Structured Analysis, Decision Support and the Systems Proposal

MS (AI & DS) | DFDs, data dictionaries, process specifications, structured decisions, DSS |

Anurag Srivastava (DoES) ABV-IIITM Gwalior

Plan for the 60 minutes

Segment	Minutes	What we cover
Introduction	0–5	Why structured analysis matters; the toolkit
Data flow diagrams	5–17	Symbols, rules, context diagram, levelling
Data dictionaries	17–24	Purpose, notation, examples, modern equivalents
Process specs, structured decisions	24–35	Structured English, decision tables, trees
Semi-structured DSS	35–46	Decision types, architecture, DSS in the AI era
Systems proposal	46–53	Contents, cost-benefit, writing and presenting
Activity and wrap-up	53–60	Mini-case in pairs; key takeaways

Learning outcomes

1

Model

Draw context and level-1 data flow diagrams.

2

Document

Write data dictionary entries and process specifications.

3

Distinguish

Separate structured from semi-structured decisions; explain DSS.

4

Propose

Structure a systems proposal with a cost-benefit case.

By the end: you can turn a vague business problem into documented, reviewable and fundable requirements.

The analysis toolkit: how the pieces fit



DFD	Data dictionary	Process specs	Decision support	Proposal
Where does data move?	What does each data item mean?	How is each process carried out?	How do we support semi-structured decisions?	How do we sell the solution?

Idea: structured analysis turns a vague problem into diagrams, definitions and rules everyone can check.

DFD: four symbols



External entity

Source or sink of data outside the system.



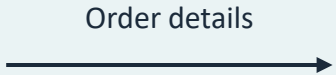
Process

Transforms inputs into outputs. Named verb + noun.



Data store

Data held at rest for later use.



Data flow

Data in motion. Always named.

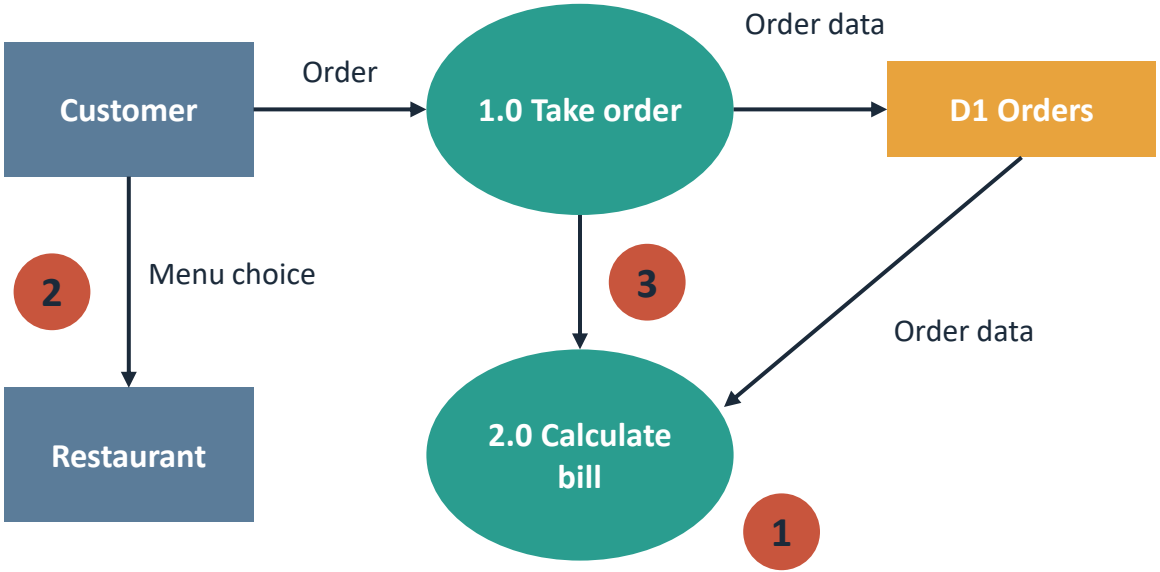
Notation: Gane-Sarson draws processes as rounded rectangles; Yourdon-DeMarco uses circles. The ideas are identical.

DFD rules and common errors

Error or rule	What it means	How to fix
Black hole	Process has inputs but no outputs	Add the output flow
Miracle	Process has outputs but no inputs	Add the missing input
Grey hole	Outputs cannot be derived from inputs	Add the missing data
Illegal flow	Entity to entity, or store to store	Route via a process
Unnamed flow	Arrow carries no label	Name every flow

Check: every process needs at least one input and one output, and every flow needs a name.

Case: spot the errors in this DFD



Answers

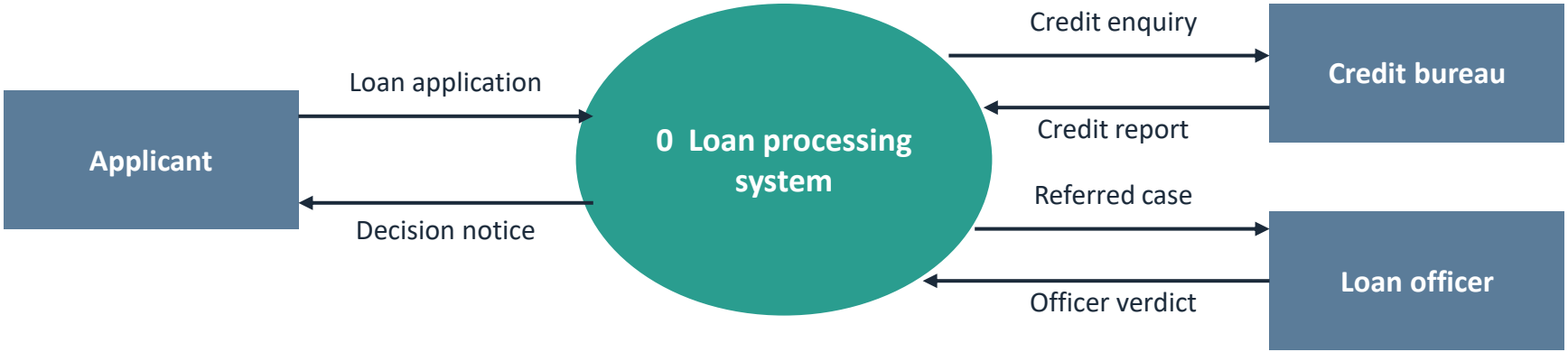
1 Black hole: 2.0 has input but no output.

2 Illegal flow: entity to entity must pass through a process.

3 Unnamed flow: every arrow needs a label.

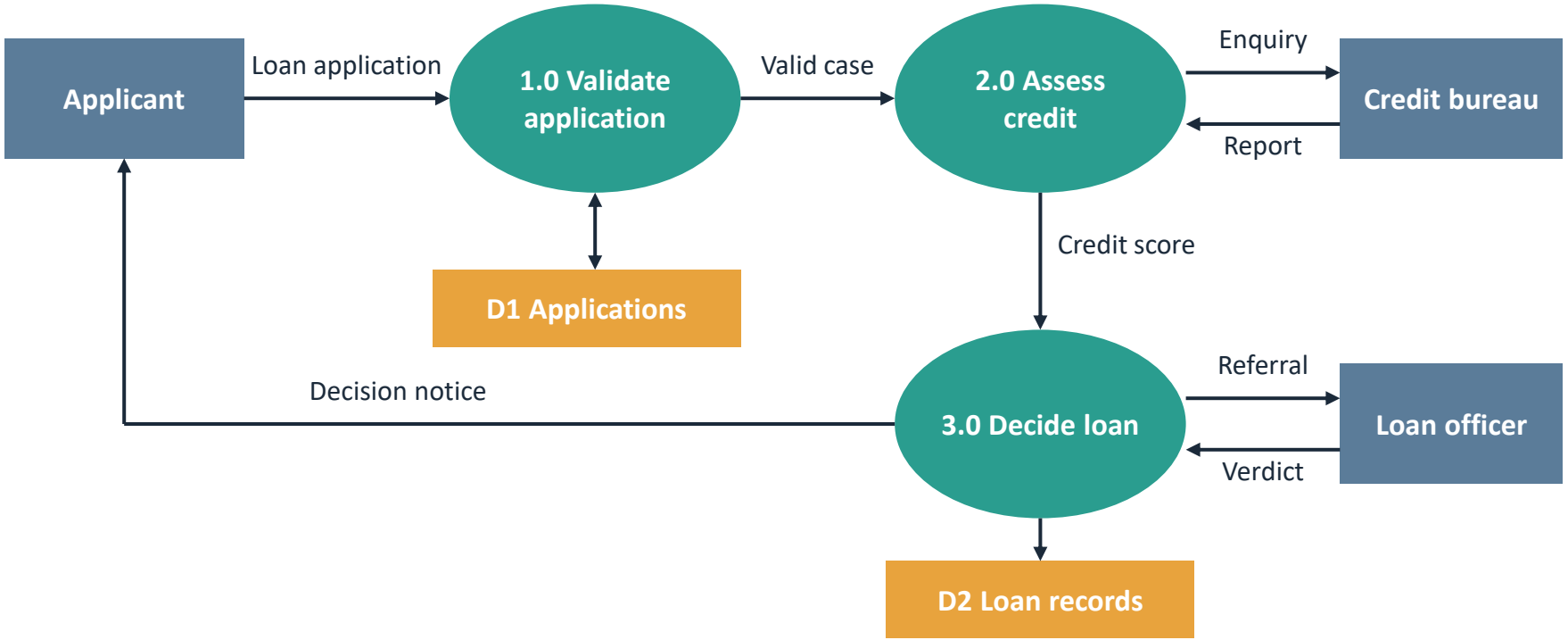
Try first: cover the answers and find the three errors in 60 seconds.

Context diagram (Level 0): loan processing

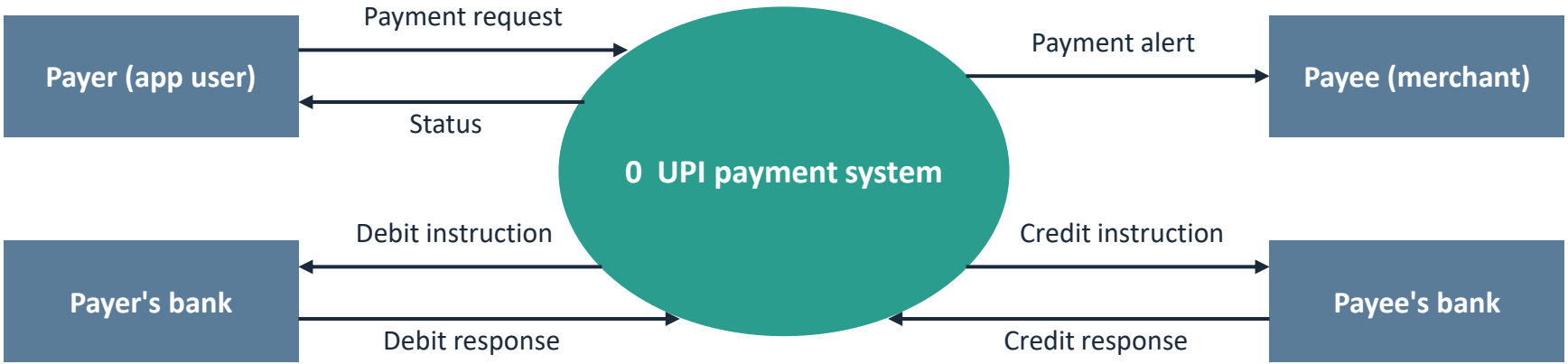


Purpose: show the boundary: what is inside, who it talks to, and what data crosses it.

Level-1 DFD: exploding the single process



Case: UPI payment, context diagram



Note: simplified and illustrative. Banks and merchants are external entities; only data that crosses the boundary is shown.

Building DFDs: steps, levelling and balancing

1

List

Entities, flows and stores from the requirements.

2

Context

Draw one process and its entities.

3

Explode

Split into 3 to 7 sub-processes.

4

Level down

Repeat until processes are primitive.

5

Balance

Child inputs and outputs equal the parent's.

6

Name

Verb + noun, numbered 1.0, 1.1, 1.2.

Logical vs physical: start logical (what happens); add physical detail (who, where, how) only for design.

What is a data dictionary?

A structured repository that defines every data element, data structure, data flow and data store used in a system.

Data element

Smallest unit: Loan_Amount.

Data structure

Group of elements: Loan_Application.

Flow and store

What each arrow and store carries.

Metadata

Type, length, range, owner, aliases.

Why: one agreed definition per term prevents conflicting reports and failed integrations.

Data dictionary notation

Symbol	Meaning	Example
=	is composed of	Address = Street + City + PIN
+	and (sequence)	Name = First_Name + Last_Name
[|]	either / or (selection)	[Salaried | Self_Employed]
( )	optional	(Middle_Name)
{ }	repetition	{Loan_Item}
* *	comment	*primary key*

Tip: composite structures are built from elements; elements are defined once, with type, length and range.

Data dictionary: worked example

```
Loan_Application =
  Applicant_ID +
  Applicant_Name +
  Loan_Amount +
  Tenure_Months +
  [Salaried | Self_Employed] +
  Monthly_Income +
  (Co_Applicant_ID) +
  {Existing_Loan}
```

Element: Loan_Amount
Alias: Requested_Amount
Type: Numeric, 9 digits
Range: 10,000 to 5,000,000
Used in: 1.0 Validate application, 2.0 Assess credit

Note: entries link back to DFD processes, so changes to a data item show which processes are affected.

Case: one field, three meanings

Sales

customer_id = email address

Billing

customer_id = 10-digit account number

Support

customer_id = phone number

Impact on an ML project

Joins fail or silently mismatch, giving duplicate rows, wrong features and an unreliable model.

Dictionary fix

Define Customer_ID once; list aliases Email_Key, Account_No, Phone; keep a mapping table.

Lesson: many data-quality bugs are definition bugs. Write the definition before you write the join.

Data dictionaries in modern data and AI work

Schemas

Typed column definitions and constraints.

e.g. SQL DDL, dbt docs

Data catalogs

Searchable metadata, owners and tags.

e.g. DataHub, Unity Catalog

Lineage

Where data came from and what uses it.

automated DFD

ML documentation

Describe datasets and models for reuse.

datasheets, model cards

Link: data quality, governance and explainability all begin with knowing exactly what each field means.

Process specifications (mini-specs)

Structured English

Plain language with IF, ELSE, FOR and WHILE.

Best for sequences and loops

Decision table

Conditions and actions in a grid of rules.

Best for many condition combinations

Decision tree

Branching diagram from conditions to outcomes.

Best for step-by-step branching

Mini-spec: describes how one primitive DFD process turns its inputs into outputs, precisely enough to build and test.

Structured decisions

A decision is structured when

- Inputs are known in advance
- Rules are explicit and agreed
- Same inputs always give the same outcome
- It can be automated or put in a rule engine

The analyst's job

- Elicit the rules from experts
- Write them in a precise notation
- Test them against real examples
- Flag cases the rules cannot cover

Examples: reorder stock below a threshold, compute tax, grade a quiz, check basic loan eligibility.

Structured English

```
IF existing_default = YES
   REJECT application
ELSE IF credit_score >= 700
   IF income >= 3 x EMI
      APPROVE application
   ELSE
      REFER to loan officer
ELSE
   IF income >= 3 x EMI
      REFER to loan officer
   ELSE
      REJECT application
```

Writing rules

- Use only sequence, decision and repetition
- Verbs in capitals, indent for nesting
- Use names from the data dictionary
- No technical jargon

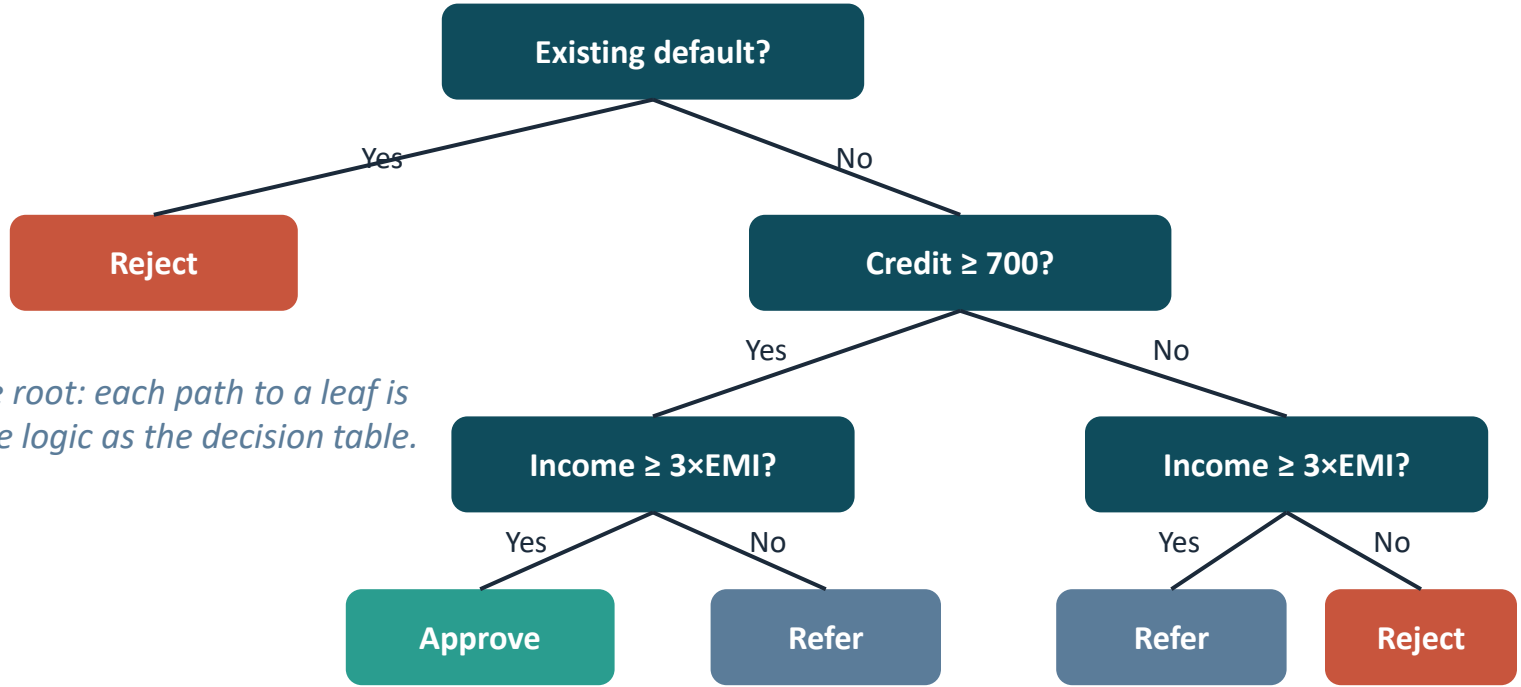
Limit: deeply nested IFs become hard to read; switch to a decision table when conditions multiply.

Decision tables

Condition or action	R1	R2	R3	R4	R5
Existing default?	N	N	N	N	Y
Credit score ≥ 700?	Y	Y	N	N	–
Income ≥ 3 × EMI?	Y	N	Y	N	–
Approve	X				
Refer to officer		X	X		
Reject				X	X

Reading: Y = yes, N = no, – = does not matter. A complete table covers every combination of conditions.

Decision trees



Read from the root: each path to a leaf is one rule. Same logic as the decision table.

Case: e-commerce discount rules

Condition or action	R1	R2	R3	R4
Member?	Y	Y	N	N
Order ≥ ₹2,000?	Y	N	Y	N
Discount 15%	X			
Discount 10%		X		
Discount 5%			X	
No discount				X

Try it: a member orders ₹1,500, so R2 fires: 10% off. Add a festival-sale condition and the table needs $2^3 = 8$ rules.

Choosing a notation

Tool	Strength	Weakness	Use when
Structured English	Familiar; handles loops	Hard with many conditions	Sequences, calculations
Decision table	Complete; exposes gaps	Poor for sequence	Many condition combinations
Decision tree	Visual and easy to explain	Grows fast	Few conditions, branching

Practice: use a table to check completeness, then draw a tree or write structured English to communicate it.

Three kinds of decisions

Type	Rules	Example	Best support
Structured	Known, fixed	Reorder stock; basic eligibility	Automate (process spec)
Semi-structured	Partly known	Credit referral, budget allocation	Decision support system
Unstructured	Unknown	New business strategy	Expert judgement, discussion

Key point: semi-structured means part of the decision can be computed and part needs human judgement.

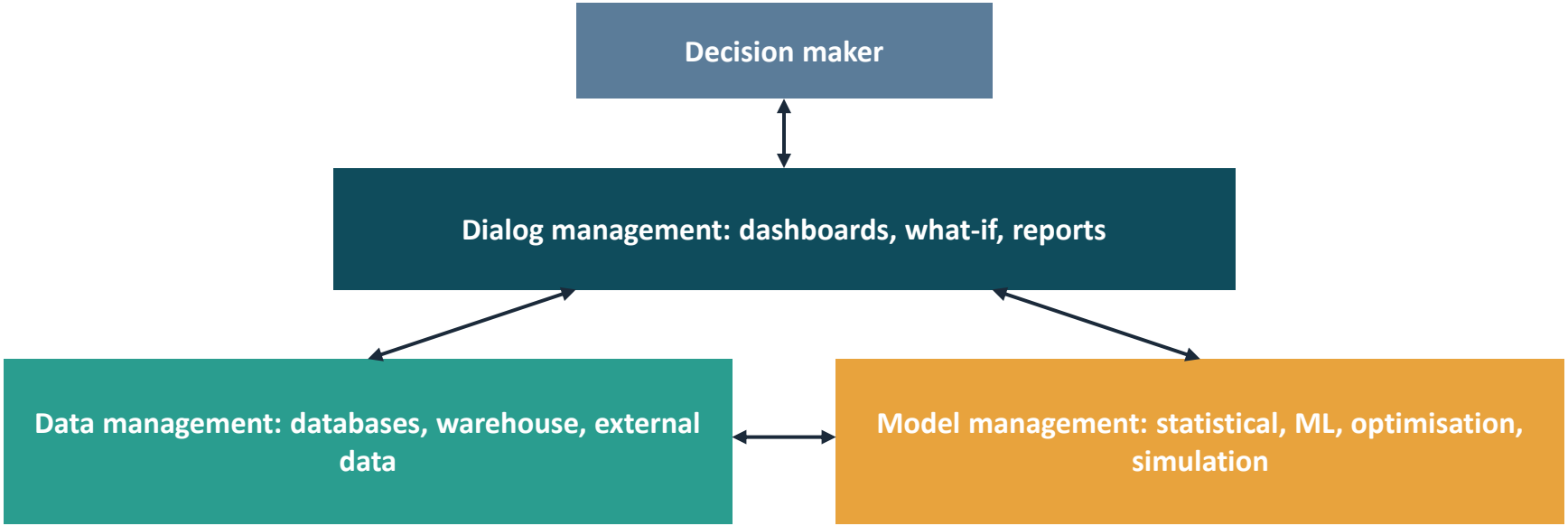
What is a decision support system?

An interactive computer-based system that helps decision makers use data and models to solve semi-structured problems.

1. **Interactive**
   User explores and re-runs
2. **Supports judgement**
   Human keeps the final call
3. **Flexible**
   Adapts as questions change

Contrast: a transaction system records what happened; a DSS helps decide what to do next.

DSS architecture



Flow: the user asks a question, models run on managed data, and results return through the dialog layer.

DSS in the AI era

Model base

ML models predict and score.

e.g. default risk

Data layer

Warehouse or lakehouse feeds the models.

fresh and governed

Explainability

Show reasons, not just scores.

e.g. SHAP values

Human oversight

Analysts can override; overrides are logged.

audit trail

Principle: the machine handles the structured part; the human handles the judgement part.

Case: a DSS for loan referrals

1

System scores and explains

Model gives default probability, top factors and similar past cases.

2

Analyst judges

Runs what-if on income or tenure; overrides with a recorded reason.

3

Loop learns

Decisions and outcomes feed monitoring and retraining.

Semi-structured: rules settle the clear cases; the DSS and the analyst handle the grey zone.

Case: a clinical DSS that fell short

What was reported A widely used commercial sepsis-alert model was externally evaluated at one hospital (2021). It performed worse than its developer reported, missing many cases while raising many alerts.

Validate locally

Test on your own patients and data.

Watch alert fatigue

Too many alerts get ignored.

Monitor and retrain

Performance drifts over time.

Keep humans in the loop

Clinicians can question the score.

Lesson: a DSS is only as good as its validation. Never trust a vendor score without testing it locally.

The systems proposal

Purpose

Persuade management to approve and fund the solution.

Audience

Executives, users, IT and finance.

Timing

End of analysis, before design and build.

Outcome

A go or no-go decision and agreed scope.

Remember: decision makers read the executive summary first, and often only that.

What goes into a systems proposal

Section	What it contains
Cover letter and title page	Purpose, author, date
Executive summary	Problem, recommendation and cost on one page
Problem and scope	Current system, objectives, boundaries
Findings	DFDs, data dictionary, requirements
Alternatives	Two or three options with pros and cons
Recommendation	Chosen option with cost-benefit analysis
Plan and risks	Timeline, staffing, risks, mitigation
Appendices	Detailed diagrams and supporting data

Evaluating alternatives: cost-benefit

Item	₹ lakh
Development	8
Hardware	2
Training	1
One-time cost	**11**
Annual saving	5
Annual maintenance	1
Net annual benefit	**4**

Payback

One-time cost ÷ net annual benefit: 11 ÷ 4 = 2.75 years.

NPV

Discount future benefits to today; accept if NPV > 0.

Intangibles

Faster decisions and better service count even if not in rupees.

Rule: recommend the option with acceptable payback, positive NPV and manageable risk.

Case: a one-page executive summary

Problem: Loan decisions take about five days and referred cases are reviewed inconsistently.
Solution: Automate clear approvals and rejections with documented rules; give officers a decision support dashboard for referrals.
Cost: ₹11 lakh one-time, ₹1 lakh a year.
Benefit: ₹5 lakh a year saved; payback in 2.75 years.
Ask: Approve ₹11 lakh and a six-month build.

1. **Problem**: why act now
2. **Solution**: what we will build
3. **Cost**: full and honest
4. **Benefit**: tangible and intangible
5. **Ask**: the decision needed

Illustrative: five sentences, one for each label. A busy executive should be able to decide from this page alone.

Class activity (5 minutes, in pairs)

1

Draw

Context diagram: one process, entities, named flows.

2

Define

Write one data dictionary entry for a key flow.

3

Decide

Write one rule as a decision table or structured English.

4

Support

Name one semi-structured decision a DSS could help with.

Pick one: online food ordering, library lending or hospital appointment booking.

Key takeaways

1. DFDs show how data moves; context diagram first, then level down and balance
2. Data dictionaries define every item so diagrams and systems agree
3. Process specs write down structured decisions: structured English, tables and trees
4. DSS supports semi-structured decisions: machines compute, humans judge
5. The systems proposal turns analysis into a funded decision

Backup: types of DSS

Type	Core support	Example
Data-driven	Query and analyse large data	Sales dashboards, OLAP
Model-driven	Simulation, optimisation, forecasting	Supply-chain optimiser
Knowledge-driven	Rules and expert knowledge	Clinical decision rules
Communication-driven	Group collaboration	Shared planning tools
Document-driven	Search unstructured documents	Policy and contract search

Trend: AI blurs the types: models, data, rules and documents increasingly work together.

Backup: writing the proposal

Written proposal

- Executive summary first
- Evidence in findings and appendices
- Plain language, numbered sections
- Proofread numbers and diagrams

Oral presentation

- Start with the problem and the ask
- One message per slide
- Show the context diagram, not every DFD
- Prepare for cost and risk questions

Audience test: could a busy executive decide from your first page?