखरखाव SDLC का अंतिम चरण है, लेकिन यह सबसे लंबे समय तक चलने वाला होता है (सिस्टम के जीवन का 60-70% समय)। इस चरण में सिस्टम के लाइव होने के बाद आने वाले बदलावों, सुधारों और समस्याओं का समाधान किया जाता है। रखरखाव चार प्रकार का होता है:
(1) सुधारात्मक रखरखाव (Corrective Maintenance) – लाइव सिस्टम में मिले बग्स और त्रुटियों को ठीक करना।
(2) अनुकूली रखरखाव (Adaptive Maintenance) – बदलते वातावरण (जैसे OS अपडेट, नया हार्डवेयर) के अनुसार सिस्टम को बदलना।
(3) पूर्णता रखरखाव (Perfective Maintenance) – नई सुविधाएँ जोड़ना और प्रदर्शन सुधारना।
(4) निवारक रखरखाव (Preventive Maintenance) – भविष्य की समस्याओं को रोकने के लिए कोड को दोबारा लिखना या डॉक्यूमेंटेशन अपडेट करना। इस चरण में हेल्प डेस्क, टिकटिंग

सिस्टम (जैसे Jira), और संस्करण नियंत्रण का उपयोग किया जाता है। अच्छा रखरखाव सिस्टम की उम्र और उपयोगिता को बढ़ाता है।

SDLC के लोकप्रिय मॉडल (Popular SDLC Models) वॉटरफॉल, एजाइल, स्पाइरल

वॉटरफॉल मॉडल (Waterfall Model) –
सबसे पुराना और सरल, जिसमें प्रत्येक चरण अगले चरण से पहले पूरा होता है (जैसे झरना)। यह तब उपयुक्त है जब आवश्यकताएँ स्पष्ट और बदलने की संभावना न हो।

एजाइल मॉडल (Agile Model) –
आधुनिक और लोकप्रिय, जिसमें विकास छोटे-छोटे चक्रों (sprints) में होता है (Scrum, Kanban)। यह बदलती आवश्यकताओं के लिए लचीला है और ग्राहक को बार-बार डेमो दिखाता है।

स्पाइरल मॉडल (Spiral Model) –
जोखिम विश्लेषण पर केंद्रित, प्रत्येक चक्र में प्रोटोटाइप बनाकर जोखिम कम किया जाता है। V-Model – टेस्टिंग पर जोर, जहाँ प्रत्येक विकास चरण का एक समानांतर परीक्षण चरण होता है। RAD मॉडल (Rapid Application Development) – कम समय में प्रोटोटाइप और पुनरावृत्ति द्वारा तेजी से विकास। एजाइल मॉडल आजकल सबसे अधिक उपयोग किया जाता है।

MIS और SDLC के बीच संबंध
MIS (प्रबंधन सूचना प्रणाली) का विकास भी SDLC प्रक्रिया के माध्यम से ही किया जाता है। जब कोई संगठन एक नया MIS (जैसे इन्वेंटरी प्रबंधन प्रणाली या ग्राहक संबंध प्रबंधन) बनाने का निर्णय लेता है, तो वह SDLC के सभी सात चरणों से गुजरता है। आवश्यकता विश्लेषण में MIS की रिपोर्टिंग आवश्यकताओं को समझा जाता है, डिजाइन में डेटाबेस और डैशबोर्ड बनाए जाते हैं, टेस्टिंग में रिपोर्ट की सटीकता की जाँच की जाती है, और रखरखाव में नई रिपोर्ट जोड़ी जाती हैं। इस प्रकार, SDLC MIS विकास के लिए एक व्यवस्थित रूपरेखा प्रदान करता है। MIS पेशेवरों के लिए SDLC को समझना अनिवार्य है क्योंकि अधिकांश MIS परियोजनाएँ SDLC का पालन करती हैं।

तुलना सारांश (Summary Table for Exam)

पक्ष	विवरण
MIS परिभाषा	प्रबंधकों को निर्णय के लिए जानकारी देने वाली प्रणाली
MIS घटक	डेटा संग्रहण, प्रोसेसिंग, भंडारण, वितरण
MIS उदाहरण	सेल्स MIS, इन्वेंटरी MIS, फाइनेंशियल MIS
SDLC परिभाषा	सिस्टम विकास की चरणबद्ध प्रक्रिया
SDLC चरण (7)	योजना → आवश्यकता → डिजाइन → कोडिंग → टेस्टिंग → तैनाती → रखरखाव
प्रसिद्ध मॉडल	वॉटरफॉल, एजाइल (Scrum/Kanban), स्पाइरल, V-Model, RAD
MIS-SDLC संबंध	MIS को SDLC विधि से विकसित किया जाता है

फैक्ट फाइंडिंग प्रोसेस (Fact Finding Process)

फैक्ट फाइंडिंग वह प्रक्रिया है जिसमें सिस्टम एनालिस्ट मौजूदा सिस्टम के बारे में तथ्य, आवश्यकताएँ, समस्याएँ और उपयोगकर्ताओं की अपेक्षाएँ एकत्रित करता है। यह SDLC के आवश्यकता विश्लेषण (Requirement Analysis) चरण का सबसे महत्वपूर्ण हिस्सा है। इस प्रक्रिया का उद्देश्य सटीक, पूर्ण और अद्यतन जानकारी इकट्ठा करना है ताकि नया सिस्टम उपयोगकर्ताओं की वास्तविक आवश्यकताओं को पूरा कर सके। फैक्ट फाइंडिंग के बिना विकसित किया गया MIS या कोई भी सिस्टम विफल होने की संभावना अधिक होती है क्योंकि वह अधूरी या गलत जानकारी पर आधारित होता है। इस प्रक्रिया में सिस्टम एनालिस्ट को धैर्य, संचार कौशल और विश्लेषणात्मक क्षमता की आवश्यकता होती है।

फैक्ट फाइंडिंग प्रक्रिया के चरण (Steps of Fact Finding Process)

फैक्ट फाइंडिंग प्रक्रिया में निम्नलिखित चरण होते हैं: (1) आवश्यकता की पहचान (Identify Need) – क्यों और किस उद्देश्य से जानकारी चाहिए। (2) स्रोतों की पहचान (Identify

Sources) – जानकारी किससे (उपयोगकर्ता, प्रबंधक, दस्तावेज़) और कहाँ से लेनी है। (3) तकनीक का चयन (Select Technique) – कौन सी फैक्ट फाइंडिंग तकनीक (इंटरव्यू, प्रश्नावली, अवलोकन आदि) उपयुक्त है। (4) डेटा संग्रहण (Collect Data) – चयनित तकनीकों का उपयोग करके तथ्य एकत्र करना। (5) डेटा विश्लेषण (Analyze Data) – एकत्रित डेटा को छाँटना, सारांशित करना और विरोधाभासों को पहचानना। (6) निष्कर्ष निकालना (Draw Conclusions) – डेटा के आधार पर समस्याओं और समाधानों की पहचान। (7) प्रलेखन (Documentation) – सभी तथ्यों और निष्कर्षों को SRS (Software Requirement Specification) में दस्तावेजित करना। ये चरण SDLC के आवश्यकता चरण का अभिन्न अंग हैं।

फैक्ट फाइंडिंग तकनीकें (Fact Finding Techniques)

फैक्ट फाइंडिंग के लिए सिस्टम एनालिस्ट निम्नलिखित छह तकनीकों का उपयोग करता है:
(1) इंटरव्यू (Interview) – व्यक्तिगत या समूह में प्रश्न पूछना।
(2) प्रश्नावली (Questionnaire) – लिखित प्रश्नों का सेट भेजकर उत्तर प्राप्त करना।
(3) अवलोकन (Observation) – उपयोगकर्ताओं को काम करते हुए देखना।
(4) दस्तावेज़ विश्लेषण (Document Analysis) – मौजूदा रिपोर्ट, फॉर्म, नियम पुस्तिकाओं का अध्ययन।
(5) कार्यशाला / जेएडी (Workshop/JAD) – प्रमुख स्टेकहोल्डर्स के साथ समूह चर्चा सत्र।
(6) नमूना / सर्वेक्षण (Sampling/Survey) – बड़ी जनसंख्या से प्रतिनिधि नमूना लेकर डेटा संग्रह। प्रत्येक तकनीक के अपने फायदे और नुकसान हैं, और अक्सर संयोजन (combination) का उपयोग किया जाता है। इन तकनीकों का उपयोग SDLC के आवश्यकता चरण में सबसे अधिक होता है।

इंटरव्यू तकनीक (Interview Technique)

इंटरव्यू सबसे सामान्य और प्रभावी फैक्ट फाइंडिंग तकनीक है, जिसमें सिस्टम एनालिस्ट व्यक्तिगत रूप से उपयोगकर्ताओं या प्रबंधकों से प्रश्न पूछता है। इंटरव्यू दो प्रकार के होते हैं: असंरचित इंटरव्यू (Unstructured) – खुले प्रश्न (open-ended) पूछे जाते हैं, कोई पूर्व निर्धारित प्रश्नावली नहीं होती, जिससे गहरी जानकारी मिलती है। संरचित इंटरव्यू (Structured) – पूर्व निर्धारित प्रश्नों की सूची (closed-ended) होती है, जिससे तुलना और सांख्यिकीय विश्लेषण आसान होता है। इंटरव्यू के लाभ: लचीलापन, स्पष्टीकरण की संभावना, गैर-मौखिक संकेतों को पकड़ना, और उच्च प्रतिक्रिया दर। सीमाएँ: समय लेने वाला, महँगा, और एनालिस्ट के कौशल पर निर्भर। SDLC के आवश्यकता चरण में इंटरव्यू का व्यापक उपयोग होता है। अच्छे इंटरव्यू के लिए तैयारी (प्रश्न पूर्व निर्धारित करना), सक्रिय श्रवण, और नोट्स बनाना आवश्यक है।

प्रश्नावली तकनीक (Questionnaire Technique)

प्रश्नावली एक लिखित दस्तावेज़ होता है जिसमें प्रश्नों का एक सेट होता है, जिसे उपयोगकर्ताओं को भरकर लौटाना होता है। यह तकनीक तब उपयुक्त है जब उपयोगकर्ता भौगोलिक रूप से बिखरे हुए हों या बड़ी संख्या में हों। प्रश्न दो प्रकार के होते हैं: खुले प्रश्न (Open-ended) – उत्तरदाता स्वतंत्र रूप से अपनी राय लिख सकता है, जैसे "सिस्टम की सबसे बड़ी समस्या क्या है?"। बंद प्रश्न (Closed-ended) – पूर्व निर्धारित विकल्प (हाँ/नहीं, रेटिंग स्केल, बहुविकल्पी) होते हैं, जैसे "क्या आप रिपोर्ट से संतुष्ट हैं? (हाँ/नहीं/कभी-कभी)"। प्रश्नावली के लाभ: कम लागत, बड़ी जनसंख्या तक पहुँच, अनामिकता (गोपनीयता), और विश्लेषण में आसानी। सीमाएँ: कम प्रतिक्रिया दर, स्पष्टीकरण की कोई संभावना नहीं, और सतही जानकारी। SDLC में प्रश्नावली का उपयोग प्रारंभिक जानकारी इकट्ठा करने या बड़े पैमाने पर सर्वेक्षण के लिए किया जाता है। एक अच्छी प्रश्नावली में स्पष्ट निर्देश, सरल भाषा, और तार्किक क्रम होना चाहिए।

अवलोकन तकनीक (Observation Technique)

अवलोकन तकनीक में, सिस्टम एनालिस्ट उपयोगकर्ताओं को उनके दैनिक कार्य करते हुए प्रत्यक्ष रूप से देखता है, बिना किसी हस्तक्षेप के। यह तकनीक उन तथ्यों को उजागर करती है जो उपयोगकर्ता इंटरव्यू में बताना भूल सकते हैं या जानबूझकर छिपा सकते हैं। अवलोकन दो प्रकार के होते हैं: स्पष्ट अवलोकन (Overt Observation) – उपयोगकर्ताओं को पता होता है कि उन्हें देखा जा रहा है, जिससे हॉथोर्न प्रभाव (behaviour change) हो सकता है। गुप्त अवलोकन (Covert Observation) – उपयोगकर्ताओं को पता नहीं होता, जिससे वास्तविक व्यवहार देखने को मिलता है, लेकिन यह नैतिक रूप से संदिग्ध हो सकता है। अवलोकन के लाभ: उच्च सटीकता, वास्तविक दुनिया का डेटा, और अलिखित नियमों की पहचान। सीमाएँ: समय लेने वाला, केवल वर्तमान प्रक्रियाएँ दिखता है (भूत या भविष्य नहीं), और उपस्थिति से व्यवहार बदल सकता है। SDLC में अवलोकन का उपयोग विशेष रूप से तब किया जाता है जब मौजूदा प्रक्रिया में अक्षमताओं या अड़चनों (bottlenecks) की पहचान करनी हो। अवलोकन के दौरान विस्तृत नोट्स, टाइमिंग, और चेकलिस्ट का उपयोग करना चाहिए।

दस्तावेज़ विश्लेषण तकनीक (Document Analysis)

दस्तावेज़ विश्लेषण में, सिस्टम एनालिस्ट मौजूदा सिस्टम से संबंधित सभी लिखित सामग्रियों का अध्ययन करता है, जैसे रिपोर्ट, फॉर्म, नियम पुस्तिकाएँ, नीति दस्तावेज़, प्रक्रिया मैनुअल, और पिछली परियोजना के दस्तावेज़। यह तकनीक उन तथ्यों को प्रकट करती है जो उपयोगकर्ता स्मरण नहीं रख सकते या जानबूझकर छिपाते हैं। दस्तावेज़ विश्लेषण के लाभ: कम लागत, निष्पक्ष (unbiased), ऐतिहासिक डेटा उपलब्ध, और कहीं भी किया जा सकता है। सीमाएँ: दस्तावेज़ पुराने या अद्यतन नहीं हो सकते, कुछ प्रक्रियाएँ दस्तावेजित ही नहीं होतीं, और दस्तावेज़ों में विरोधाभास हो सकता है। SDLC के आवश्यकता चरण में, दस्तावेज़ विश्लेषण का उपयोग मौजूदा सिस्टम की आधार रेखा (baseline) स्थापित करने के लिए किया जाता है। विश्लेषण के दौरान, एनालिस्ट को दस्तावेज़ों की

सटीकता, पूर्णता, स्थिरता और अद्यतनता की जाँच करनी चाहिए। इसका आउटपुट "प्रलेखन सारांश रिपोर्ट" होता है।

डेटा एनालिसिस (Data Analysis)

डेटा एनालिसिस वह प्रक्रिया है जिसमें एकत्रित किए गए कच्चे डेटा (raw data) को साफ (cleaning), रूपांतरित (transforming), और मॉडलिंग (modeling) करके उपयोगी जानकारी, निष्कर्ष और निर्णय लेने के लिए सार्थक अंतर्दृष्टि (insights) निकाली जाती है। SDLC में, डेटा एनालिसिस मुख्यतः आवश्यकता विश्लेषण और डिजाइन चरणों में किया जाता है, जहाँ फैक्ट फाइंडिंग से प्राप्त डेटा का विश्लेषण करके सिस्टम की कार्यात्मक आवश्यकताओं को परिष्कृत (refine) किया जाता है। MIS के संदर्भ में, डेटा एनालिसिस का अर्थ है संगठन के विभिन्न स्रोतों से आने वाले डेटा का विश्लेषण करके रिपोर्ट, ट्रेंड, और निर्णय-समर्थन जानकारी तैयार करना। डेटा एनालिसिस में सांख्यिकीय तरीके, डेटा माइनिंग, और प्रेडिक्टिव मॉडलिंग जैसी तकनीकों का उपयोग होता है। एक अच्छा डेटा एनालिसिस सिस्टम को डेटा-ड्रिवेन (data-driven) बनाता है और MIS को अधिक प्रभावी बनाता है।

डेटा एनालिसिस के चरण (Steps of Data Analysis)

डेटा एनालिसिस प्रक्रिया में निम्नलिखित पाँच चरण होते हैं:

(1) आवश्यकताएँ निर्धारित करना (Requirement Determination) – कौन से प्रश्नों के उत्तर चाहिए? कौन सा डेटा आवश्यक है?

(2) डेटा संग्रहण (Data Collection) – फैक्ट फाइंडिंग तकनीकों (इंटरव्यू, प्रश्नावली, दस्तावेज़) से डेटा इकट्ठा करना।

(3) डेटा सफाई (Data Cleaning) – अधूरे, दोहरे, गलत या असंगत डेटा को हटाना या सुधारना।

(4) डेटा विश्लेषण (Data Analysis) – डेटा पर सांख्यिकीय या गुणात्मक तरीकों को लागू करना, जैसे औसत, प्रतिशत, रुझान विश्लेषण, सहसंबंध।

(5) व्याख्या और प्रस्तुति (Interpretation & Presentation) – परिणामों को चार्ट, ग्राफ, डैशबोर्ड, या रिपोर्ट के रूप में प्रबंधकों को प्रस्तुत करना। SDLC में, यह विश्लेषण SRS (Software Requirement Specification) दस्तावेज़ को अंतिम रूप देने में सहायक होता है। MIS में, डेटा एनालिसिस रीयल-टाइम डैशबोर्ड और निर्णय समर्थन प्रणाली (DSS) का आधार बनता है।

डेटा डिक्शनरी (Data Dictionary)

डेटा डिक्शनरी (जिसे मेटाडेटा रिपोजिटरी भी कहते हैं) एक केंद्रीकृत भंडार है जो डेटाबेस या सिस्टम में संग्रहीत सभी डेटा तत्वों (data elements) के बारे में विस्तृत जानकारी (मेटाडेटा) रखता है। यह डेटा के बारे में डेटा है – जैसे प्रत्येक डेटा फील्ड का नाम, प्रकार (नंबर, टेक्स्ट, तारीख), आकार (अधिकतम लंबाई), प्रारूप (format), डिफ़ॉल्ट मान, वैधता नियम (जैसे >0, <100), और विवरण। SDLC के डिजाइन चरण (Design Phase) में डेटा डिक्शनरी बनाई जाती है और कोडिंग, टेस्टिंग और रखरखाव में इसका उपयोग किया जाता है। डेटा डिक्शनरी के लाभ: डेटा अखंडता (integrity) सुनिश्चित करना, डेटा अतिरेक (redundancy) को कम करना, सिस्टम के सभी डेवलपर्स के बीच साझा शब्दावली प्रदान करना, और परिवर्तन प्रबंधन (change management) में सहायक होना। उदाहरण: "Customer_ID: डेटा प्रकार INTEGER, आकार 10, अद्वितीय, NOT NULL, विदेशी कुंजी (foreign key) से संदर्भित"।

डेटा डिक्शनरी के घटक (Components of Data Dictionary)

एक पूर्ण डेटा डिक्शनरी में निम्नलिखित जानकारी होती है:

(1) डेटा तत्व का नाम (Data Element Name) – जैसे Employee_Name, Salary, Date_of_Joining।

(2) डेटा प्रकार (Data Type) – CHAR, VARCHAR, INT, DATE, BOOLEAN, FLOAT।

(3) आकार (Size/Length) – अधिकतम वर्णों या बाइट्स की संख्या, जैसे VARCHAR(50) – 50 वर्ण।

(4) प्रारूप (Format) – तारीख के लिए YYYY-MM-DD, फोन नंबर के लिए (XXX) XXX-XXXX।

(5) डिफ़ॉल्ट मान (Default Value) – यदि उपयोगकर्ता कोई मान नहीं डालता तो क्या डाला जाए (जैसे Status = 'Active')।

(6) वैधता नियम (Validation Rules) – जैसे Age BETWEEN 18 AND 60, Salary > 0, Email MUST CONTAIN '@'।

(7) अनुमत मान (Allowed Values) – जैसे Gender IN ('M', 'F', 'Other')।

(8) अद्वितीयता (Uniqueness) – क्या यह फील्ड दोहरा हो सकता है (Unique / Not Unique)।

(9) अशक्त अनुमति (Null Allowed) – क्या यह फील्ड खाली हो सकता है (NULL / NOT NULL)।

(10) तालिका और स्तंभ संदर्भ (Table & Column Reference) – यह फील्ड किस तालिका और स्तंभ में संग्रहीत है। ये सभी घटक डेटाबेस डिज़ाइन और सिस्टम विकास के लिए आवश्यक हैं।

निर्णय तालिका (Decision Table)

निर्णय तालिका (Decision Table) एक सारणीबद्ध (tabular) प्रतिनिधित्व है जो विभिन्न स्थितियों (conditions) और उनके संगत क्रियाओं (actions) के बीच तार्किक संबंध को दर्शाती है। यह जटिल व्यावसायिक नियमों (business rules) और निर्णय तर्क (decision logic) को स्पष्ट, संक्षिप्त और त्रुटि-मुक्त रूप में प्रस्तुत करने का एक शक्तिशाली उपकरण है। SDLC के आवश्यकता विश्लेषण और डिजाइन चरणों में निर्णय तालिका का उपयोग उन परिस्थितियों में किया जाता है जहाँ कई इनपुट स्थितियों के आधार पर कई आउटपुट क्रियाएँ होती हैं (जैसे लोन स्वीकृति नियम, टैक्स गणना, छूट नियम)। निर्णय तालिका के चार मुख्य भाग होते हैं: स्थिति स्टंप (Condition Stub) – सभी स्थितियों की सूची, क्रिया स्टंप (Action Stub) – सभी क्रियाओं की सूची, स्थिति प्रविष्टियाँ (Condition Entries) – प्रत्येक स्थिति के लिए Y (हाँ) / N (नहीं) या अन्य मान, और क्रिया प्रविष्टियाँ (Action Entries) – बताती हैं कि किस नियम के लिए कौन सी क्रिया करनी है। निर्णय तालिका के लाभ: पूर्णता (सभी संभावित संयोजनों की जाँच), स्पष्टता (जटिल तर्क को समझना आसान), और प्रोग्रामर के लिए सीधे कोड में बदलने योग्य।

निर्णय तालिका का उदाहरण एवं संरचना

निर्णय तालिका की संरचना को एक उदाहरण से समझते हैं: एक कंपनी में लोन स्वीकृति नियम – शर्तें: (1) Income > 30,000? (2) CIBIL स्कोर > 750? (3) Existing Loan?। क्रियाएँ: (A) Loan Approved, (B) Loan Rejected, (C) Manager Approval Required। नियम 1: Y, Y, N → Loan Approved। नियम 2: Y, N, N → Loan Rejected। नियम 3: Y, Y, Y → Manager Approval। नियम 4: N, Y, N → Loan Rejected। नियम 5: N, N, N → Loan Rejected। इस प्रकार तालिका में 3 शर्तों के लिए 2^3 = 8 संभावित नियम होते हैं। SDLC में, निर्णय तालिका पूर्णता (completeness) सुनिश्चित करती है – कोई भी स्थिति संयोजन अनदेखा नहीं रहता, अतिरेक (redundancy) और विरोधाभास (contradiction) को पहचानती है। निर्णय तालिका को सीधे प्रोग्रामिंग में if-else या switch-case स्टेटमेंट्स में बदला जा सकता है। यह प्रलेखन (documentation) और रखरखाव (maintenance) के लिए भी अत्यंत उपयोगी है क्योंकि व्यावसायिक नियम बदलने पर केवल तालिका के कुछ कॉलम बदलने होते हैं।

निर्णय वृक्ष (Decision Tree)

निर्णय वृक्ष (Decision Tree) एक चित्रात्मक (graphical) प्रतिनिधित्व है जो निर्णय नियमों को एक पेड़ (tree) के रूप में दिखाता है, जिसमें प्रत्येक आंतरिक नोड एक स्थिति (condition) को, प्रत्येक शाखा (branch) एक निर्णय (decision) या परीक्षण (test) के परिणाम को, और प्रत्येक पत्ती नोड (leaf node) एक क्रिया (action) या निष्कर्ष को दर्शाता है। निर्णय वृक्ष का उपयोग SDLC के आवश्यकता विश्लेषण और डिजाइन चरणों में जटिल व्यावसायिक तर्क को सरल और दृश्य (visual) रूप में प्रस्तुत करने के लिए किया जाता है। निर्णय वृक्ष निर्णय तालिका के समान ही उद्देश्य रखता है, लेकिन अंतर यह है कि निर्णय वृक्ष अधिक सहज (intuitive) और समझने में आसान होता है, विशेष रूप से गैर-तकनीकी उपयोगकर्ताओं के लिए। उदाहरण: लोन स्वीकृति प्रक्रिया – रूट नोड (क्या Income > 30,000?) → यदि हाँ तो अगला नोड (CIBIL > 750?) → यदि हाँ तो लोन स्वीकृत, अन्यथा अस्वीकृत। निर्णय वृक्ष के लाभ: दृश्य स्पष्टता, सरलता, और गलतियों को ढूँढ़ना आसान। सीमा: जब नियम बहुत अधिक हों (100+) तो पेड़ बहुत बड़ा और जटिल हो सकता है।

निर्णय तालिका और निर्णय वृक्ष में अंतर

निर्णय तालिका और निर्णय वृक्ष दोनों ही व्यावसायिक नियमों को प्रदर्शित करने के उपकरण हैं, लेकिन उनमें अंतर है:

(1) रूप (Format) – निर्णय तालिका सारणीबद्ध (tabular) है, निर्णय वृक्ष चित्रात्मक (graphical) है।

(2) जटिलता (Complexity) – निर्णय तालिका कई स्थितियों (10-15) को आसानी से संभाल सकती है, जबकि निर्णय वृक्ष 4-5 स्थितियों से अधिक होने पर बहुत बड़ा हो जाता है। (3) पूर्णता (Completeness) – निर्णय तालिका पूर्णता (सभी संयोजनों की जाँच) सुनिश्चित करना आसान बनाती है, निर्णय वृक्ष में छूटे हुए नियमों को देखना मुश्किल होता है। (4) उपयोगकर्ता – निर्णय वृक्ष गैर-तकनीकी उपयोगकर्ताओं के लिए अधिक सहज है, जबकि निर्णय तालिका प्रोग्रामर और सिस्टम एनालिस्ट के लिए अधिक सटीक है।

(5) SDLC में उपयोग – दोनों का उपयोग आवश्यकता विश्लेषण और डिजाइन चरणों में किया जाता है, प्रायः संयोजन में (पहले निर्णय तालिका बनाएँ, फिर उसे निर्णय वृक्ष में बदलें)। MIS में, इनका उपयोग व्यावसायिक नियमों (जैसे डिस्काउंट पॉलिसी, लोन अप्रूवल, अटेंडेंस नियम) को प्रलेखित करने के लिए किया जाता है।

फॉर्म डिजाइन (Form Design)

फॉर्म डिजाइन का अर्थ है डेटा इनपुट और आउटपुट के लिए उपयोगकर्ता इंटरफेस (UI) स्क्रीन या कागज़ी फॉर्मों को इस प्रकार डिज़ाइन करना कि डेटा एंट्री सरल, सटीक, त्रुटि-रहित और कुशल हो। SDLC के डिजाइन चरण (Design Phase) में फॉर्म डिजाइन किया जाता है और यह सिस्टम के उपयोगकर्ता स्वीकार्यता (user acceptance) के लिए सबसे महत्वपूर्ण कारकों में से एक है। MIS

के संदर्भ में, फॉर्म दो प्रकार के होते हैं: इनपुट फॉर्म (Input Forms) – जहाँ उपयोगकर्ता डेटा दर्ज करता है (जैसे ग्राहक पंजीकरण फॉर्म), और आउटपुट फॉर्म (Output Forms) – जहाँ सिस्टम डेटा प्रदर्शित करता है (जैसे बिक्री रिपोर्ट, इन्वॉइस)। एक अच्छा फॉर्म डिजाइन डेटा एंट्री की गति बढ़ाता है, त्रुटियाँ कम करता है, उपयोगकर्ता प्रशिक्षण का समय घटाता है

UNIT-V (MIS & System Development Life Cycle)

SDLC & Fact Finding (सिस्टम विकास और तकनीकें):

1-सिस्टम डेवलपमेंट लाइफ साइकिल के विभिन्न घटकों की विस्तार से व्याख्या कीजिए। (D.C.A. - June 2023)

2-एसडीएलसी (SDLC) के विभिन्न चरण क्या हैं? (PGDCA - May/June 2019)

3-फैक्ट फाइंडिंग प्रोसेस एवं तकनीकों को उदाहरण सहित विस्तारपूर्वक समझाइए। (D.C.A. - June 2023)

4-फीजिबिलिटी स्टडी क्या है? प्रत्यक्ष/अप्रत्यक्ष, लागत/लाभ अनुपात की व्याख्या कीजिए। (PGDCA - May/June 2019)

Questions for DCA 2 IT trends only (मल्टीमीडिया और डिस्ट्रिब्यूटेड सिस्टम)

मल्टीमीडिया क्या है? मल्टीमीडिया हार्डवेयर और सॉफ्टवेयर की आवश्यकता को समझाइए। (D.C.A. - June 2023)

मल्टीमीडिया की विशेषताओं और उपयोगों को परिभाषित करें। डिजिटल वीडियो और उनके प्रसंस्करण के तरीके समझाइए। (D.C.A. - June 2023)

मल्टीमीडिया की रोज के दैनिक जीवन में क्या उपयोगिता है? (D.C.A. - Jan 2019)

मल्टीमीडिया क्या है? मल्टीमीडिया के विभिन्न सॉफ्टवेयरों की व्याख्या कीजिए। (D.C.A. - Feb 2019)

मल्टीमीडिया क्या है? विभिन्न प्रकार के मल्टीमीडिया को उनके उपयोग के आधार पर समझाइए। (D.C.A. - Feb 2019)

मल्टीमीडिया क्या है? मल्टीमीडिया सॉफ्टवेयर क्या हैं? (D.C.A. - Feb 2019 / May/June 2019)

टेक्स्ट डॉक्यूमेंट बनाने के लिए विभिन्न प्रकार के फोंट का वर्णन करें। (D.C.A. - June 2023)

मल्टीमीडिया में ध्वनि का क्या महत्व है ? मल्टीमीडिया में मोना और स्टीरियो ध्वनि के बीच में अंतर करो। (D.C.A. - June 2023)

वर्चुअल वास्तविकता का वर्तमान में क्या उपयोग है? विस्तार से समझाइए। (D.C.A. - Jan 2019)

वर्चुअल रियलिटी की विस्तार से व्याख्या कीजिए। वर्चुअल रियलिटी के उपयोगों को भी समझाइए। (D.C.A. - Feb 2019)

वर्चुअल रियलिटी पर संक्षिप्त टिप्पणी लिखिए, एवं इसके उपयोग भी लिखिए। (D.C.A. - Feb 2019)

मल्टीमीडिया संलेखन उपकरण क्या हैं? (D.C.A. - Feb 2019 / May/June 2019)

वीडियो फाइलों पर एक नोट लिखिए। वीडियो फाइलों के स्वरूपों का भी वर्णन कीजिए। (D.C.A. - Feb 2019 / May/June 2019)

मिडी (संगीतकार संगीत वाद्ययंत्र डिजिटल इंटरफेस) की अवधारणा को समझाइए। (D.C.A. - Feb 2019)

मिडी (म्यूजिकल इंस्ट्रूमेंट डिजिटल इंटरफेस) की अवधारणा को समझाइए। (D.C.A. - May/June 2019)

डिस्ट्रिब्यूटेड सिस्टम से आप क्या समझते हैं? डिस्ट्रिब्यूटेड सिस्टम के फायदे और नुकसान लिखिए। (D.C.A. - Jan 2019)

डिस्ट्रिब्यूटेड सिस्टम क्या होता है? डिस्ट्रिब्यूटेड सिस्टम के फायदे और नुकसान लिखिए। (D.C.A. - Jan 2019)

डिस्ट्रिब्यूटेड सिस्टम क्या है? इसके फायदे और नुकसानों की व्याख्या कीजिए। (D.C.A. - Feb 2019)

डिस्ट्रिब्यूटेड सिस्टम क्या होता है? इसके लाभ एवं हानियों के साथ विस्तार से बताइए। (D.C.A. - Feb 2019)

इन्फॉर्मेशन सिक्योरिटी मैनेजमेंट डिजिटल डिवाइड क्या है? समझाइए। (D.C.A. - Feb 2019)