Noun Hunting Technique: There are many techniques to capture requirements and present a cohesive picture of real-world system. Most widely used techniques to capture system requirements include DFA/DCA and Domain Modeling. The main purpose of all these techniques is to present the system working in a single glance so the analyst can transform system into software.

Introduction: Domain model is a widely used technique for capturing of entities in a real-world system. There are multiple ways we can construct a domain model. Noun hunting is used for its simplicity and effectiveness.

Steps for Noun Hunting:
The technique is divided into steps that each produce a meaningful result. These steps and their details is as follows.

1. **Collection of Entities**
   - Read the requirement document carefully. Underline or mark nouns appearing in each sentence. Write those nouns separately.
2. **Elicitation of Entities**
   - In the list of collected entities select those that are related to our system. Discard all other irrelevant entities.
3. **Domain Model construction**
   - Use right-angle rectangles to write these entities.
4. **Establish Relationships**
   - Connect entities using requirement document with each other.
5. **Naming Relationships**
   - Write suitable titles on each relationship such that when read they form a sentence.
6. **Establish Special Relationships**
   - Special relationships include containment (aggregation, composition) inheritance, specialization, interfaces (abstract classes) etc. Each project has a depth of detail meaning a domain model may go through multiple evaluations to complete.

CASE STUDY EXAMPLE:
FCIT has placed tea dispensers in campus for students to enjoy free tea and cold beverages. Each machine has an interface that recognizes a student by ID card. Each student has a monthly quota and by using it can obtain cups of beverage from the machine. The machine has three types of beverages.

1. Tea
2. Coffee
3. Cold Coffee

The student used his/her ID card to activate machine. The machine has a beverage maker that deals all the orders. In an order student decides first on type of beverage he/she wants. The student mentions the product and machine confirms by showing availability message. Then student specifies number of cups for each product type. The machine checks availability of quota from the quota registers and prompts with message. In case of available quota the

machine asks for a confirmation of order. After confirmation machines dispense order by using a dispenser. And updates the state. Student then collects his/her order and leaves.

STEP – 1

FCIT has placed **tea dispensers** in **campus** for **students** to enjoy free **tea** and **cold** **beverages**. Each **machine** has an **interface** that recognizes a **student** by **ID card**. Each **student** has a monthly **quota** and by using it can obtain **cups of beverage** from the **machine**. The **machine** has three types of **beverages**.

1. **Tea**
2. **Coffee**
3. **Cold Coffee**

The **student** used his/her **ID card** to activate **machine**. The machine has a **beverage maker** that deals all the **orders**. In an **order student** decides first on type of **beverage** he/she wants. The **student** mentions the **product** and **machine** confirms by showing availability **message**. Then **student** specifies **number of cups** for each **product** type. The **machine** checks **availability** of **quota** from the **quota registers** and prompts with **message**. In case of available quota the **machine** asks for a **confirmation of order**. After **confirmation machines** dispense **order** by using a **dispenser**. And updates the **state**. **Student** then collects his/her **order** and leaves.

STEP – 2

Candidate Entities	Discarded Entities
Dispenser/Tea Dispenser	FCIT
Student	Campus
ID Card	Cups
Order	Interface
Quota	State
Beverage (Tea, Coffee, Cold Coffee)	Hot/Cold
Quota Register	Machine
	Product
Beverage Maker	Availability
Message	
Dispenser	

STEP – 3,4,5,6

A UML class diagram showing:
- DrinkDispensingMachine
- Student
- QuotaRegister (uses)
- Dispenser (uses)
- ID_Card (validates, has)
- updates QuotaRecord
- QuotaRegister contains QuotaRecord
- BeverageMaker (uses)
- manages Order
- places Order (Student)
- dispense Order
- BeverageMaker makes Beverages
- Order → Beverages
- Beverages generalizes Coffee, ColdCoffee, Tea