Introduction to Knowledge Modeling

Session 1

📚 🧠 🔗

Joseph Azar

CP58 - UTBM Sevenans

What is Knowledge?

> **Knowledge = Information + Context + Relationships**

Simple example:

- **Data:** "Ti-6Al-4V"
- **Information:** "Ti-6Al-4V is a material"
- **Knowledge:** "Ti-6Al-4V is a titanium alloy used in aircraft engines because it has high strength and low weight"

What is Knowledge Modeling?

> **Knowledge Modeling = Organizing information in a structured way**

Just like:

- 📁 Organizing files in folders
- 🗃️ Creating a database for products
- 🗺️ Drawing a map of your factory

But for KNOWLEDGE, not just data!

Why Do We Model Knowledge?

Problem #1: Information Overload

Engineering has TONS of information:

- Thousands of parts in a car
- Hundreds of suppliers
- Millions of possible materials
- Complex manufacturing processes

How do you make sense of it all?

Why Do We Model Knowledge?

Problem #2: Finding Answers is Hard

Questions engineers ask:

- "Which parts use this material?"
- "If this supplier fails, what's affected?"
- "What's the best path for this robot?"
- "Which design satisfies requirement R42?"

Without structure, finding answers takes DAYS!

Why Do We Model Knowledge?

Solution: Knowledge Models Help You

- ✅ **Organize** complex information
- ✅ **Search** quickly
- ✅ **Discover** hidden connections
- ✅ **Answer** complex questions
- ✅ **Share** knowledge with others

Examples in Mechanical Engineering

Let's look at 3 simple examples

⚙️ 🏭 🤖

Example 1: Bill of Materials (BOM)

What is a BOM?

A list of all parts in a product

Example: Simple Engine

- Engine contains → Cylinder Block
- Cylinder Block contains → 4 Cylinders
- Each Cylinder contains → 1 Piston
- Each Piston contains → 3 Rings

Example 1: BOM Answer

Answer: 4 cylinders × 1 piston × 3 rings = **12 rings**

This is KNOWLEDGE:

- Not just data (parts list)
- But **relationships** (what contains what)
- And **structure** (hierarchy)

We need to MODEL this knowledge!

Example 2: Supply Chain

Supply Chain = Network of suppliers providing parts

Example:

- Factory makes Cars
- Supplier A provides Engines
- Supplier B provides Batteries (to Supplier A)
- Supplier C provides Steel (to Factory)

Question: "If Supplier B fails, what's impacted?"

Example 2: Supply Chain Answer

Answer:

- Supplier B fails → Supplier A has no batteries
- Supplier A fails → Factory has no engines
- Factory fails → No cars!

This is KNOWLEDGE:

- Understanding **connections**
- Finding **impact**
- Tracing **dependencies**

Example 3: Robot Path Planning

Problem: Robot needs to move from A to B, avoiding obstacles

Possible paths:

- Path 1: A → C → D → B (distance: 15m)
- Path 2: A → E → B (distance: 10m)
- Path 3: A → F → G → B (distance: 20m, has obstacle!)

Question: "What's the shortest valid path?"

Example 3: Robot Path Answer

Answer: Path 2 (A → E → B) is shortest at 10m

This is KNOWLEDGE:

- Modeling **space** as connections
- Finding **optimal** solutions
- Avoiding **obstacles**

How Do We Model Knowledge?

There are several approaches:

1. Tables (Relational Databases)

Excel, SQL databases

2. Documents

Word files, PDFs, wikis

3. Ontologies

Formal definitions, rules

4. Graphs ✨

Networks, connections

Approach 1: Tables (Relational Databases)

Example: Parts Table

Part ID	Part Name	Parent ID
1	Engine	-
2	Cylinder	1
3	Piston	2

Approach 1: Tables - Pros & Cons

✅ Good For:

- Simple, structured data
- Fast searches on single table
- Well understood, widely used

❌ Bad For:

- Complex relationships (many JOINs)
- Unknown depth ("how many levels?")
- Questions like "find all connections"

Approach 2: Documents

Example: Design Document

"The engine consists of a cylinder block, which contains 4 cylinders. Each cylinder has a piston made of aluminum alloy..."

✅ Good For:

- Human reading
- Detailed explanations
- Flexible content

❌ Bad For:

- Searching
- Automated processing
- Answering questions

Approach 3: Ontologies

Ontology = Formal definitions + Rules

Example:

- "A Piston IS-A MechanicalComponent"
- "MechanicalComponent HAS-A Material"
- "IF component IS-A Piston THEN it REQUIRES a Cylinder"

Very powerful but complex! Used in aerospace, medical fields. Good for reasoning and validation.

Why Graphs for Knowledge Modeling?

🔗 🕸️ 🗺️

Why Graphs?

> **Because most engineering knowledge is about CONNECTIONS!**

Look at our examples:

- BOM: Parts **connected** in hierarchy
- Supply Chain: Suppliers **connected** to each other
- Robot Path: Locations **connected** by paths

Graphs are MADE for modeling connections!

Why Graphs?

✅ Graphs are Good For:

- Modeling relationships
- Finding connections
- Traversing networks
- Path finding
- Impact analysis
- Pattern discovery

🎯 Perfect For:

- Social networks
- Supply chains
- Transportation
- Bill of Materials
- Requirements
- Knowledge!

Introduction to Graph Theory

📊 🔢 🧮

Let's learn the basics!

What is a Graph?

> **Graph = Nodes (circles) + Edges (lines)**

```
A — B — C
|       |
  D — E
```

5 nodes (A, B, C, D, E) connected by lines

Graph Terminology

Node (or Vertex)

The "things" in your graph

Examples: Person, Part, City, Machine

Edge (or Relationship)

The "connections" between things

Examples: knows, contains, supplies, connected-to

Real Example: Social Network

```
Alice —knows→ Bob

↓ knows          ↓ knows

Charlie ←knows— David
```

Nodes: Alice, Bob, Charlie, David (people)

Edges: "knows" relationships

Real Example: BOM as Graph

```
Engine
↓ contains
Cylinder Block
↓ contains (4x)
Cylinder
↓ contains
Piston
```

Nodes: Engine, Cylinder Block, Cylinder, Piston

Edges: "contains" relationships

Undirected Graphs

Edges have NO direction (two-way)

Alice ——— Bob

"Alice knows Bob" = "Bob knows Alice"

Examples: Friendship, road connections, co-workers

Directed Graphs

Edges have direction (one-way arrow)

Alice ——→ Bob

"Alice follows Bob" ≠ "Bob follows Alice"

Examples: Twitter follows, hierarchy, BOM

When to Use Which?

Use Undirected When:

- Friendship (mutual)
- Physical connections (roads)
- Co-occurrence

Use Directed When:

- Hierarchy (manager → employee)
- Flow (supplier → factory)
- Dependency (A requires B)
- BOM (assembly contains part)

Weighted Graphs

> **Weighted Graph = Edges have VALUES (weights)**

```
Paris —— 450km —— Lyon
  |                 |
400km             300km
  |                 |
```

Marseille —— 320km —— Nice

Weight = Distance between cities

What Can Weights Represent?

Distance

City A —5km— City B

Capacity

Pipe A —100L/s— Pipe B

Cost

Part A —$50— Part B

Strength

Person A —80%— Person B

Time

Task A —2hrs— Task B

Quantity

Assembly —4— Bolts

What Can We DO With Graphs?

🔍 🛤️ 🎯

Common Graph Operations

Operation 1: Graph Traversal

Traversal = Visiting all connected nodes

Example Question:

"Starting from Engine, what are ALL the parts it contains?"

Answer: Visit Engine, then visit all parts it contains, then visit all parts THEY contain, and so on...

Traversal Example: BOM

Start: Engine

↓ traverse

Cylinder Block → Cylinder (×4) → Piston (×4) → Ring (×12)

Result: Complete list of all parts!

Engine contains: Block, 4 Cylinders, 4 Pistons, 12 Rings

Operation 2: Shortest Path

Shortest Path = Find the route with minimum cost

Example Question:

"What's the fastest route from Paris to Nice?"

Algorithm: Dijkstra's Algorithm

Shortest Path Example

Map:

```
Paris —450km— Lyon —320km— Nice
  |                          |
———————— 600km ————————
```

Path 1: Paris → Lyon → Nice = 450 + 320 = **770km**

Path 2: Paris → Nice = **600km** ✅ SHORTEST!

Shortest Path Applications

Navigation

GPS finding fastest route

Logistics

Cheapest shipping route

Robotics

Optimal robot path

Networks

Data packet routing

Used EVERYWHERE in engineering!

Operation 3: Connectivity Analysis

> **Connectivity = Are two nodes connected?**

Example Questions:

- "Can we reach Factory B from Factory A?"
- "Is this network fully connected?"
- "Which parts are isolated?"

Connectivity Example: Supply Chain

Supplier A → Factory 1 → Customer X

Supplier B → Factory 2 → Customer Y

(No connection between the two chains!)

Finding: Two separate components!

Supplier A's problems won't affect Customer Y ✅

Operation 4: Cycle Detection

Cycle = Path that returns to starting point

A → B → C → D → A

(Back to A! This is a cycle)

Cycles are GOOD for:

Closed mechanical loops (4-bar linkage)

Cycles are BAD for:

Dependency chains (circular dependencies!)

Operation 5: Finding Important Nodes

> **Centrality = How "important" is a node?**

Question: "Which supplier is most critical?"

Answer: The one connected to the most factories!

> **Use Case:** Risk analysis

- Which parts are single points of failure?
- Which processes are bottlenecks?
- Which nodes should we monitor?

Real Problems Graphs Solve

💡 🔧 ✅

Problem 1: Bill of Materials

❌ Without Graphs:

Complex SQL queries, slow for deep hierarchies

✅ With Graphs:

- **Query:** "What parts are in Engine?"
- **Operation:** Simple traversal
- **Speed:** Instant, any depth!

Performance: 100-1000× faster than SQL for deep BOMs!

Problem 2: Supply Chain Impact

❌ Without Graphs:

Manual tracing through spreadsheets, takes days

✅ With Graphs:

- **Query:** "If Supplier X fails, what's affected?"
- **Operation:** Traverse forward from Supplier X
- **Result:** Complete impact list in seconds

Real Example: During COVID chip shortage, companies with graph databases responded faster!

Problem 3: Robot Path Planning

❌ Without Graphs:

Trial and error, inefficient paths

✅ With Graphs:

- **Model:** Space as graph (nodes = positions)
- **Algorithm:** A* (Dijkstra with heuristic)
- **Result:** Optimal path avoiding obstacles

Used in: Warehouse robots, self-driving cars, drones

Problem 4: Requirements Traceability

The Problem:

"If Requirement R42 changes, what tests are affected?"

❌ Without Graphs:

Excel traceability matrix → unmanageable, quickly outdated

Requirements Traceability with Graphs

✅ With Graphs:

```
Requirement R42

↓ SATISFIED_BY

Design D7

↓ IMPLEMENTED_BY

Component C3

↓ VERIFIED_BY

Test T5, Test T9
```

Query: Traverse from R42 → Answer in milliseconds!

Summary: What We Learned

1. Knowledge Modeling

Organizing information in structured ways

2. Why Graphs?

Perfect for modeling CONNECTIONS and RELATIONSHIPS

3. Graph Basics

Nodes, Edges, Directed, Weighted

Summary: Graph Operations

1. Traversal: Visit all connected nodes

2. Shortest Path: Find optimal route (Dijkstra)

3. Connectivity: Are nodes connected?

4. Cycles: Detect circular paths

5. Centrality: Find important nodes

Summary: Real Applications

Bill of Materials

Fast hierarchical queries

Robot Path

Optimal navigation

Supply Chain

Impact analysis

Requirements

Traceability tracking

Graphs are EVERYWHERE in engineering!

Questions?

❓ 💬 🙋

Thank you!

joseph.azar@univ-fcomte.fr