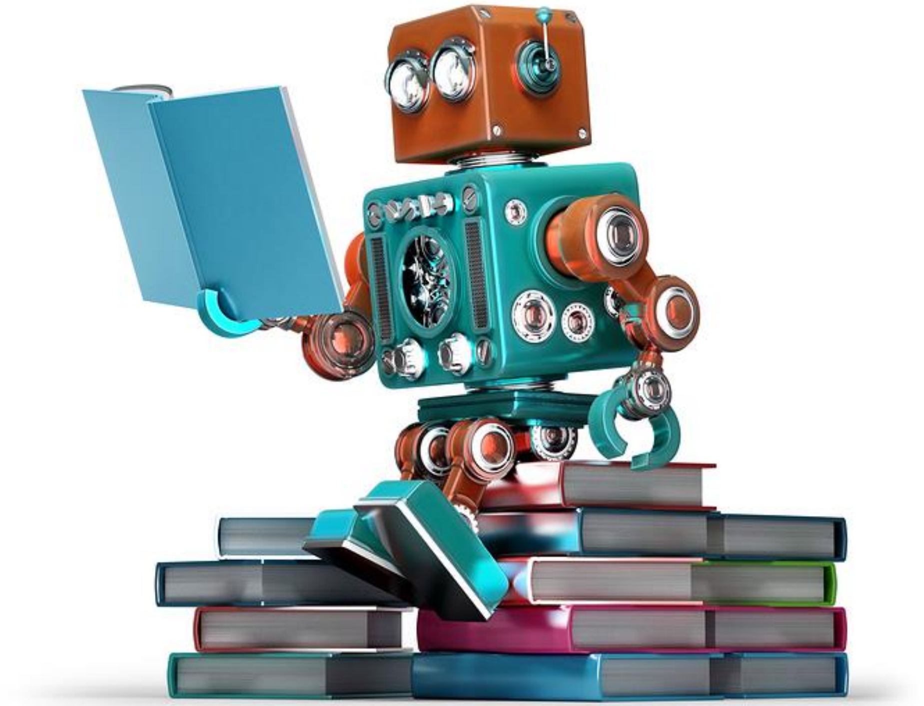
MACHINE REASONING

DAY 2



https://robohub.org/wp-content/uploads/2016/11/bigstock-Retro-Robot-Reading-A-Book-Is-110707406.jpg

DAY 2 AGENDA

2.1 Knowledge Representation

2.2 Knowledge Acquisition (Business Rules)

2.3 Knowledge Models (Acquired → Represented)

Format Acquired Knowledge into Representation Templates

2.4 Knowledge Modelling **Workshop**

DAY 2 TIMETABLE

No	Time	Topic	By Whom	Where
1	9 am	2.1 Knowledge Representation	GU Zhan (Sam)	Class
2	10.10 am	Morning Break		
3	10.30 am	2.2 Knowledge Acquisition 2.3 Knowledge Models	GU Zhan (Sam)	Class
4	12.10 pm	Lunch Break		
5	1.30 pm	2.4 Knowledge Modelling Workshop Tutorial	GU Zhan (Sam) All	Class
6	3.10 pm	Afternoon Break		
7	3.30 pm	2.4 Knowledge Modelling Workshop	All	Class
8	4.50 pm	Summary and Review	All	Class
9	5 pm	End		

2.1
KNOWLEDGE REPRESENTATION

2.1 KNOWLEDGE REPRESENTATION

- **A scheme by which to represent knowledge in a machine/computer, in a way that allows the computer system to use or manipulate it to solve difficult problems**
  - Unlike humans, knowledge must be 'transplanted/saved' into machine reasoning system/memory
  - Representation goes hand-in-hand with reasoning/inference mechanism (computer algorithm)
  - Large amount of knowledge is usually needed to solve complex problems

Data Structure

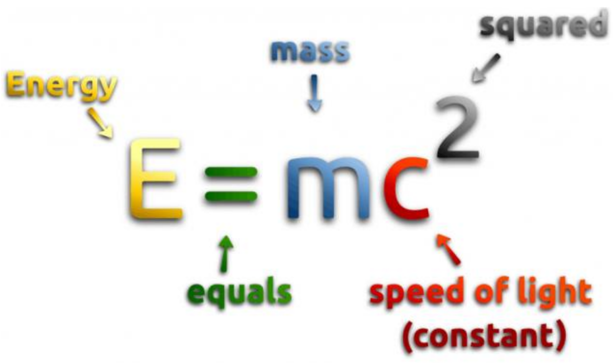
Processing Logic

- **Explicit knowledge representation (think of documentation) enables business knowledge management and retention.**

2.1 KNOWLEDGE REPRESENTATION

Forms of Knowledge Representation

- Natural Language
- Formula
- Formal Logic
- Semantic Web
- Frames
- Ontology
- Knowledge Graph
- Database
- Rules
- And many other forms…



And

p	q	$p \cdot q$
T	T	T
T	F	F
F	T	F
F	F	F

Or

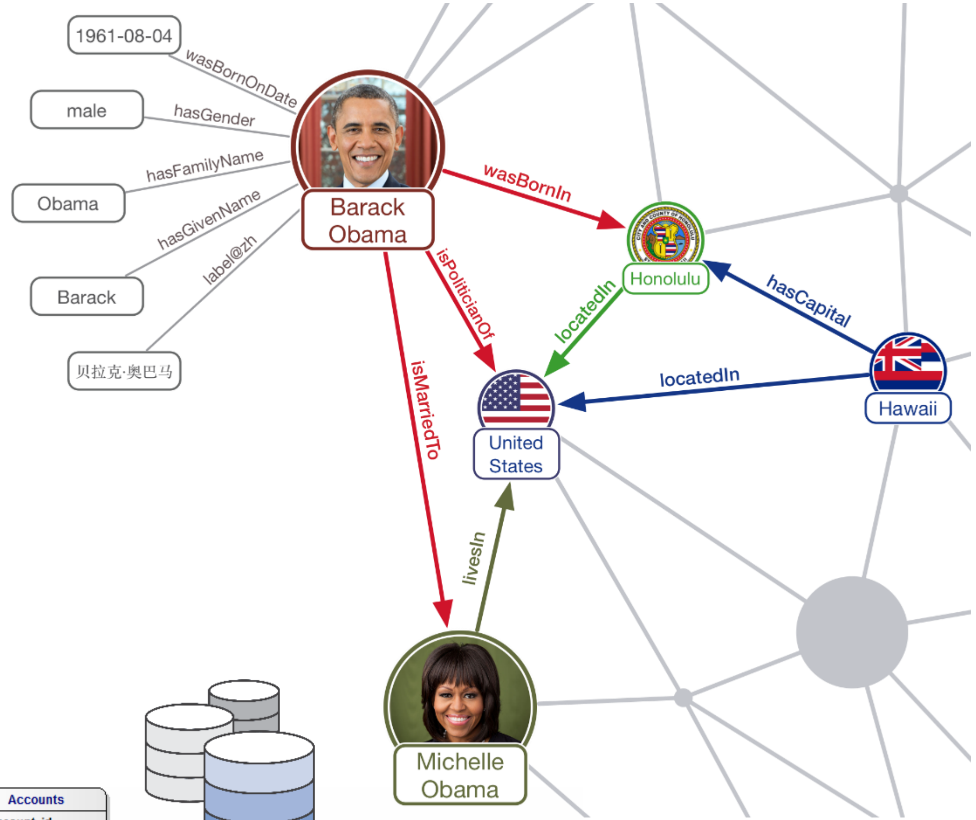
p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

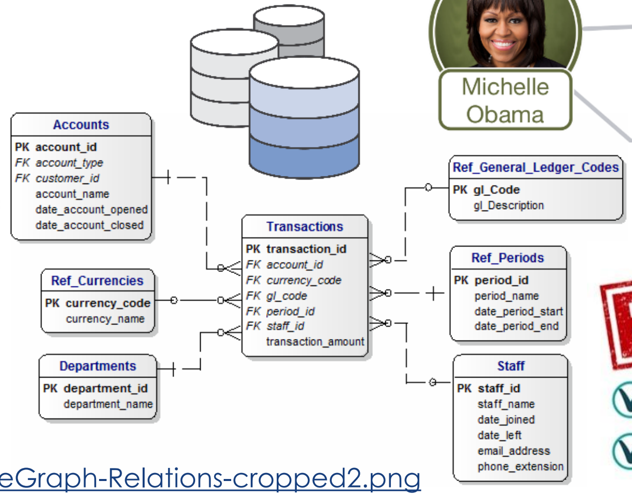
If . . . then

p	q	$p \supset q$
T	T	T
T	F	F
F	T	T
F	F	T

Not

p	$\sim p$
T	F
F	T





https://www.ambiverse.com/wp-content/uploads/2017/01/KnowledgeGraph-Relations-cropped2.png

2.1 KNOWLEDGE REPRESENTATION

Formal Logic – Propositional Logic

- **Propositional Logic**
  - Examples of propositions: propositional sentence
    - ***p*** = "Sam has flu."
    - ***q*** = "Sam has cough."

And

p	q	$p \cdot q$
T	T	T
T	F	F
F	T	F
F	F	F

Or

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

If . . . then

p	q	$p \supset q$
T	T	T
T	F	F
F	T	T
F	F	T

Not

p	$\sim p$
T	F
F	T

  - What about sentence: ***s*** = "x + y = 5", where x and y are variables?

    ☺ Not a proposition, as its truth cannot be defined unless x and y are assigned specific values

2.1 KNOWLEDGE REPRESENTATION

Formal Logic – Propositional Logic

- **The syntax of propositional logic expression is constructed using propositions and connectives**
  - All propositions must be either **true** or **false** (referred to as **truth value** of the proposition)
- **Connectives**
  - $\neg$ negation "not"
  - $\vee$ disjunction "or"
  - $\wedge$ conjunction "and"
  - $\rightarrow$ implication "if … then"
  - $\leftrightarrow$ bi-conditional "if and only if"
- **Complex expressions using connectives**
  - $p \wedge q$ = "Sam has flu. **AND** Sam has cough."
  - $p \wedge \neg q$ = "Sam has flu. **AND** Sam has no cough."
  - $p \rightarrow q$ = "**IF** Sam has flu **THEN** Sam has cough."

2.1 KNOWLEDGE REPRESENTATION

Formal Logic – First Order Logic

- **First Order Logic (Predicate Calculus)**
  - Propositional logic assumes the world contains: **facts**
  - First order logic (also called first-order predicate calculus, or predicate logic) assumes the world contains:
    - **Objects (Class)** : people, houses, numbers, colors, baseball games, …
    - **Constants (Instance)** : The White House, Sam GU Zhan, π, NUS,...
    - **Variables** : x, y, a, b, ...
    - **Relations (Predicate)** : is student, has leg, eats, is bigger than, is part of, is red, is round, prime to, come between, is one more than, …
    - **Functions (Predicate)** : father of, best friend, square root of, sum of, …
    - **Connectives** : $\neg \rightarrow \wedge \vee \leftrightarrow$
    - **Equality** : =
    - **Quantifiers** : $\forall \exists$

2.1 KNOWLEDGE REPRESENTATION

Formal Logic – First Order Logic

- **Sentences of First Order Logic**
  - **Term** (noun) is an expression that refers to an object.
    - FatherOf(DiDi) : "father of DiDi"
    - ¬ FatherOf(DiDi) : "not father of DiDi"
    - FatherOf(x) : "someone's father"
    - Sam, Jessie, DiDi, Machine Reasoning Course, PhD, …
    - Integer: x, y, z (variables of object: all integer numbers)
  - **Atomic Sentence** (semantics) is formed from **one** predicate symbol followed by **one** parenthesized list of terms
    - IsFriend(Jessie, Sam) : Jessie is friend to Sam.
    - IsFriend(Jessie, FatherOf(DiDi)) : Jessie is friend to DiDi's father.

relational predicate

functional predicate

2.1 KNOWLEDGE REPRESENTATION

Formal Logic – First Order Logic

And

p	q	$p \cdot q$
T	T	T
T	F	F
F	T	F
F	F	F

If . . . then

p'	q'	$p' \supset q'$
T	T	T
T	F	F
F	T	T
F	F	T

- **Sentences of First Order Logic**
  - **Complex Sentence** (semantics) is made from **Atomic Sentences** using logical connectives
    - IsClassmate(Jessie, Mary) ∧ IsFriend(Jessie, Sam) → IsFriend(Mary, Sam)

p' q'

- **Establish Truth of Predicate**
  - Predicates need to be propositionalized for use in reasoning
  - **Method 1**: Assign specific value to predicate expressions (similar to instantiation)
    - StudyAt(x, NUS) → Smart(x) What's the scope of x?
    - x = Sam
    - Conclusion: StudyAt (Sam, NUS) → **Smart(Sam)**

2.1 KNOWLEDGE REPRESENTATION

Formal Logic – First Order Logic

- **Establish Truth of Predicate**
  - **Method 2A**: Universal quantifier: ∀

    We want to express "Everyone studying at NUS is smart."

    √ ∀x StudyAt(x, NUS) → Smart(x) : "For everyone, if the person is studying at NUS then the (**same**) person is smart" (For those are not studying at NUS, we don't know.)

    × ∀x StudyAt(x, NUS) ∧ Smart(x) : "Everyone (all persons in Singapore) is studying at NUS and all (**these**) persons are smart." **incorrect semantic**
  - **Method 2B**: Existential quantifier: ∃

    We want to express "Someone studying at NUS is smart."

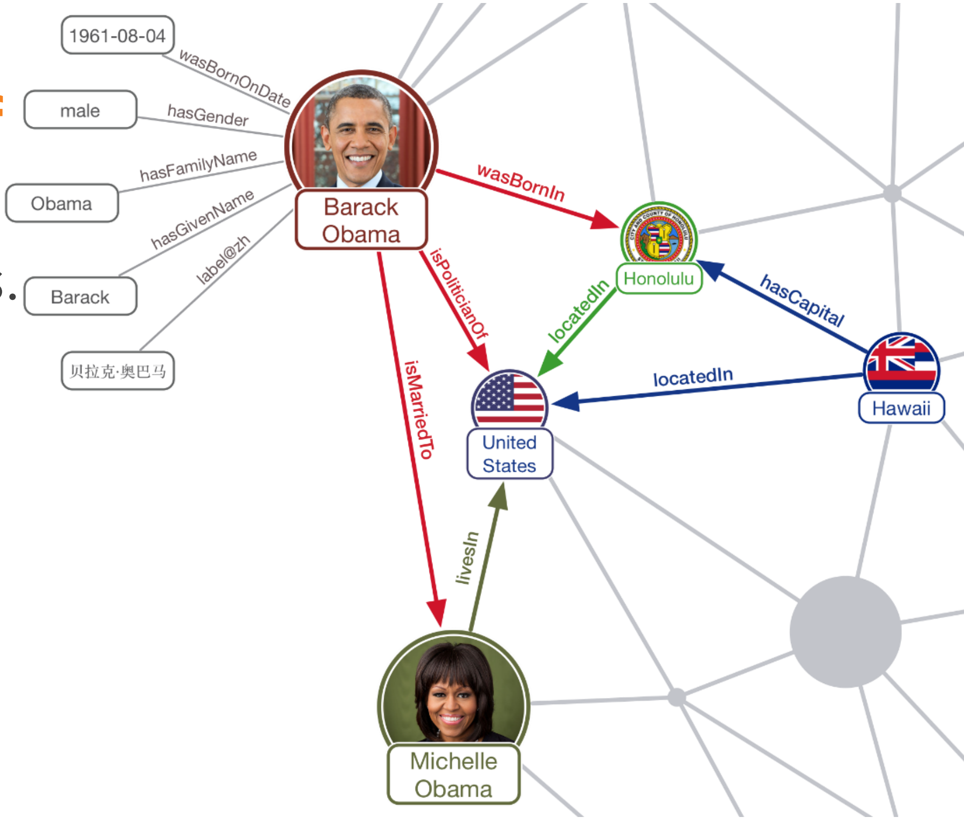
    √ ∃x StudyAt(x, NUS) ∧ Smart(x) : "There is someone studying at NUS and this (**same**) person is smart."

    × ∃x StudyAt(x, NUS) → Smart(x) : "There is someone, when he/she is studying at NUS then he/she is smart." (When this (**same**) person is having a rest, then he/she may not be smart.) **incorrect semantic**

2.1 KNOWLEDGE REPRESENTATION

Semantic Web, Knowledge Graph

- **Semantic web is a model for word concepts in human cognition, consisting of nodes, links and labels**
  - **Nodes** represent objects, concepts, or situations. They can be instances (individual objects as in Knowledge Graph) or classes (generic objects as in Semantic Web)
  - **Links** between nodes represent a relationship
  - **Labels**
    - Labels on nodes indicate the name of the object, concept, etc.
    - Labels on links describe the type of relationship between nodes



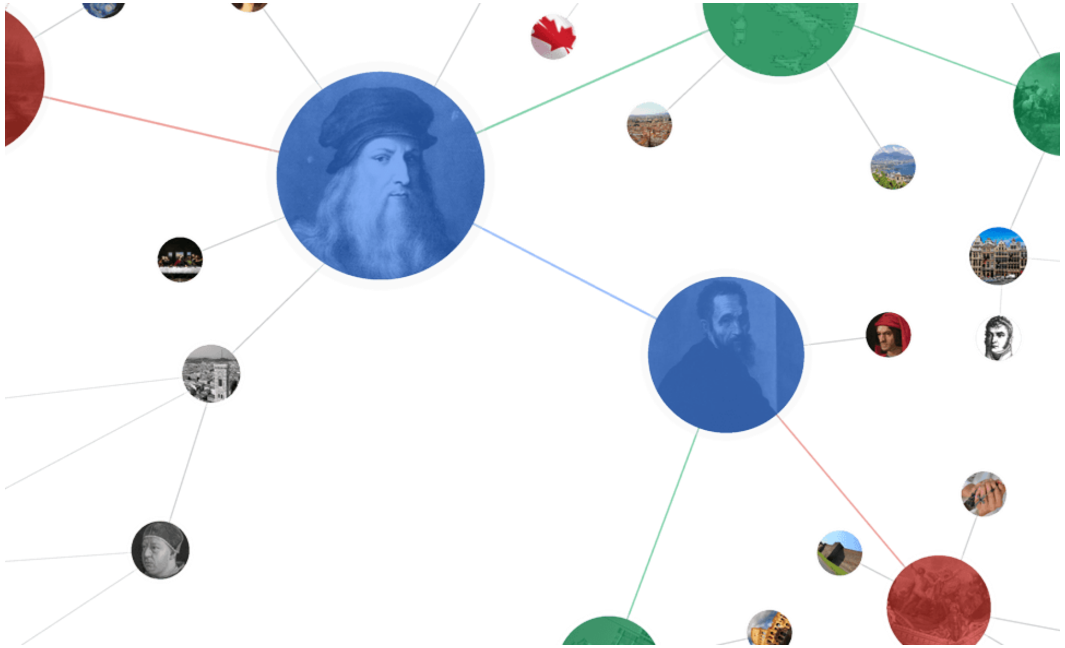
https://www.ambiverse.com/wp-content/uploads/2017/01/KnowledgeGraph-Relations-cropped2.png

- **Reasoning question: What's the relationship between Barack and Michelle Obama?**

2.1 KNOWLEDGE REPRESENTATION

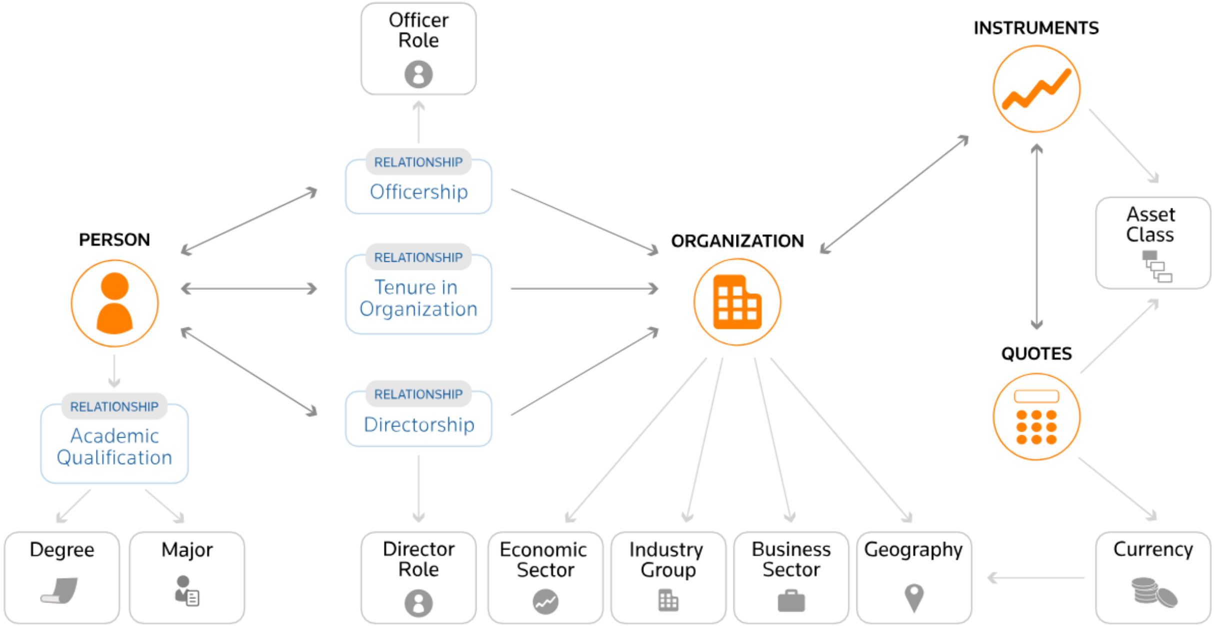
Semantic Web, Knowledge Graph

- Google Knowledge Graph



https://www.tampa-seo.com/wp-content/uploads/static-graph.png

- Thomson Reuters Knowledge Graph product: Perm ID

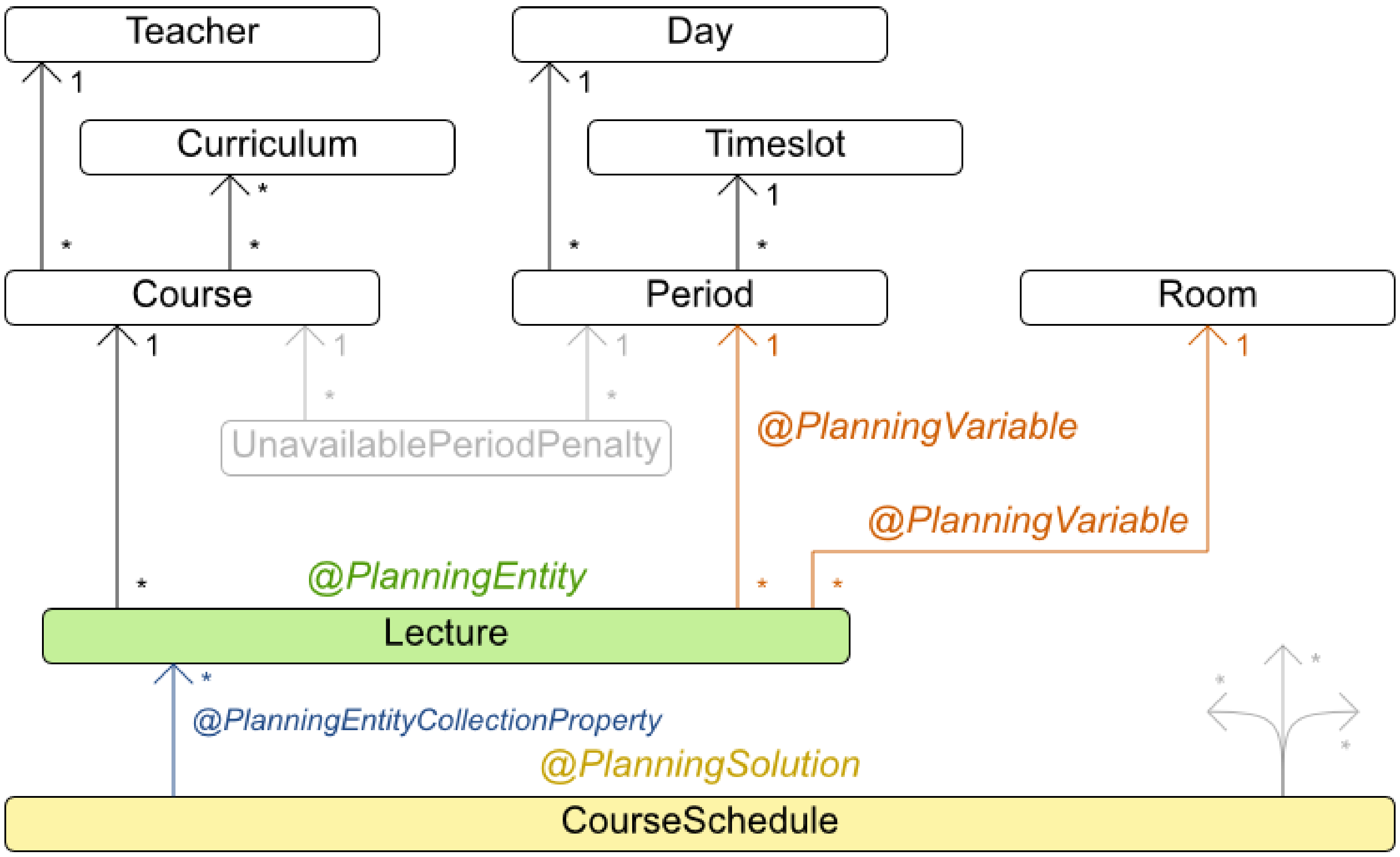


https://permid.org/

2.1 KNOWLEDGE REPRESENTATION

Domain Ontology / OO Classes / DB Schema

Curriculum course class diagram



2.1 KNOWLEDGE REPRESENTATION

Rules

- **Represent problem-solving knowledge as**

  “**IF/WHEN** … **THEN** …” rules

  - Is the classic technique for representing domain knowledge in an machine reasoning system
  - Is also a very natural way of human decision making

- **A rule consists of two parts:**
  - The **IF** part
    - called the **antecedent** or **premise** or **condition**
  - The **THEN** part
    - called the **consequent** or **conclusion** or **action**

2.1 KNOWLEDGE REPRESENTATION
Rules

- **Basic Rule Syntax**

  IF <antecedent>
  THEN <consequent>

  IF person X is ill
  THEN person X need rest a lot

2.1 KNOWLEDGE REPRESENTATION

Rules

- **Multi-antecedent Rule**

IF	<antecedent 1>
AND/OR	<antecedent 2>
	…
AND/OR	<antecedent n>
THEN	<consequent>

IF	person X is ill
AND	person X is a lecturer
THEN	person X cannot rest at home, but go to class

2.1 KNOWLEDGE REPRESENTATION

Rules

- **Multi-consequent Rule**

IF <antecedent 1>
THEN <consequent 1>
<consequent 2>
…
<consequent m>

☺ The relationship between the multiple consequents is understood as AND, depending on the implementation of software for developing reasoning systems.

2.1 KNOWLEDGE REPRESENTATION

Rules

- **Example rules in application**

IF ‘age of the customer’ < 18
AND ‘cash withdrawal’ > $1,000
THEN ‘signature of the parent’ is required

IF ‘taxable income’ > $16,238
THEN ‘Medicare levy’ = ‘taxable income’ * 1.5 %

2.1 KNOWLEDGE REPRESENTATION
Rules

- **Rule Types Example:**

  Rules can represent relations, recommendations/directives and heuristics

  - **Relation**

    IF The 'fuel tank' is empty
    THEN The engine will not start

  - **Recommendation**

    IF The season is autumn
    AND The sky is cloudy
    AND The forecast is drizzle
    THEN The advice is 'take an umbrella'

  - **Heuristic**

    IF PIE is jammed
    THEN Switch to AYE (or ask for working from home)

2.1 KNOWLEDGE REPRESENTATION

Rules

- **Rules in Rule/Process Reasoning System are considered relatively independent**

  Each rule represents a single chunk of knowledge

  - **IF** pet_size = medium **THEN** recommend = cats or small dogs

- **Rules are based on a priori knowledge or heuristics**

  Rules are derived from domain experts who uses experiential knowledge and "rules-of-thumb"

  - **IF** buyer = female **THEN** recommend = hamster

- **Rules can incorporate uncertainties**

  Real life business situations are plagued with uncertainties that make decision-making difficult (or flexible)

  - **IF** work = long_hours **THEN** recommend = dog (30% confidence in rule conclusion)

2.1 KNOWLEDGE REPRESENTATION

Exercise 2.1

- **Convert the following knowledge about animals into WHEN/THEN rules:**
  - animals with hair as their body covering are mammals
  - animals that feed their young with milk are mammals
  - animals with feathers as their body covering are birds
  - animals that fly and reproduce by eggs are birds
  - mammals that eat meat are carnivores
  - mammals with pointed teeth, claws on their feet, and eyes that point forward are carnivores
  - mammals that eat grass are herbivores
  - mammals with hooves on their feet are herbivores
  - carnivores that have a tawny colour and dark spots as their marking are cheetahs
  - carnivores that have a tawny colour and dark stripes as their marking are tigers
  - herbivores that have a tawny colour and dark spots as their marking and long necks are giraffes
  - herbivores that have a black and white colour are zebras
  - birds that walk and are black and white and have a long neck are ostriches
  - birds that swim and are black and white are penguins
  - birds that fly and are black and white are albatrosses

2.1 KNOWLEDGE REPRESENTATION

Rules – KIE Drools

a driving license application.

```
public class Applicant {
    private String name;
    private int age;
    private boolean valid;
    // getter and setter methods here
}
```

Now that we have our data model we can write our first rule. We assume that the application uses rules to reject invalid applications. As this is a simple validation use case we will add a single rule to disqualify any applicant younger than 18.

```
package com.company.license

rule "Is of valid age"
when
    $a : Applicant( age < 18 )
then
    $a.setValid( false );
end
```

To make the Drools engine aware of data, so it can be processed against the rules, we have to ***insert*** the data, much like with a database. When the Applicant instance is inserted into the Drools engine it is evaluated against the constraints of the rules, in this case just two constraints for one rule. We say ***two*** because the type **Applicant** is the first object type constraint, and **age < 18** is the second field constraint. An object type constraint plus its zero or more field constraints is referred to as a pattern. When an inserted instance satisfies both the object type constraint and all the field constraints, it is said to be matched. The **$a** is a binding variable which permits us to reference the matched object in the consequence. There its properties can be updated. The dollar character ('$') is optional, but it helps to differentiate variable names from field names. The process of matching patterns against the inserted data is, not surprisingly, often referred to as ***pattern matching***.

Source https://docs.jboss.org/drools/release/latest/drools-docs/html_single/index.html#_methods_versus_rules

4.1.3. Methods versus Rules

People often confuse methods and rules, and new rule users often ask, "How do I call a rule?" After the last section, you are now feeling like a rule expert and the answer to that is obvious, but let's summarize the differences nonetheless.

```
public void helloWorld(Person person) {
    if ( person.getName().equals( "Chuck" ) ) {
        System.out.println( "Hello Chuck" );
    }
}
```

- Methods are called directly.
- Specific instances are passed.
- One call results in a single execution.

```
rule "Hello World" when
    Person( name == "Chuck" )
then
    System.out.println( "Hello Chuck" );
end
```

- Rules execute by matching against any data as long it is inserted into the Drools engine.
- Rules can never be called directly.
- Specific instances cannot be passed to a rule.
- Depending on the matches, a rule may fire once or several times, or not at all.

Source https://docs.jboss.org/drools/release/latest/drools-docs/html_single/index.html#_methods_versus_rules

4.1.4. Cross Products

Earlier the term "cross product" was mentioned, which is the result of a join. Imagine for a moment that the data from the fire alarm example were used in combination with the following rule where there are no field constraints:

```
rule "Show Sprinklers" when
    $room : Room()
    $sprinkler : Sprinkler()
then
    System.out.println( "room:" + $room.getName() +
                        " sprinkler:" + $sprinkler.getRoom().getName() );
end
```

In SQL terms this would be like doing `select * from Room, Sprinkler` and every row in the Room table would be joined with every row in the Sprinkler table resulting in the following

```
room:office sprinkler:office
room:office sprinkler:kitchen
room:office sprinkler:livingroom
room:office sprinkler:bedroom
room:kitchen sprinkler:office
room:kitchen sprinkler:kitchen
room:kitchen sprinkler:livingroom
room:kitchen sprinkler:bedroom
room:livingroom sprinkler:office
room:livingroom sprinkler:kitchen
room:livingroom sprinkler:livingroom
room:livingroom sprinkler:bedroom
room:bedroom sprinkler:office
room:bedroom sprinkler:kitchen
room:bedroom sprinkler:livingroom
room:bedroom sprinkler:bedroom
```

These cross products can obviously become huge, and they may very well contain spurious data. The size of cross products is often the source of performance problems for new rule authors. From this it can be seen that it's always desirable to constrain the cross products, which is done with the variable constraint.

```
rule
when
    $room : Room()
    $sprinkler : Sprinkler( room == $room )
then
    System.out.println( "room:" + $room.getName() +
                        " sprinkler:" + $sprinkler.getRoom().getName() );
end
```

This results in just four rows of data, with the correct Sprinkler for each Room. In SQL (actually HQL) the corresponding query would be `select * from Room, Sprinkler where Room == Sprinkler.room`.

```
room:office sprinkler:office
room:kitchen sprinkler:kitchen
room:livingroom sprinkler:livingroom
room:bedroom sprinkler:bedroom
```

Source https://docs.jboss.org/drools/release/latest/drools-docs/html_single/index.html#_cross_products

2.1 KNOWLEDGE REPRESENTATION

Rules – KIE Drools

- KIE Drools rule is declarative language. It's functional similar to structured query language SQL.
- In logical reasoning context, When/Then rules (in knowledge base) are considered universally true, thus can be 'declared'.

CashFlow Rule

```
select * from Account acc,
     Cashflow cf, AccountPeriod ap
where acc.accountNo == cf.accountNo and
      cf.type == CREDIT
      cf.date >= ap.start and
      cf.date <= ap.end
acc.balance += cf.amount
```

```
rule "increase balance for AccountPeriod Credits"
    when
        ap  : AccountPeriod()
        acc : Account()
        cf  : CashFlow( type == CREDIT,
                        accountNo == acc.accountNo,
                        date >= ap.start && <= ap.end )
    then
        acc.balance  += cf.amount;
end
```

Two rules can be used to determine the debit and credit for that quarter and update the Account balance. The two rules below constrain the cashflows for an account for a given time period. Notice the "&&" which use short cut syntax to avoid repeating the field name twice.

```
rule "increase balance for credits"
when
  ap : AccountPeriod()
  acc : Account( $accountNo : accountNo )
  CashFlow( type == CREDIT,
            accountNo == $accountNo,
            date >= ap.start && <= ap.end,
            $amount : amount )
then
  acc.balance  += $amount;
end
```

```
rule "decrease balance for debits"
when
  ap : AccountPeriod()
  acc : Account( $accountNo : accountNo )
  CashFlow( type == DEBIT,
            accountNo == $accountNo,
            date >= ap.start && <= ap.end,
            $amount : amount )
then
  acc.balance -= $amount;
end
```

Earlier we showed how rules would equate to SQL, which can often help people with an SQL background to understand rules. The two rules above can be represented with two views and a trigger for each view, as below:

```
select * from Account acc,
              Cashflow cf,
              AccountPeriod ap
where acc.accountNo == cf.accountNo and
      cf.type == CREDIT and
      cf.date >= ap.start and
      cf.date <= ap.end
```

```
trigger : acc.balance += cf.amount
```

```
select * from Account acc,
              Cashflow cf,
              AccountPeriod ap
where acc.accountNo == cf.accountNo and
      cf.type == DEBIT and
      cf.date >= ap.start and
      cf.date <= ap.end
```

```
trigger : acc.balance -= cf.amount
```

Source https://docs.jboss.org/drools/release/latest/drools-docs/html_single/index.html#_cashflow_example

KIE IDE Menu iss-admin

Spaces » ISS-MR » Mortgage_Process_ISS_MR » master »

MortgageMachineReasoning

Project Explorer

<default> » com » myspace » MortgageMachineReasoning.rc

BUSINESS PROCESSES

DATA OBJECTS

FORMS

GUIDED DECISION TABLES

GUIDED RULES

OTHERS

Mortgage... Save Delete Rename Copy Validate Download Latest Version View Alerts

Model Overview Source Data Objects

```
package com.myspace.mortgage_app;

import java.lang.Number;

rule "MortgageMachineReasoning"
    dialect "mvel"
    ruleflow-group "mortgagemachinereasoning"
    when
        app : Application( mortgageamount >= ( app.property.saleprice - app.downpayment ) )
    then
        app.setInlimitMR( true );
end
```

Create Guided Rule: MortgageMachineReasoning.rdrl

To check whether:

mortgage amount >= property sale price - down payment

2.2
KNOWLEDGE ACQUISITION (BUSINESS RULES)

2.2 KNOWLEDGE ACQUISITION (BUSINESS RULES)

- **Knowledge Acquisition is the transfer and transformation of problem solving knowledge into a form that can be used to build intelligent systems.**
- **Knowledge acquisition is also called:**
  - Knowledge capture
  - Knowledge elicitation
  - Requirements engineering
- **Personnel involved:**
  - Knowledge holder, e.g. subject matter expert (SME); process owner
  - Knowledge engineer, e.g. business analyst; system analyst; consultant

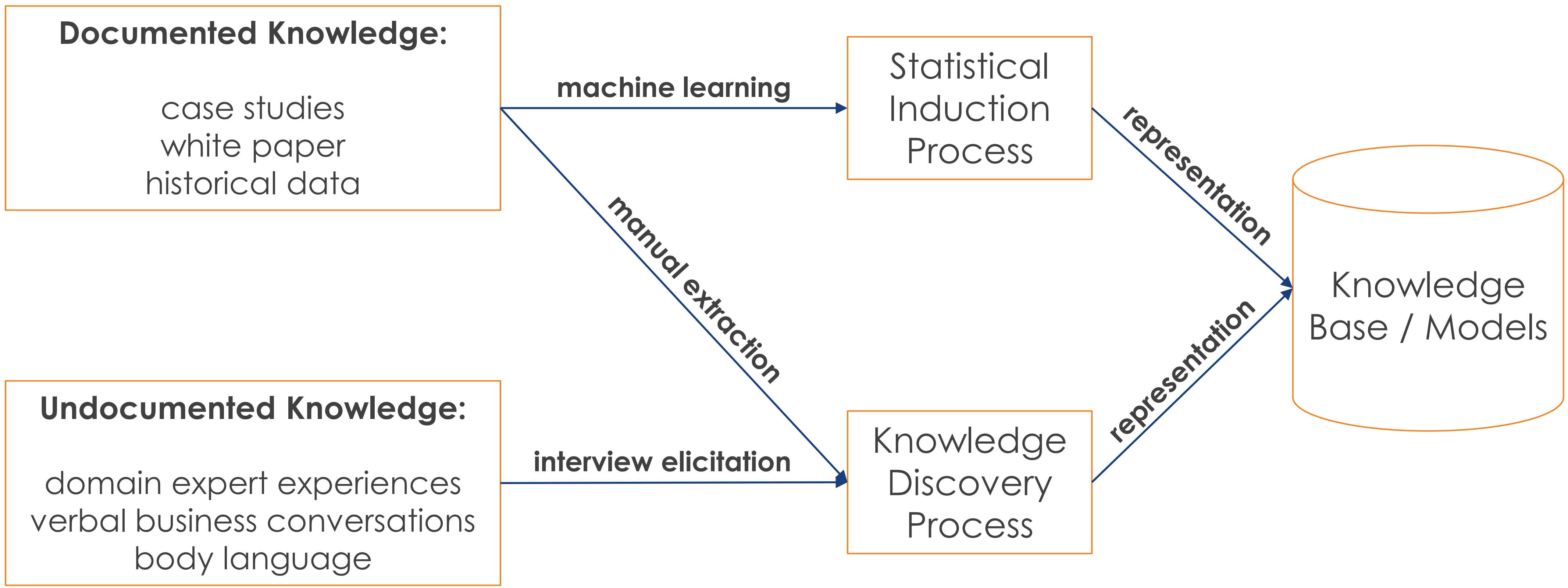
2.2 KNOWLEDGE ACQUISITION (BUSINESS RULES)

Knowledge Sources

- **Two types of knowledge can be used to build KBS/RBS:**
  - **Documented sources**
    - Collection from printed sources
    - Machine learning (rule induction, decision tree, neural network, deep learning, etc.)
  - **Undocumented sources**
    - Tacit knowledge that can only be captured by elicitation from human experts

2.2 KNOWLEDGE ACQUISITION (BUSINESS RULES)

Acquisition Methods



2.2 KNOWLEDGE ACQUISITION (BUSINESS RULES)

Interview Elicitation

- **Elicitation is acquisition of tacit knowledge from a subject matter expert**
- **The Knowledge elicitation approach:**
  - Capture knowledge using interviews
  - Interpret & Analyze the transcripts and data obtained
  - Build knowledge models (knowledge representation)
  - Use the knowledge models to guide further elicitation
  - Verify & Validate the captured knowledge
  - Stop when the knowledge model is enough for building business reasoning system

2.2 KNOWLEDGE ACQUISITION (BUSINESS RULES)

Interview Best Practices

- **More Beneficial interviewing expert at their workplace**
- **Make sure the meeting place is quiet and free from interruptions**
- **Before the interview:**
  - Do background research on the domain area
  - Background check on the domain expert
  - Design and phrase your questions
  - Email questions to domain expert
  - Acquire and prepare the tools for the interview
- **During the interview:**
  - Introductions & Social preliminaries
  - State purpose of interview
  - Give a brief on the roles and responsibilities
  - Be courteous; Listen closely; Avoid arguments
  - Investigate each topic in detail
  - Evaluate session outcome
- **Observe confidentiality**

2.2 KNOWLEDGE ACQUISITION (BUSINESS RULES)

Exercise 2.2

Exercise

- Analyse Airport Gate Assignment System (AGAS) interview transcript.
- Identify three or more missing information which require further interview or investigation.

2.3

KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

- **After acquiring domain knowledge from the experts and other sources, how do we present a comprehensive view of this knowledge?**
- **Knowledge Models (Templates for knowledge representation)**
  - A knowledge model is a group of **structured representations** of knowledge that allows us to better understand the domain and the processes involved in decision making.
  - Documented models provide rich descriptions of domain knowledge that is **independent** of any particular software implementation.
  - These models also serve as a basis for communication among stakeholders: experts, analysts, developers, and end users.

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Document Templates

• Concept Dictionary	
• Concept Tree	
• Composition Tree	
• Decision Tree	
• Goal Reduction Tree	
• Inference Diagram	KIE Rule Flow Groups: Task (Activity/Sub-Goal) level
• Attribute Worksheet	KIE Data Model: Object; Field; Type
• HMI/UI System-User Dialogue	KIE Form: Task level
• Rules & Decision Table	KIE Guided Rules; Decision Table
• Dependency Diagram	KIE Rule Flow Groups: Rule level
• Flowchart (Workflow)	KIE Process Flow: Task level for Business Functions
• Activity Flow Diagram	KIE Process Flow: Task level for Business Teams/Roles
• RACI Matrix	KIE Business Teams/Departments/Roles/Groups

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

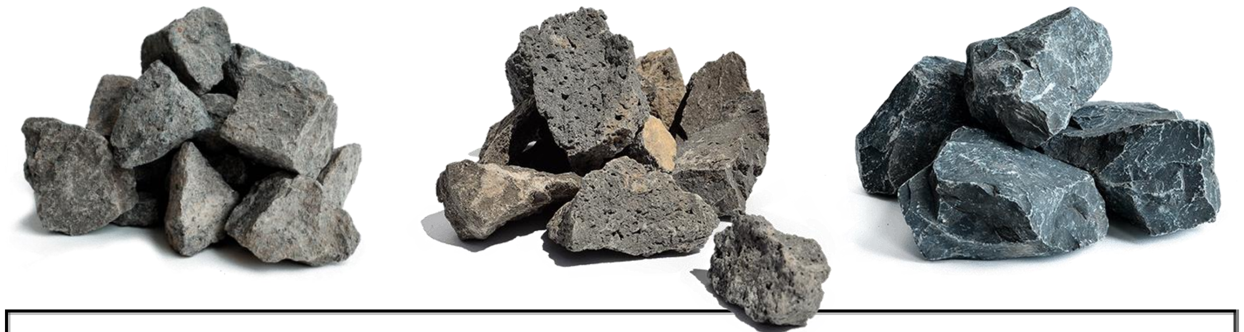
Concept Dictionary

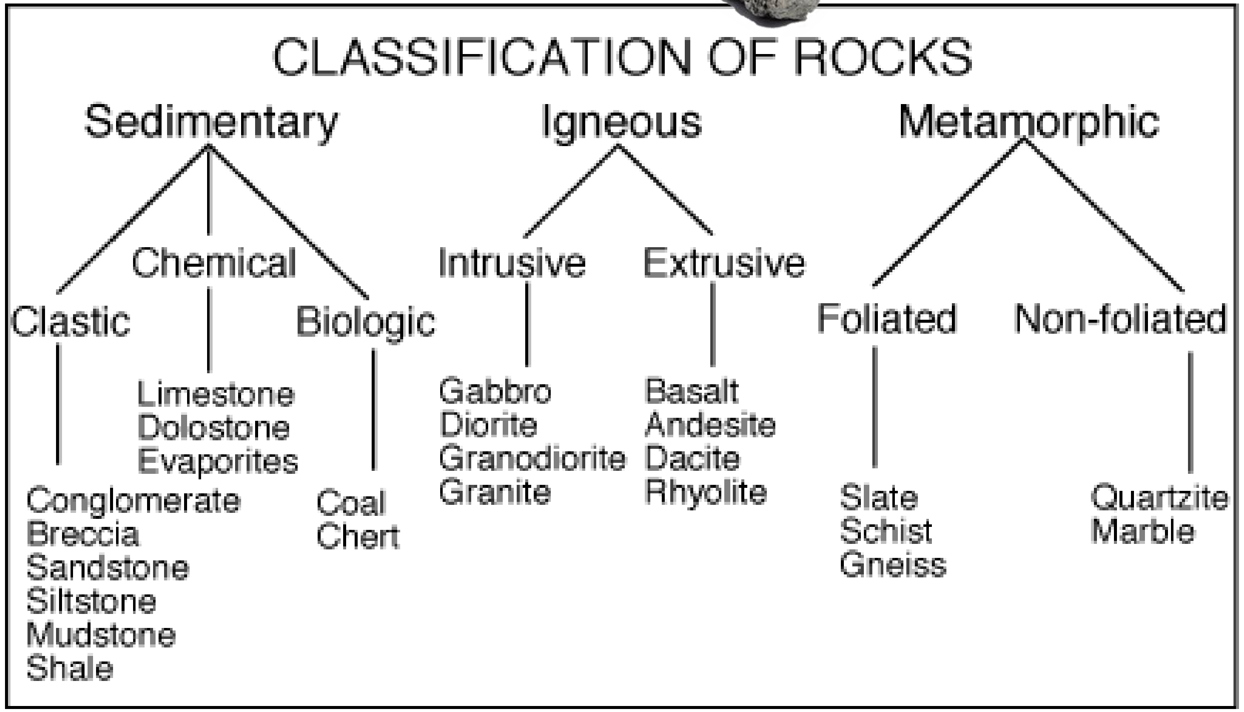
Concept	Synonyms Abbr	Meaning
Esophagus	Gullet Oesophagus	Sometimes known as the gullet. Muscular tube through which food passes from pharynx to the stomach
Duodenum		The first section of the small intestine
Peptic ulcer	PUD	Area of the gastrointestinal tract that is extremely painful. Mucosal erosions equal to or greater than 0.5cm.
Hyperacidity	Acid dyspepsia, Amalpitta	A condition of excreting more than the normal amount of hydrochloric acid in the stomach

- A concept dictionary contains the list of **all relevant concepts** that are used in the problem domain to solve the problem.
- The dictionary provides a detailed explanation of the concept and can include any information that is useful for a good understanding of the concept.
- It can be similar to a **glossary**.
- It does not have any particular format.

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Concept Tree

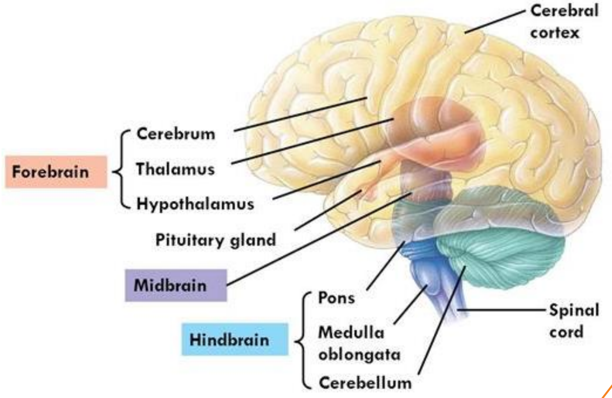




- Tree that shows concepts and the classes and sub-classes.
- All relationships must be "is a".
- Check the tree by looking at the lowest and highest nodes and asking "is <sub-concept> a type of <concept>".
- Nodes should have clear & complete names.
- Captured terminology and landscape of the domain.

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Composition Tree



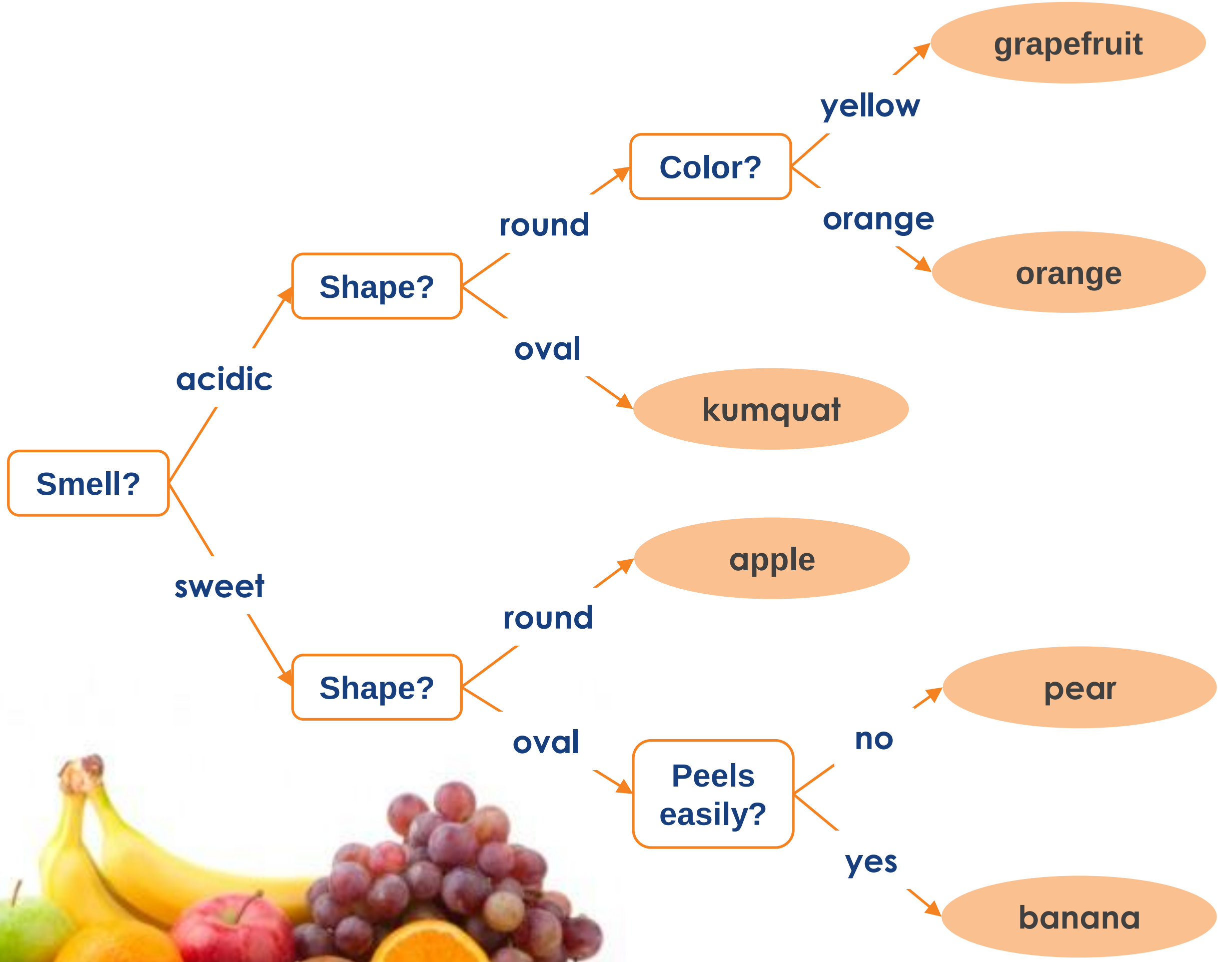
- Brain
  - Forebrain
    - Telencephalon
      - Cerebral cortex
      - Basal ganglia
      - Limbic system
    - Diencephalon
      - Thalamus
      - Hypothalamus
  - Midbrain
    - Mesencephalon
      - Tectum
      - Tegmentum
  - Hindbrain
    - Metencephalon
      - Cerebellum
      - Pons
    - Meyelencephalon
      - Medulla oblogata

- **Detailed concept breakdowns into its constituent parts.**
- **All relationships must be "part of".**
- **Understand things as**
  - Products (parts of a machinery)
  - Organisations (your organisation chart)
  - Documents (the table of contents)

http://blog.askiitians.com/wp-content/uploads/2016/05/Brain-1.jpg

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

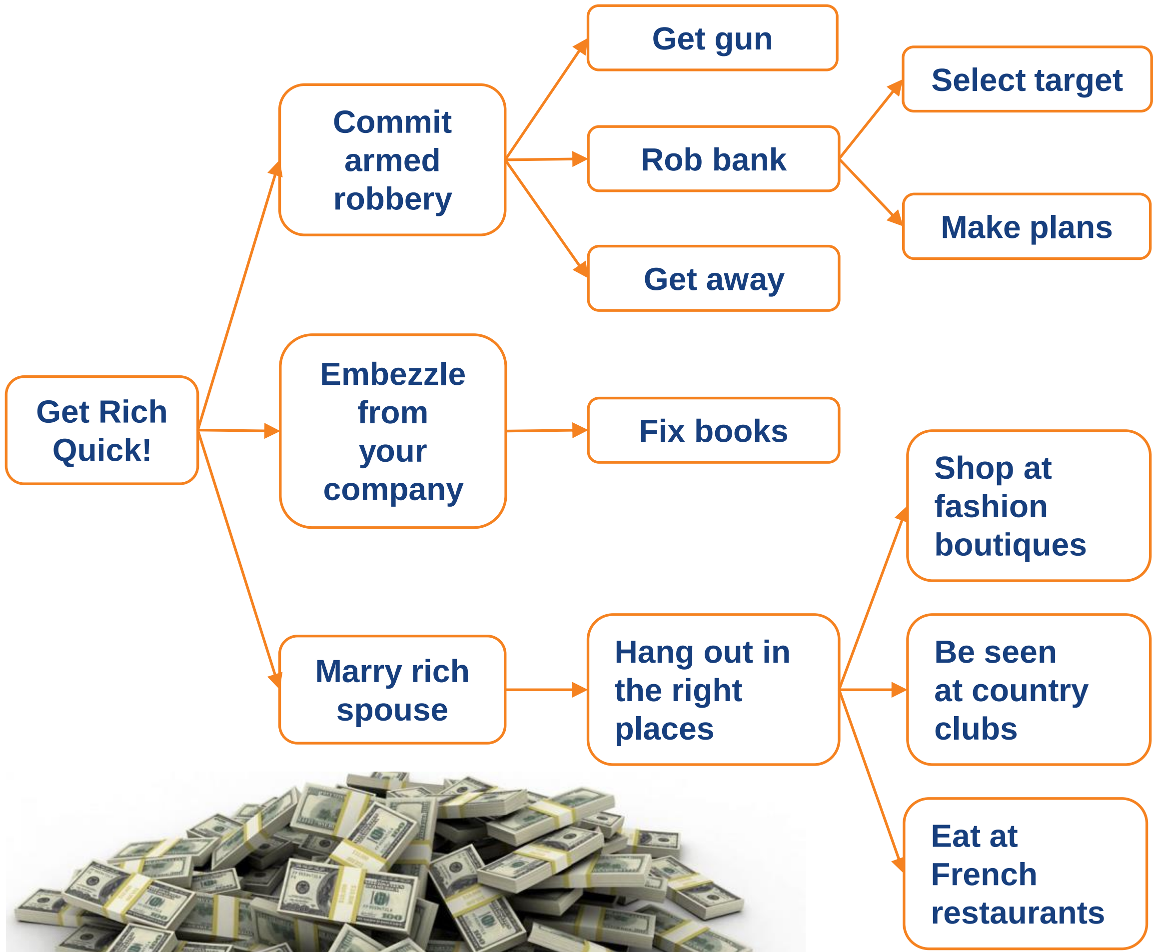
Decision Tree



- Tree shows the alternative courses of action or casual consequences for a particular decision.
- Condition/Rule based domain knowledge
- A snapshot of the experts knowhow

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Goal Reduction Tree



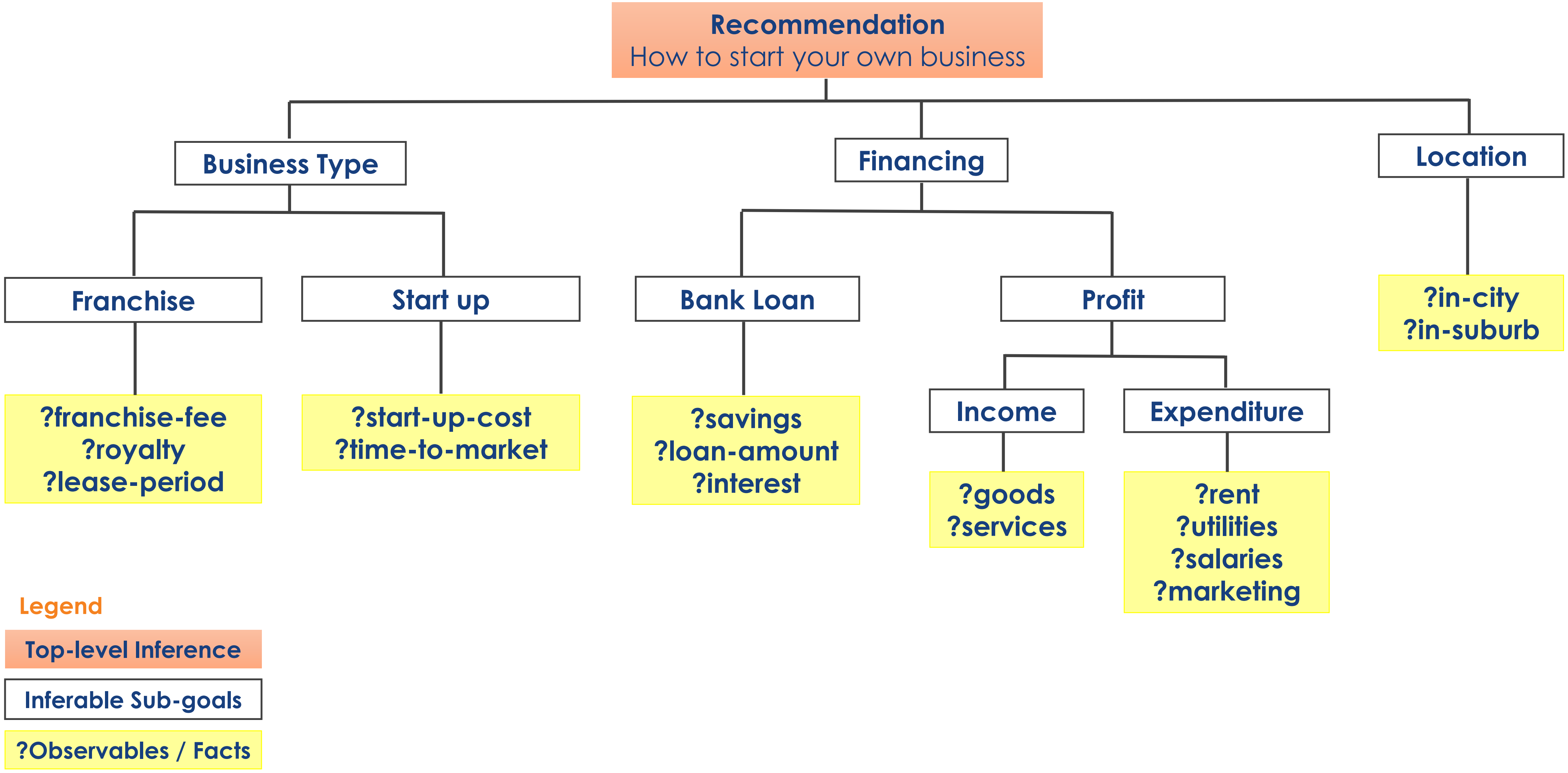
- Indicate relations between goal & sub-goals
- Incorporates AND/OR links (OR is implicit)
- Typical patterns of problem-solution behavior can be analyzed

Goal Reduction Tree vs. Decision Tree

- **Goal Reduction Trees are**
  - More “action” oriented while the Decision Tree is more static and causal.
  - Useful for planning a overall strategy “sub-goals” for problem solving
- **Decision Trees are**
  - More detailed conditional rule sets
  - Have “labeled links” where every node has a specific question/condition to check

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Inference Diagram KIE Rule Flow Group; Task



2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Inference Diagram KIE Rule Flow Group; Task

- A Inference Diagram is a type of goal reduction tree, extended with detailed observable/controllable **input factors**.
- A Inference Diagram consists of problem-solving **factors** arranged in a **hierarchical** tree structure.
- The top-level (tree root) node is the final decision goal.
- The intermediate nodes are the sub-factors or **sub-goals**.
- The bottom-level (tree leaves) nodes are the **input factors**.
- An Inference Diagram shows the different level of inferences in the decision process.

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Attribute Worksheet KIE Data Model: Object; Field; Type

Sub-goal	Attribute	Inferable or Observable	KIE Field Type & Value			English Translation
KIE Data Model: Data Object	**KIE Data Model: Object Field**	**KIE Form: User Interface**	**String, Integer, Float, Boolean, Date, etc.**	**Value Range**	**Value Unit**	**KIE Data Model: Comments**
Franchise	franchise-fee	Observable	Float	1 - 50,000	SGD $	The price to be paid for the franchise
	royalty	Observable	Float	1 - 10,000	SGD $	The monthly fee payable to franchisor
	lease-period	Observable	Integer	1 - 5	years	The Franchise Lease period
Profit	**Income**	**Inferable**	Float	1 – 1,000,000	SGD $	Annual income from sale of goods
	Expenditure	Inferable	Float	1 – 1,000,000	SGD $	Annual expenditure from sale of goods
Income	goods	Observable	Float	1 – 1,000,000	SGD $	The sales proceeds from goods sold
	services	Observable	Float	1 – 1,000,000	SGD $	Sales proceeds from services rendered
Location	in-city	Observable	Boolean	True or False	N.A.	Planned shop in city area
	in-suburb	Observable	Boolean	True or False	N.A.	Planned shop in suburb area

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

UI System-User Dialogue KIE Form

Sub-goal	Franchise
System Questions / KIE Forms	**Example User Response**
How much is the franchise Fee?	$10,000
How much is the monthly royalty fee?	$2,000
How long is the lease period?	3 years

- Compose User Interface (UI) System-User Dialogues
- Define **questions** that will be asked by the reasoning system; Or **forms** to be filled by end user.
- Define responses that the user is expected to give
- Use System-User Dialogue tables

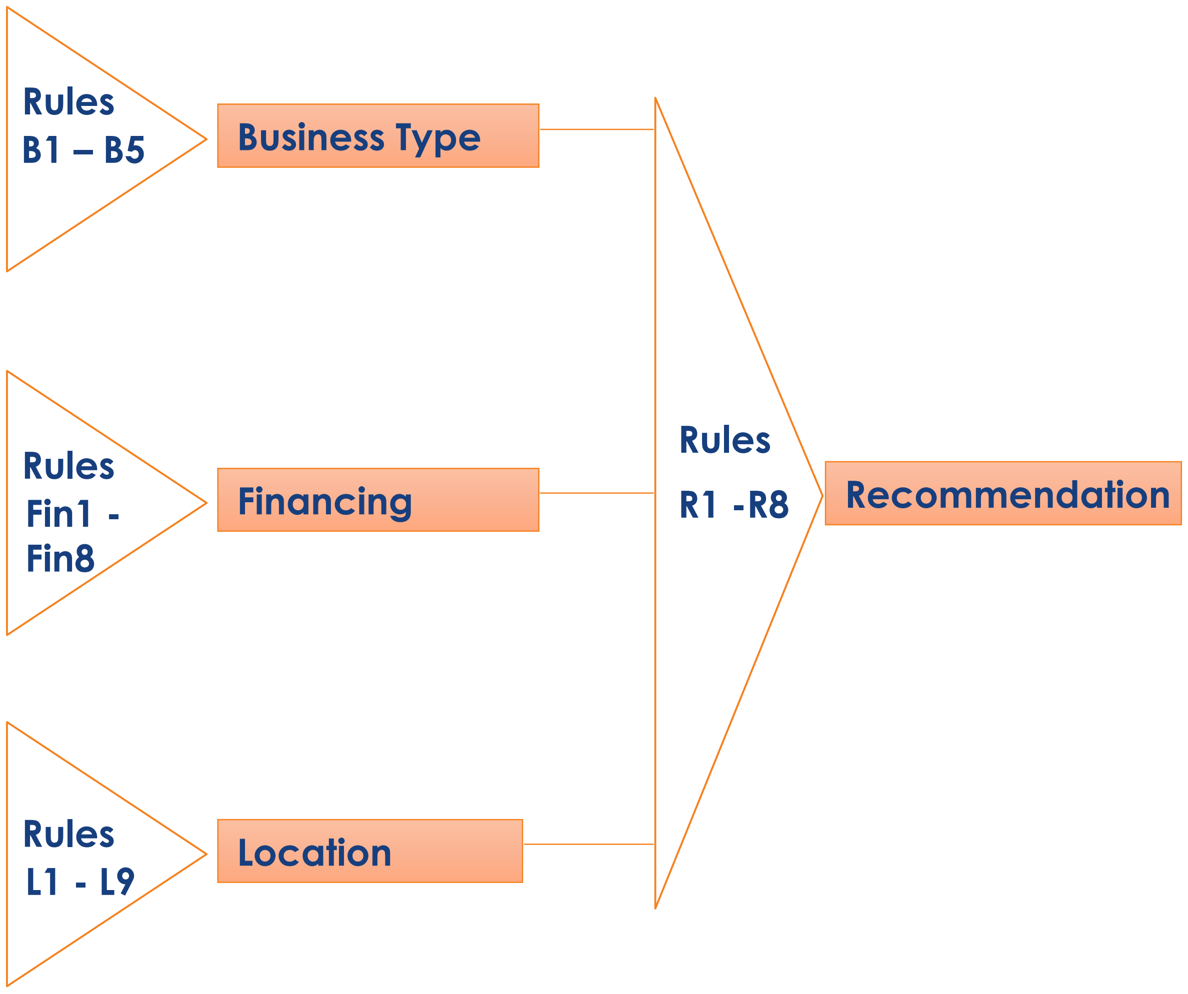
2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Rules & Decision Table KIE Guided Rules; Decision Table

Rule No.	Condition 1	Logical Operand	Condition 2	Sub-goal
F-1	franchise-fee ≤ threshold1	AND	royalty ≤ threshold2	Franchise = ok
F-2	franchise-fee ≤ threshold1	AND	royalty > 20% x franchise-fee	Franchise = not-ok
F-?	…	…	…	…

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

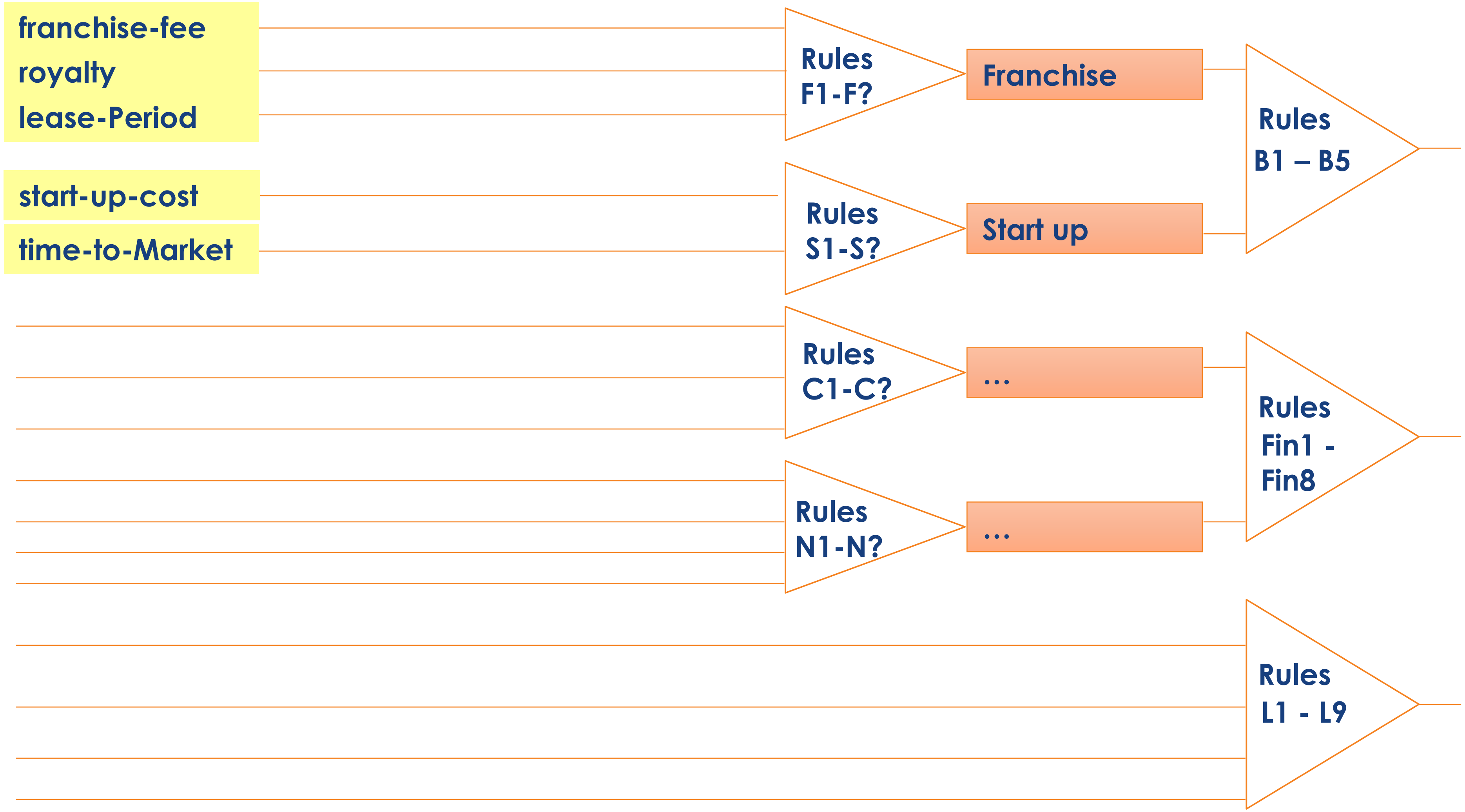
Dependency Diagram   KIE Rule Flow Groups



- Dependency Diagram captures **sub-goals** with corresponding **rules**.
- It also shows the relationship between the different sub-goals (rule groups).
- To be used to construct rule flow groups based on sub-goals, in KIE Task.

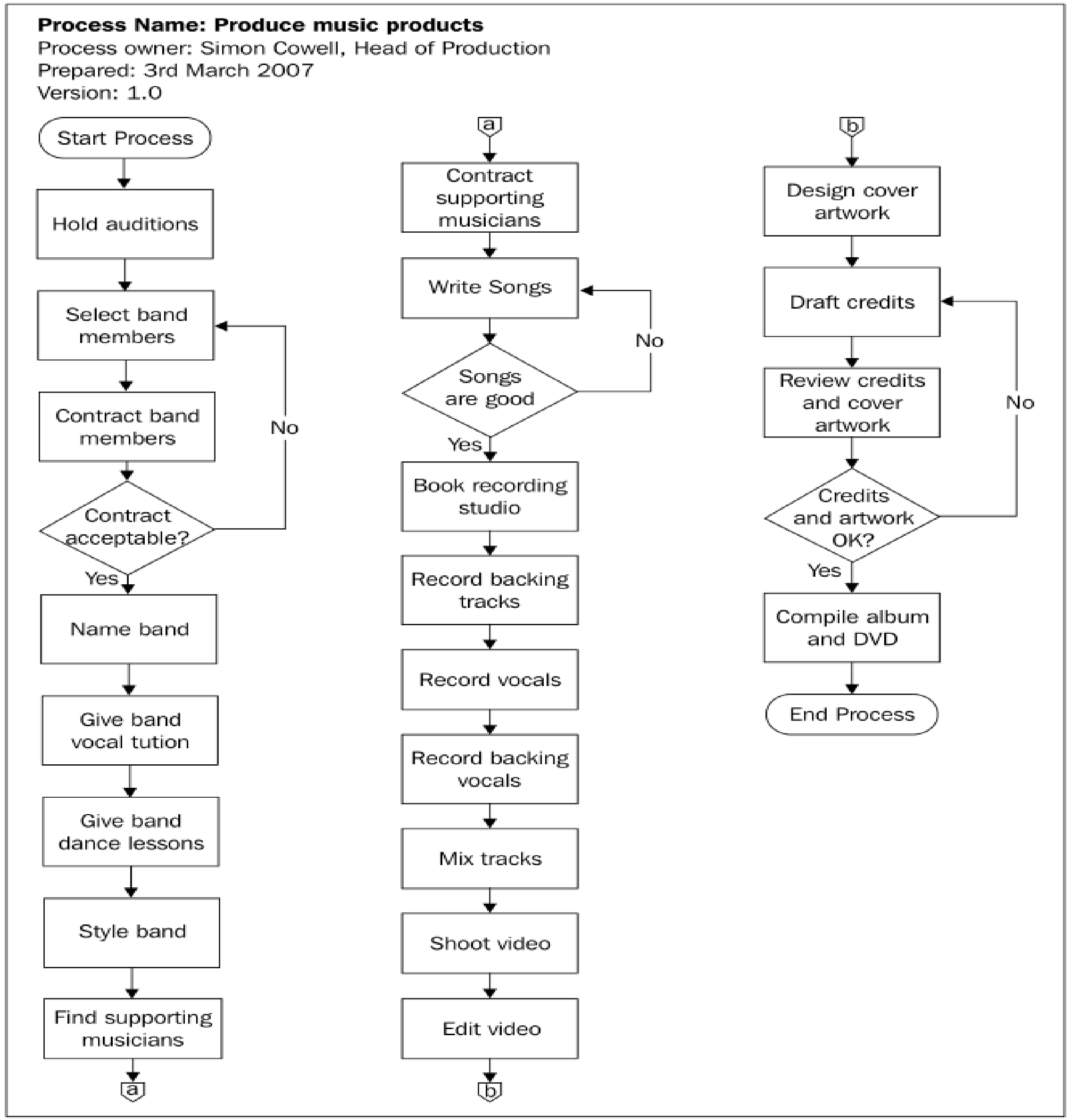
2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Dependency Diagram KIE Rule Flow Groups



2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Flowchart KIE Process Flow: Task level for Business Functions

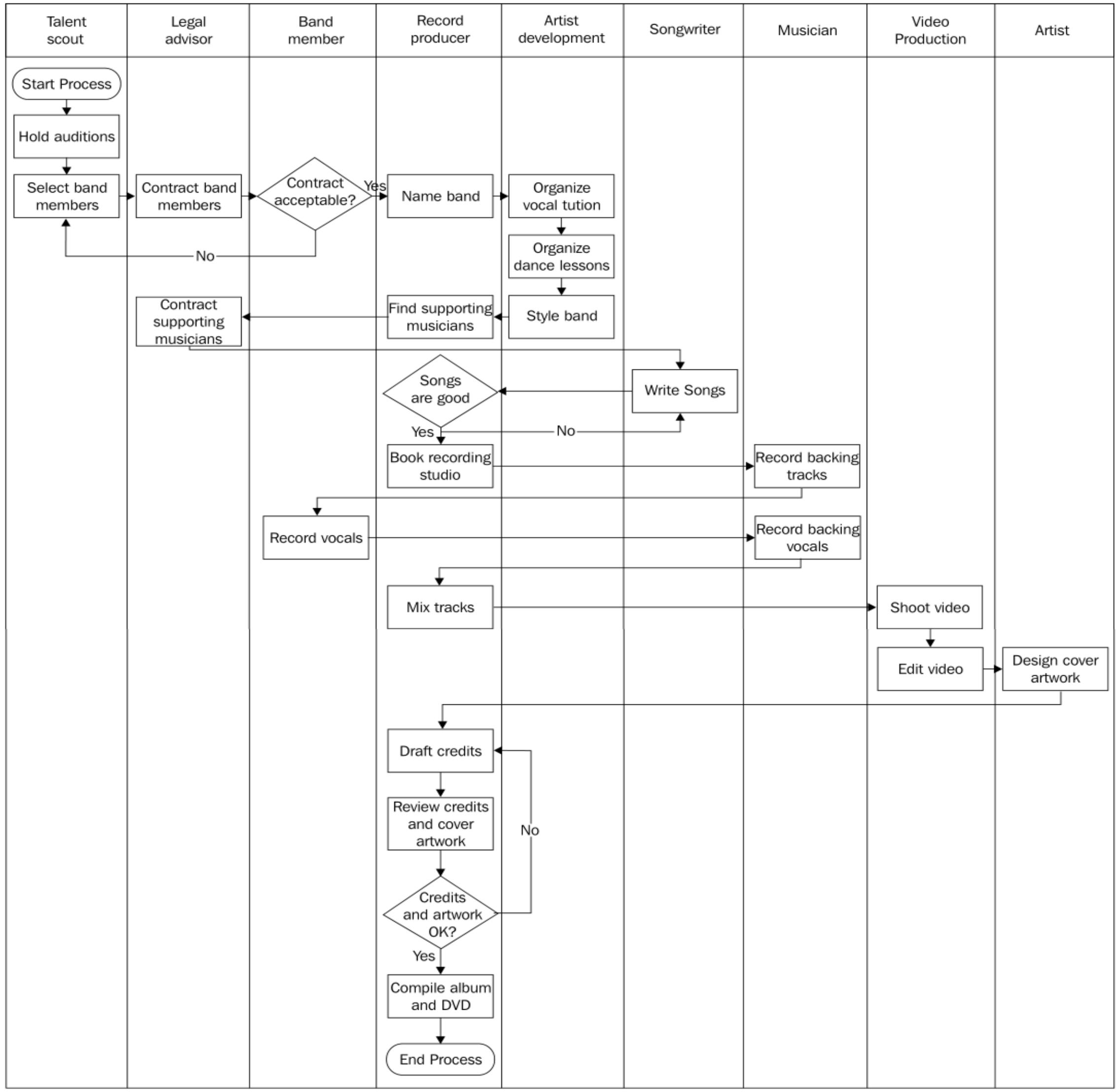


- The key to develop Flowchart to model/capture business process workflow, is to define the **sequence of activities**, and to identify those points where the flow can go two ways, depending on the circumstances.
- Write the activity name in as few words as possible, e.g. **Verb + Noun** pairs
- Write **decision points** as clear questions to which the answer is either "yes/true" or "no/false".

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Activity Flow Diagram KIE Process Flow: Task level for Business Teams/Roles

Process Name: Produce music products
Process owner: Simon Cowell, Head of Production
Prepared: 3rd March 2007
Version: 1.0



- Activity Flow Diagram captures “who does what (activity)” in the workflow.
- Identify roles and responsibilities (Swimlanes)
- A single activity should map to a single role. If this doesn't seem possible, then consider whether the activity should actually be split out into multiple activities.
- Expand flowchart by drawing swimlanes for each activity under the identified roles/teams.

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

RACI Matrix KIE Business Teams/Departments/Roles/Groups

Role \ Process step	Hold auditions	Select band members	Contract band members	Name band	Organise vocal tuition	Organise dance lessons	Stylise band	Find supporting musicians	Contract supporting musicians	Write songs	Book recording studio	Record backing vocals	Record vocals	Record backing vocals	Mix tracks	Shoot video	Edit video	Design cover artwork	Draft credits	Review credits and cover artwork	Compile album and DVD
Note	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Talent Scout	AR	AR	R					C													
Legal Advisor			AR						AR											R	
Band Member				I	I	I	I				I		R			R				R	
Record Producer			C	AR			C	AR	R	C	AR	AR	AR	AR	AR	R	C	R	AR	R	AR
Artist Development				C	AR	AR	AR			C								C		R	
Songwriter										AR										R	
Musician											I	R		R						R	
Video Production																AR	AR			R	R
Artist																		AR		R	

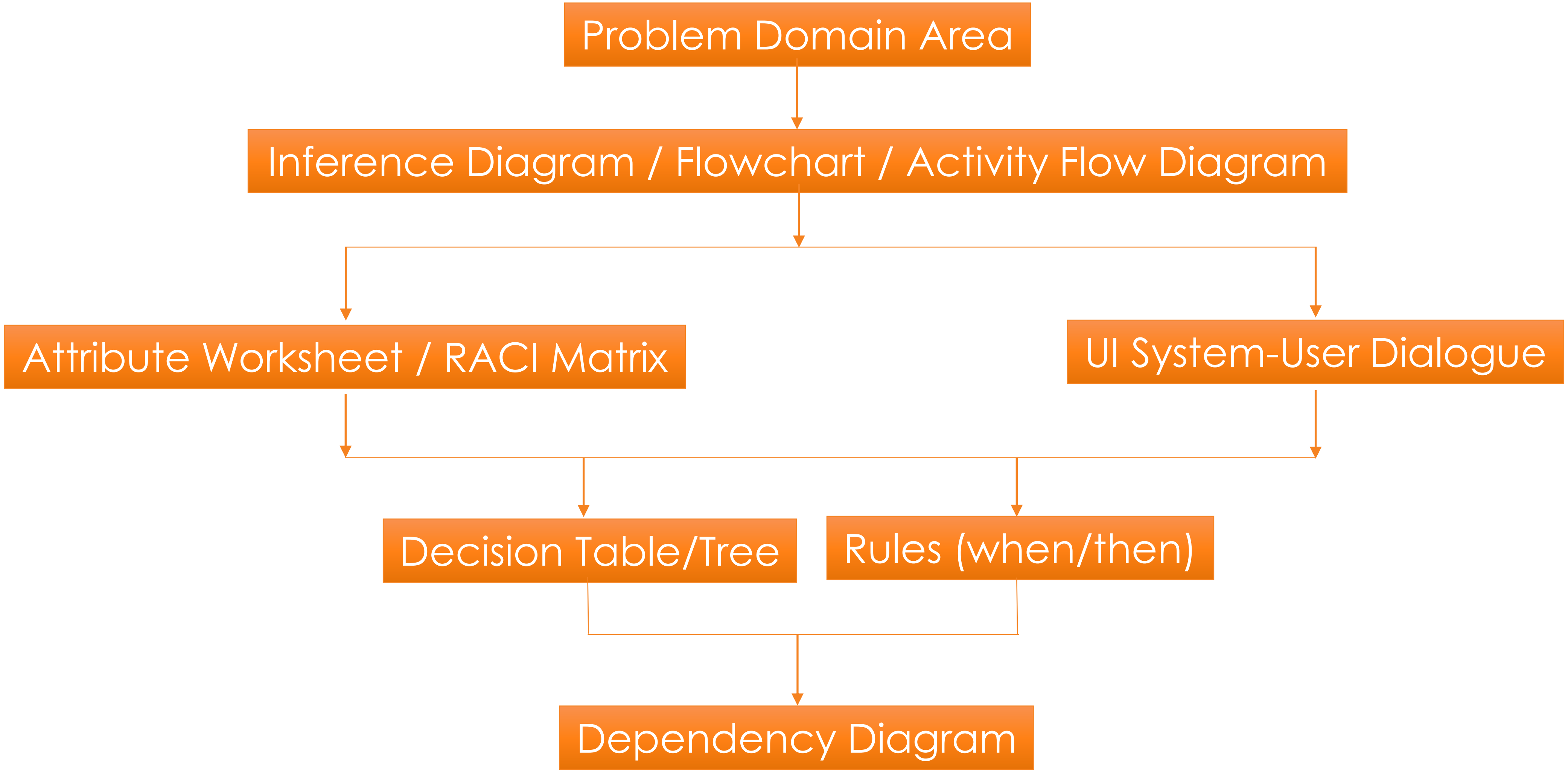
Key

R - Responsible	Actually completes the activity - responsibility can be shared. Degree of responsibility is determined by the "A".
A - Accountable	Has Yes/No authority - there can only be one "A" per activity
C - Consulted	Involved prior to decision or action - two-way communication
I - Informed	Needs to know of the decision or action - one-way communication.

- **Responsible; Accountable; Consulted; Informed**
  - **R** for responsible means, "the person who actually does the activity". Responsibility for an activity can be shared, if necessary.
  - **A** for accountable means, "the buck stops here", and the role has ultimate yes/no authority.
  - **C** for consulted means, "kept in the loop", and implies two-way communication prior to the activity.
  - **I** for informed means, "kept in the picture", and implies one-way communication after the activity.
- **Best Practices:**
  - There can only be one accountability (**A**) per activity.
  - Recommend one responsibility (**R**) only per activity.
  - Roles can combine both accountability and responsibility for activities.
  - Minimize the number of consults (**C**) and informs (**I**).
  - Authority must accompany accountability.
  - Don't map decision points on the RACI matrix, only activities.

2.3 KNOWLEDGE MODELS (ACQUIRED → REPRESENTED)

Design Process



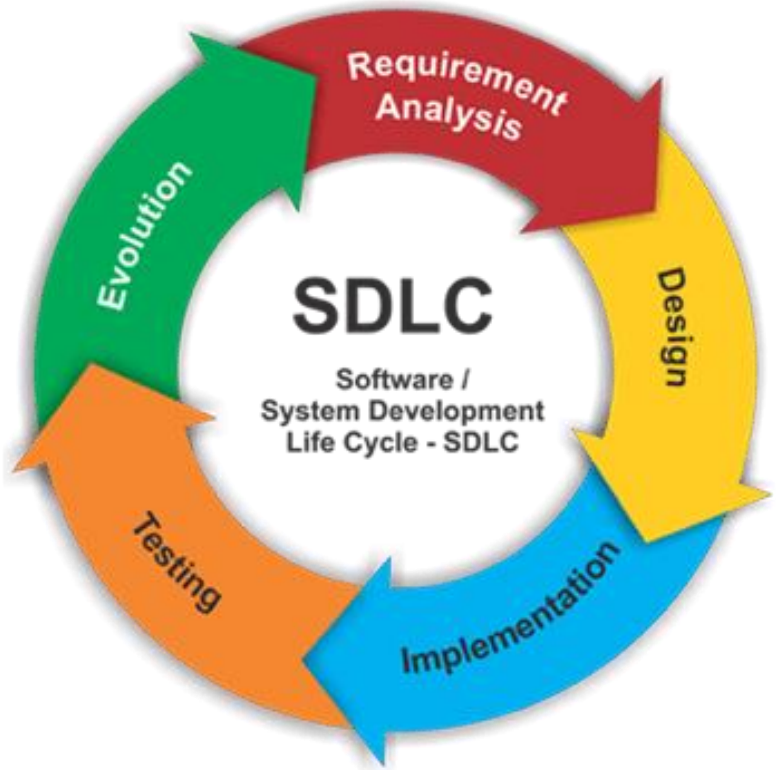
Exercise

- Refer to Airport Gate Assignment System (AGAS) case.
- Create relevant Knowledge Models.

2.4 WORKSHOP
KNOWLEDGE MODELLING

2.4 WORKSHOP KNOWLEDGE MODELLING

- **Requirement Analysis**
  - Problem selection: Identify business value and purposes
- **Design (Knowledge Representation and Acquisition)**
  - Knowledge acquisition, interviews
  - Definition of problem domain: Draw high level Inference Diagram
  - System design: Compose other relevant Knowledge Models
- **Implementation (KIE Development)**
  - System development: KIE tools
- **Testing**
  - Integrate, test, revise, deploy, and use
- **Evolution**



https://i1.wp.com/melsatar.blog/wp-content/uploads/2012/03/sdlc.png?fit=830%2C374&ssl=1

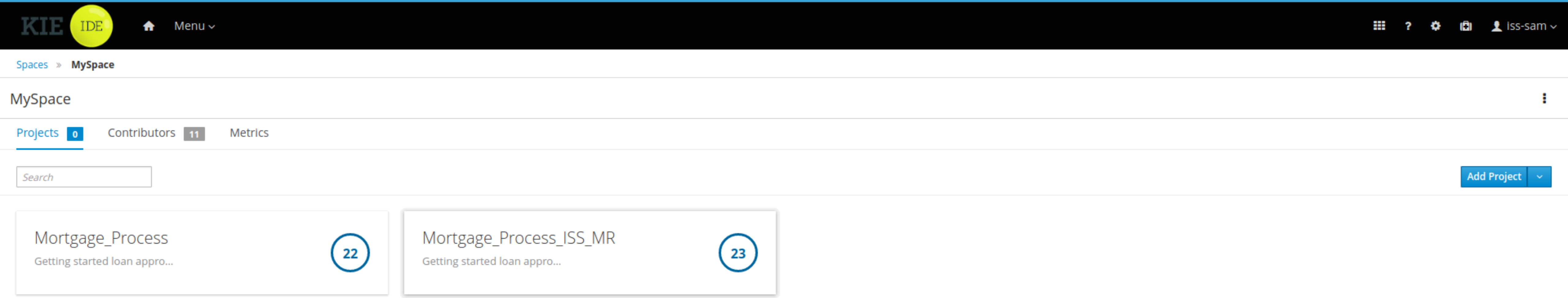
2.4 WORKSHOP KNOWLEDGE MODELLING

- **KIE System Enhancement – Individual Work**

  Enhance KIE home loan system using machine reasoning rule task
  - Enhance: Data Model: Objects; Fields; Data Types
  - Enhance: When-Then Guided Rules
  - Enhance: Business Process, Task, Form
  - Integrate, test, revise, deploy, and use
- **Knowledge Representation and Acquisition – Individual Work**

  Construct knowledge models:
  - Identify a business opportunity to use reasoning system
  - Study online documented knowledge source as knowledge acquisition
  - Compose knowledge models in spreadsheets

☺ Candidate Project: HDB BTO; Airport Gate Assignment System (AGAS); DoReMi

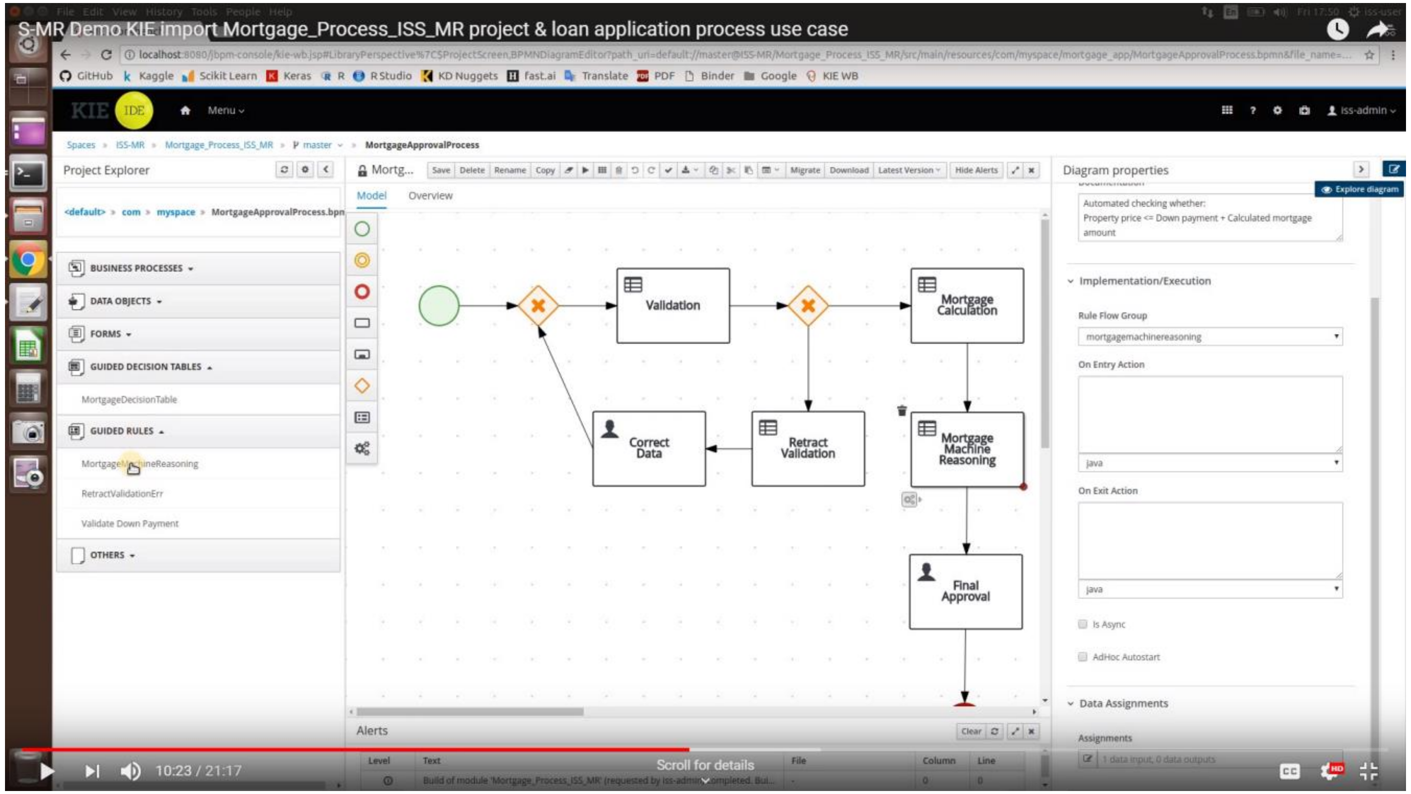


Reference video link

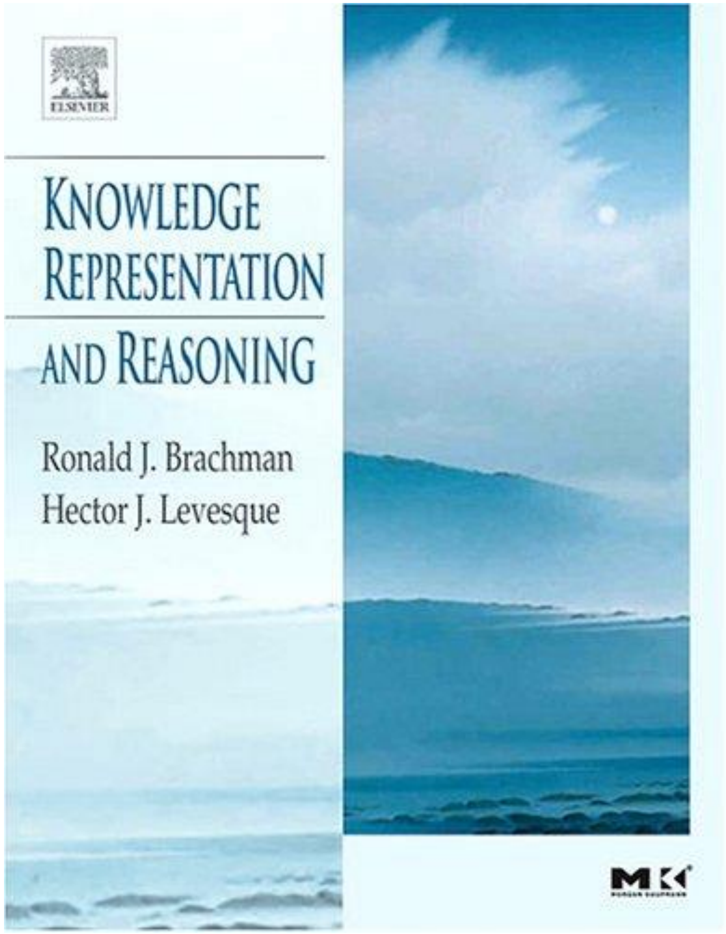
https://youtu.be/s_8rct45b84

Reference code link

https://github.com/IRS-MR/S-MR-Workshop/tree/master/S-MR-Workshop2



DAY 2 REFERENCE



1. Designing a decision service using guided rules

   https://access.redhat.com/documentation/en-us/red_hat_decision_manager/7.2/html-single/designing_a_decision_service_using_guided_rules/

2. KIE Workbench Tutorial : Human Machine Interaction

   https://www.youtube.com/watch?v=NfNnUsr66Cc

3. KIE Workbench Tutorial : Guided Decision Tables

   https://www.youtube.com/watch?v=qBgxVoc2qfw

4. Jay Pujara & Sameer Singh. (2018). Mining Knowledge Graphs from Text

   https://kgtutorial.github.io/