Investigation and Evaluation of Object Oriented Analysis techniques

Huan Zhu
This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Huan Zhu
Address:
Polhemsgatan 27A
37140 Karlskrona
Sweden
E-mail: Huanday@hotmail.com

University advisor(s):
Miroslaw Staron
APD

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.tek.bth.se
Phone : +46 457 38 50 00
Fax : + 46 457 271 25
ABSTRACT

The technique of Object Oriented Analysis (OOA) has emerged only in the last decade. Although the technique of OOA is still new, its popularity has been increasing and it has already entered the mainstream of object oriented system development. This thesis makes a summary of four OOA methods and investigates the behaviors of all methods under different criteria. Through comparing the four methods, differences between methods are shown and analysts can select the appropriate one to meet his/her requirements.

**Keywords:** domain model, metric, method, class, attribute, association, and inheritance
## CONTENTS

1. Introduction ................................................................. 1
2. Related work ............................................................... 2
3. Introduction to Object Oriented Analysis .......................... 3
   3.1 Object Oriented Analysis ........................................... 3
   3.2 Domain model ........................................................ 3
4. Model construction process .......................................... 4
   4.1 Identification of candidate classes ............................ 4
      4.1.1 Coad and Yourdon’s method ............................... 4
      4.1.2 Craig Larman’s method .................................... 5
      4.1.3 Ivar Jacobson’s method .................................... 5
      4.1.4 KRB Seven-Step method ................................... 6
   4.2 The refinement of candidate classes .......................... 6
      4.2.1 Coad and Yourdon’s method ............................... 6
      4.2.2 Craig Larman’s method .................................... 7
      4.2.3 KRB Seven-Step method ................................... 7
   4.3 The identification of attributes ................................ 7
      4.3.1 Coad and Yourdon’s method ............................... 8
      4.3.2 Craig Larman’s method .................................... 8
      4.3.3 Ivar Jacobson’s method .................................... 9
      4.3.4 KRB Seven-Step method ................................... 9
   4.4 The identification of associations ................................ 9
      4.4.1 Coad and Yourdon’s method ............................... 9
      4.4.2 Craig Larman’s method .................................... 9
      4.4.3 Ivar Jacobson’s method .................................... 10
      4.4.4 KRB Seven-Step method ................................... 10
   4.5 The identification of inheritances ............................ 12
      4.5.1 Coad and Yourdon’s method ............................... 12
      4.5.2 Craig Larman’s method .................................... 12
      4.5.3 Ivar Jacobson’s method .................................... 13
      4.5.4 KRB Seven-Step method ................................... 13
5. Case study ................................................................. 14
   5.1 Design ................................................................. 14
      5.1.1 Objective ..................................................... 14
      5.1.2 Instruments (metrics) ...................................... 14
   5.2 Case study operation ............................................. 17
      5.2.1 Process ...................................................... 17
      5.2.2 Threats to this case study ............................... 18
6. Conclusions ............................................................... 29
7. References ............................................................... 30
8. Appendix ................................................................. 31
1 INTRODUCTION

Object oriented development is very popular in current IT industry and has rapidly become accepted as the preferred system development method in many software companies. During the process of developing object-oriented system, Object Oriented Analysis (OOA) is the first activity to be applied. Through an effective Object Oriented Analysis, system analysts and problem domain experts can have a clear understanding of problem domain, and the interactions between them will be improved correspondingly. Usually, there are several outcomes to any object-oriented analysis; these outcomes include the requirement model, classes, attributes of classes, operations or services of classes and associations of classes etc.

Object Oriented Analysis is an important activity. However, an important question is how we should perform it systematically and effectively? It is also important how we should identify classes, attributes of classes, associations of classes and inheritances? To find answers for the above questions, this thesis summarizes four OOA methods used to guide analysts in creating domain models effectively. Furthermore, if system analysts have certain requirement related to OOA, which method should be chosen to meet the requirement is a significant issue. To find the answer to this question, a case study is performed as part of this thesis to evaluate the four methods’ abilities in creating domain models. Several goals, questions and metrics are defined in case study to compare domain model in a measurable way. Through comparing the four methods’ behaviors, one relationship between methods (regarding to one metric) is discovered and selections of best method (regarding to several metrics) are discussed and presented.

This thesis is structured in the following way. At first, work related with this thesis is discussed. In the second part, an introduction to object oriented analysis is given. The introduction covers objected oriented analysis and domain model. The summary of all four methods is presented in the third part. The summary explains how these four methods can be applied to create domain models in detail. The case study is discussed in the fourth part of this thesis. Measurement results are analyzed and all conclusions are drawn in the case study part. A final conclusion of this thesis is presented in the fifth part and references are listed in the sixth part. The appendix is presented in the last part of this thesis. Appendix includes three requirement models, twelve domain models and twelve sets of candidate.
2 RELATED WORK

Currently, there are many different types of comparisons between OOA methods that have different focuses with this thesis. For example, the comparison implemented in [9] focuses on comparing the concepts, notations, processes, pragmatics, and marketability of different methods. The comparison implemented in [10] focuses on discovering differences between six OOA methods in structural, functional and behavioral modeling and identifying each method’s strengths and weaknesses.

Although the above comparisons are very useful, they overlook comparing OOA methods’ abilities in creating domain model. As we know, domain model is a central and important deliverable from OOA, and the main purpose of OOA is to create a useful domain model. If we compare several OOA methods and find their strengths and weaknesses in producing domain model, we could select an appropriate method to produce domain model meeting our requirements and perform OOA more effectively. That is why this thesis focuses on comparing several OOA methods’ abilities in creating domain model.
3  INTRODUCTION TO OOA

This section makes a brief introduction to the motivation for performing Object Oriented Analysis, and the concept of domain model is explained and discussed as well.

3.1  Object Oriented Analysis

In the development life cycle of Object Oriented system, Object Oriented Analysis is the first activity to be performed and its outcomes are the basis for Object Oriented Design. The main purpose of Object Oriented Analysis is to create two important models to obtain an understanding to the problem domain under considerations, the two models include requirement model and object model. Requirement model is the first deliverable produced in the phase of Object Oriented Analysis and it consists of various types of requirement specifications, for example use cases, context diagram and interface descriptions etc. Based on requirement model, system analysts perform a systematic study and analysis on the problem domain and create object model to visually represent the problem domain. The object model usually consists of class diagram, class definitions, attribute lists and definitions, and notes and business rules. Classes in class diagram correspond to real-world concepts in the problem domain and their attributes, operations and associations are naturally reflection of the reality in the real world. Besides requirement model and object model, some other models could be produced in the phase of Object Oriented Analysis as well, for example the business domain model. [2]

3.2  Domain model

Domain model is a visualization of real-world concepts that exists in a problem domain under study and it consists of four elements: classes, attributes of classes, associations between classes and inheritances. Each class corresponds to certain real-world concept, and it could be a thing, an idea or an object in the real world. Since domain model is only a representation of real world things or concepts in a domain of interest, therefore software components should not appear in it. Some software artifacts, like database and responsibilities should be excluded from domain model. Issues related with designing graphical user interface should not be considered in domain model as well. [4]

Domain model can be viewed as a special type of class diagram, in which all classes don’t involve the operation part. During object modeling, domain model is created at an early stage, and system analysts will refine it to produce class diagram. During the process of refinement, operations will be identified and added into each class, and other techniques could be applied in refining as well.
4 MODEL CONSTRUCTION PROCESS

The method of Coad Yourdon is extracted from [1]. The method of KRB Seven-Step is extracted from [2]. The method of Jacobson is extracted from [3]. The method of Craig Larman is extracted from [4]. During extraction, some parts of the original methods were tailored to meet the special purpose of this thesis. Since this thesis focuses on creating domain models, operations or services are not included in some parts of this thesis.

Usually we can divide the domain model construction process into five basic processes: the identification of candidate classes, the refinement of candidate classes, the identification of attributes, the identification of associations and the identification of inheritances. In the following, how each method can be applied to each of above processes is discussed.

4.1 Identification of candidate classes

The set of candidate classes is the source for identifying classes actually needed. Each method presents different technique in discovering candidate classes.

4.1.1 Coad and Yourdon’s method

To identify potential candidate classes, Coad and Yourdon in [1] suggested we should look for eight items in the requirement specification: structure, other systems, devices, things or events remembered, roles played, operational procedures, sites, and organizational units. The eight items are listed below and they are referenced from [1]:

1. Structures  
   Coad and Yourdon in [1] discussed that structure is the most important item for identifying potential classes. Analysts should find potential generalization-specialization and whole-part structures, and include those classes that are part of these structures into the candidate class list.

2. Other systems  
   Find out other software systems that the system under study will interact with. There could be many different types of interaction.

3. Devices  
   Look for potential hardware devices that may need to have interactions with the system under study and include them into the list of candidate classes.

4. Things or events remembered  
   In the problem domain, identify things or events that must be recorded in the system and include them into candidate class list.

5. Roles played  
   From the requirement specification of the system under study, find out the potential role or roles that people could play. Add those identified roles into the set of candidate class.

6. Operational procedures  
   The system under study may require keeping some procedure descriptions used to guide certain operations. Include those operational procedures into the set of candidate classes.
7. Sites
Some physical locations or sites may need to be held over time by the system under study. Include those physical locations or sites into the list of candidate class.

8. Organizational units
Some organizations may be mentioned or described in the system’s requirements. Place them into the set of candidate classes.

### 4.1.2 Craig Larman’s method

To find potential candidate classes, Larman in [4] suggested two techniques: identify noun phrases and the use of candidate class category list. In this thesis, the technique of using candidate class category list is used. [4]

The candidate class category list contains eighteen categories that could guide analysts in finding corresponding candidate classes. Each category in the table is an abstraction of real-world things; concepts etc and sets of candidate classes generated by all categories are quite independent from each other. These categories are equally important and some of them appear in other methods as well. Due to large amount of categories, the category list could generate many more candidate classes compared with other methods that have fewer categories. In the following, all of categories are listed and they are quoted from [4].

1. Physical or tangible objects
2. Containers of other things
3. Rules and policies
4. Specifications, designs, or descriptions of things
5. Things in a container
6. Catalogs
7. Places
8. Abstract noun Concepts
9. Records of finance, work, contracts, legal matters
10. Transactions
11. Organizations
12. Financial instruments and services
13. Transactions Line items
14. Events
15. Manuals, documents, reference papers, books
16. Roles of people
17. Processes
18. Other computer or electro-mechanical systems external to the system

### 4.1.3 Ivar Jacobson’s method

In this method, Jacobson in [3] classifies classes into three types: interface classes, entity classes and control classes. Interface classes act as interface between prospective system users and the system. Through interface classes, system users’ actions can be translated into events or operations in the system, and events or operations in the system can be translated into something that is present to system users as well. [3] To identify interface classes, there are three ways, which are listed below. All ways are reference from [3].

1. Employ system interface descriptions.
2. Identify prospective system users first, and then start from the system users to find interface classes.
3. Analyze use cases and extract interface specific functionalities.

Entity classes usually correspond to some real-world concepts. To model the information that will be kept by the system for a long time, we can use entity classes. Generally entity classes can survive use cases and the system should still keep them even if the use case has already been finished. To identify entity classes, we should check problem domain and read use cases, and discover those information that need to be handled over by the system for a long time. However, if some information cannot survive over use case, but they are essential to system, we can still model them as entity classes. [3]

Control classes usually exist in some complex use cases and they are used to unite interface classes and entity classes so that complete use case can be formed. Generally, the life cycle of control classes is transient and they exist only during the execution of certain use case. To identify control classes, we can search the use case for behaviors that remain after the identification of interface classes and entity classes is finished. Those remaining behaviors provide good guidance in finding corresponding control classes. [3]

The outcome of this step will be the set of classes that are really needed, thus the step of refining classes is not needed in Ivar Jacobson’s method.

4.1.4 KRB Seven-Step method

In this method, candidate classes should be discovered through identifying potential nouns. There are many sources for finding nouns, for example client interviews, requirement documents and brainstorming etc. In this thesis, we focus on using requirement documents. When we employ requirement documents to identify candidate classes, Brown in [2] suggested the documents should be scanned carefully for nouns and all identified nouns should be written down in a list. Each identified noun represents one candidate class. From the set of nouns, classes that are really needed by the system could be easily discovered. [2]

4.2 Refinement of candidate classes

The set of candidate classes must be refined to exclude those unneeded candidate classes. Since Ivar Jacobson’s method has already discovered all really needed classes in last step, thus it is not included in this section.

4.2.1 Coad and Yourdon’s method

To select classes that are really needed from the set of candidate classes, Coad and Yourdon discussed that we should check the candidate classes for their needed remembrance first. If one candidate class can be described with some interesting attributes or the system needs to remember something about that class, we should probably include this class into domain model. If a candidate class cannot be described in the system, but it can provide certain behavior that is distinct and essential to system, we should still include this class. [1]

There are some other criteria used in filtering out inappropriate classes as well. The first one is that class in the domain model should usually have multiple attributes. If one candidate
class has only one attribute, its relevance should be suspected. The second criterion is that class should normally have more than one object. Yet if a class with one object can really reflect the problem domain, it can still be added into domain model. [1]

During refinement, those classes that can be derived from other classes should be excluded. Instead, we should capture attributes and behaviors in classes from which the derived classes can be obtained. This technique will be applied to all other three methods as well.

### 4.2.2 Craig Larman’s method

Larman suggested several techniques to refine the set of candidate class and get the set of class really needed. First, Larman suggested that those classes that are not relevant to system requirements should be excluded. When using the candidate class category list, candidate classes are normally captured to meet the category requirement and their relevance to requirement could be overlooked. Therefore, we should check each candidate class for its pertinence to system requirements and delete those irrelevant ones. [4]

Sometimes it is hard to determine that certain candidate class should be modeled as class or attribute, Larman suggested one principle: “If we do not think of some class X as number or text in the real world, X is probably a class, not an attribute”. Furthermore, Larman stated that there should be fairly rare attributes in domain model and it is better to make a candidate class as independent class when analyst is still not certain how to model it. [4]

### 4.2.3 KRB Seven-Step method

Once the list of nouns is produced, each candidate classes should be subjected to two checks: real-world identifier and sample attributes and behaviors. The real-world identifier is used to check whether objects of certain candidate class can be distinguished by some way. To a really needed class, its objects must be distinct, and there is usually some way to tell them apart. For example, the attribute of customer ID can be used to distinguish objects belonging to the class of customer. [2]

Real-world identifier is not enough for validating a candidate class. A valid class should carry some interesting data or attributes. In other words, a valid class should have some data items that the system is interested or users may need to know. If the system does not need to know anything about certain class, it is very likely this class should be excluded. However, if a class that does not have any valid attributes can provide certain behavior that is distinct and essential to system, we still should include this class. [2]

### 4.3 Identification of attributes

Attributes are used to describe classes in the problem domain and store classes’ important information. In the following, each method’s attribute identification technique is presented.

### 4.3.1 Coad and Yourdon’s method

Coad and Yourdon gave a very general attribute definition in their book [1], the definition is: “An attribute is some data (state information) for which each object in a class has its own
value”. They designed several questions used in identifying attributes, these questions are listed below and they are quoted from [1].

1. How am I described in general?
2. How am I described in this problem domain?
3. How am I described in the context of this system’s responsibilities?
4. What do I need to know?
5. What state information do I need to remember over time?
6. What states can I be in?

We should ask the questions from the perspective of a single class, and the attributes generated by the above questions should be used to describe class and store information about class. [1]

Furthermore, the identified attributes should correspond to single value or a tightly related grouping of values so that domain model will be simpler and clearer for reading. Furthermore, if an attribute has repeating values, it is very likely that additional classes should be created. [1]

### 4.3.2 Craig Larman’s method

Larman defined attribute as “a logical data value of an object”. To identify suitable attributes for certain class, we should check this class and find out all of its data items that are relevant to system requirements. Larman suggested one technique to discover attributes, the technique is: “Include the following attributes in a domain model: Those for which the requirements (for example, use cases) suggest or imply a need to remember information”. [4] This technique can be used to identify attributes quickly. However, this technique may not ensure the identification of all attributes that are needed.

After acquiring attributes for all classes, Larman presented some way to refine the original set of attributes. He classifies the attributes into correct attributes and incorrect attribute. Correct attributes should correspond to the following data types: Boolean, Date, Number, String (Text), Time, Address, Color, Geometric (Point, Rectangle), Phone Number, Social Security Number, Universal Product code, SKU, ZIP, Enumerated types, and they will be added into domain model as attributes of class. Incorrect attributes correspond to those compound attributes, which normally are complex domain concepts, and these compound attributes should be transferred to associations or class. [4] In addition, Larman discussed some situations when correct attributes should be modeled as classes. The situations are listed below and they are referenced from [4].

1. There are separate sections in this attribute, for example the name of person. Since the name of person includes first name and last name, we should model it as independent class.
2. Certain behavior or operations is associated with this attribute, for example the attribute of social security number normally should be validated, therefore we should model it as class.
3. There are other attributes that belong to this attribute.
4. This attribute is a quantity that has a unit, for example payment amount that has a unit of currency.
5. The attribute is an abstraction or generalization of one or several types.
4.3.3 Ivar Jacobson’s method

Jacobson stated in his book that attributes are used to store information. To identify attributes for certain class, we should find out the data items that can be used to describe the class in the context of system requirements and use attributes to store important information. [3]

4.3.4 KRB Seven-Step method

Brown stated in his book: “attributes represent all the data items that any user might ever need to know about the object”. To each class, we should identify all data items that system requirement indicates needs to know and model those need-to-know data items as attributes of that class. [2]

4.4 Identification of associations

Through associations, classes interact with each other to fulfill certain system responsibility. In the following, each method’s association identification technique is discussed.

4.4.1 Coad and Yourdon’s method

To identify potential associations, all classes should be checked one by one. To each class, go to check the problem domain and system’s responsibilities, and find out other classes that need cooperation with this class to fulfill some system responsibilities. Then add instance connection lines from this class to those identified classes. [1]

Coad and Yourdon classify the aggregation, which can also be called whole-part, into three types: assembly-parts, container-contents, and collection-members. To be clearer, let’s see three typical examples. Aircraft and engine can be modeled as assembly-parts. Aircraft and pilot can be modeled as container-contents, and we can model organization and clerk as collection-members. [1]

4.4.2 Craig Larman’s method

Larman defined association as: “A relationship between types (or more specially, instances of those types) that indicates some meaningful and interesting connection”. To identify associations, he suggested using so called common association list. This common association list contains sixteen categories and each category is useful source for potential association identification. By following those categories, lots of associations may be generated. However, not all of them could be included into domain model, many of them should be filtered out. Larman discussed that we should only keep those associations that are relevant to system requirements. If system requirements don’t suggest that certain association is needed, it is very likely that association should be excluded. In addition, many redundant or derivable associations may exist, and they should be filtered out as well. In the following, all of categories are listed and they are quoted from [4].

1. A is a physical part of B
2. A is a description for B
3. A is an organizational subunit of B
4. A is a transaction related to another transaction B
5. A is a logical part of B
6. A is a line item of a transaction or report B
7. A uses or manages B
8. A is owned by B
9. A is physically contained in/on B
10. A is known/logged/reported/captured in B
11. A communicates with B
12. A is next to B
13. A is logically contained in B
14. A is a member of B
15. A is related to a transaction B
16. A is an event related to B

Larman discussed some situations when aggregation should be created. Those situations are:

1. When the lifecycle of the part is dependent on the composite. [4]
2. When there exists an obvious whole-part physical or logical structure. [4]
3. When some properties or operations of part are propagated from the composite. [4]

4.4.3 Ivar Jacobson’s method

If two classes need to know the existence of each other to solve certain task, we should add association between these two classes. To identify all associations, we could apply this rule to each class and identify all relevant classes that have mappings with this class to fulfill some system responsibilities. [3]

The aggregation in this method is reflected as that certain class is composed of other classes. In other words, if a cluster of classes is logically or physically part of another class, we can create corresponding aggregation to reflect this special type of associations. Apply this rule to each class to discover all potential aggregations. [3]

4.4.4 KRB Seven-Step method

Brown defined associations as: “relationship expressing the interactions between instances of two classes, represented by the verb that describes what they do to each other, and/or by the nouns for the roles that each plays in the life of the other”. To identify associations, Brown suggested we should discover verbs that are relevant to the operations of users’ business first. Those verbs could guide users through finding associations and will be used to name associations. [2]

To capture all possible associations, Brown suggested an interesting way: drawing pictures on board. First, from top to down list all the classes on board, and draw box representing the first class (the Top one). Start from the first class; check it against all the rest classes (which are under it) in pairs for associations. Each time after checking, draw a curved line between the pair of classes, implying these two classes have already been checked. If there are interactions that are significant in the operation of users’ business between the pair of classes, draw the box (representing the class other than the top one) and use a line to connect these two boxes, with the corresponding Verb on the line. After the top class has been compared with the rest of the classes, the same procedure is repeated for each of the rest classes till all associations are discovered. Through using the way of drawing pictures on board, the missing of any association will be avoided. There are several figures below to
illustrate this method. Figure #1 represents the initial state. In figure#2, since there is association between customer and product, box that is used to represent product is drawn and two boxes are connected using lines. In figure#3, since there is no association between customer and vendor, we don’t need to draw box used to represent vendor. In figure#4, all associations between classes are identified. [2]
Similarly to Coad and Yourdon’s method, Brown classified aggregations into three types: component-assembly, container-contents and collection-members. [2] To find aggregations, Brown designed several questions, which are listed below. These questions are quoted from [2].

1. Are there any attributes in this class that does not describe this class, but some thing within it?
2. Are there any operations or associations that are related with only “piece” or “part” of this class?

When we apply these questions in the context of domain modeling, the operation part in the above questions will not be considered.

4.5 Identification of inheritances

Inheritance represents commonalities and specialties between several classes. The mechanism of inheritance brings a lot of advantage to Object Oriented programming. In the following, each method’s inheritance identification technique is presented.

4.5.1 Coad and Yourdon’s method

In the Coad and Yourdon’s method, when several classes that are variations of certain concept have common attributes, it could indicate a super class or generalization should be created. Oppositely, if a class represents certain general concept that has some variations, it could indicate that several specializations or subclasses with additional attributes should be created. [1]

Coad and Yourdon designed several questions used in checking the validity of potential super class or subclass, these questions are listed below and they are referenced from [1].

1. Is this specialization or generalization in the problem domain?
2. Is this specialization or generalization within the system’s responsibilities?
3. Will there be inheritance?
4. Will the specializations or generalization meet the criterion used in refining candidate classes?
4.5.2 Craig Larman’s method

Larman suggested some situations when a candidate subclass of certain class should be included into domain model. Those situations are listed in the following and they are referenced from [4]:

1. The candidate subclass has more attributes than that class and those additional attributes are useful.
2. The candidate subclass has more associations than that class and those additional associations are useful.
3. The candidate subclass is treated or manipulated differently from that class in ways of interest.
4. The candidate subclass represents certain animate concept (for example, animal, robot) and that concept behaves differently than that class, in ways of interest.

Similarly, some motivations for the creation of super class are presented as well, which are listed below. The motivations are referenced from [4].

1. All potential sub classes are variations of certain similar concept.
2. All potential sub classes have same attributes.
3. All potential sub classes have same associations.

4.5.3 Ivar Jacobson’s method

Similar to Coad and Yourdon’s method, the super class or generalization are created in the reason that a cluster of classes that are variations of certain concept has common attributes, and subclasses or specializations are created because subclasses or specializations represent certain useful variations with the concept of super class, and additional useful attributes will be added into subclasses or specializations. [3]

4.5.4 KRB Seven-Step method

In this method, two factors will determine the creation of inheritance: attributes and associations. To identify potential subclasses or specializations, there are two questions to be asked, which are listed below. The questions are referenced from [2].

1. Are there some useful attributes that only apply to certain objects of this class?
2. Are there some useful associations that affect only a subset of objects within this class?

These questions should be asked from the perspective of each class. To identify potential super classes or generalizations, we could ask the question: “Are there some attributes and/or associations that appear in several classes?”. Furthermore, subclasses should represent variations of certain concept, while super class should represent certain general concept that has several variations. [2]
5 CASE STUDY

In this case study, each method is applied to create domain models for three requirement specifications and domain models is measured and analyzed to identify all methods’ features in creating domain models. This section consists of three parts. The first part is called design part, in which several sets of goal, question and metric are designed and presented. In the second part, the operation process of case study is explained. The measurement results are summarized and analyzed in the third part.

5.1 Design

To perform this case study, several goals, questions and metrics must be designed. In this section, five sets of goal, question and metric are defined and presented. In addition, the objective of this case study is described as well.

5.1.1 Objective

This case study is aimed to compare four methods’ ability in creating domain models. The comparison generates one relationship between methods (with respect to one metric) and selections of best methods (with respect to several metrics). The objective of this case study is to show the differences in terms of the case study’s results between those four methods in creating domain models so that organizations could choose the method that is needed.

5.1.2 Instruments (metrics)

The GQM method [5] is employed in the course of this case study. GQM is the abbreviation of Goal/Question/Metrics method, and it is a popular method used in defining measurement on processes, software projects and product etc. GQM method contains four phases: the planning phase, the definition phase, the data collection phase and the interpretation phase. Measurement goals, questions and metrics are defined in the definition phase. Data are collected and checked in the data collection phase, and measurement results are summarized and analyzed in the interpretation phase [5].

By following the GQM method, several goals and corresponding questions, metrics are created in this thesis. Currently, many software metrics are defined for measuring domain models and class diagrams. However, some metrics are not relevant to this thesis, and some metrics are not applicable due to the simplicity of example projects. Therefore, only several traditional metrics are selected to perform the case study. Each goal is related with one question and one or two metrics. Question is designed so that the corresponding goal could be achieved. To answer the question of each goal, one or two metrics are developed to provide answers to that question. In the following tables, all goals, and their corresponding questions and metrics are listed.

<table>
<thead>
<tr>
<th>Goal#1:</th>
<th>Establish a relationship between methods with respect to the number of candidate classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questions:</td>
<td>Under the same requirement specification, which method generates more candidate classes?</td>
</tr>
</tbody>
</table>
Metric: The number of candidate classes per model

Metric Notation: This metric is denoted as $NCC$ (Number of Candidate Classes). $NCC_K$ represents for KRB Seven-Step method, $NCC_L$ for Craig Larman’s method, $NCC_J$ for Coad and Yourdon’s method, $NCC_I$ for Ivar Jacobson’s method.

Table 1: Goal #1 and it’s metric and question

The set of candidate classes is the source for identification of really needed classes. Some analysts could prefer less candidate classes, since less candidate classes may save the time spent in refining. While some other analysts could like more candidate classes, since more candidate classes could avoid the missing of any important class. If the relationship between methods with respect to the number of candidate classes is identified, analysts can have an objective selection among all methods to meet his/her requirements.

Goal#2: Identify the best method with respect to the metric of number of classes. The method that can generate minimum number of classes is defined as the best one.

Questions: Under the same requirement specification, which method generates fewer classes?

Metric: The number of classes in the domain model

Metric Notation: This metric is denoted as $NC$ (Number of Classes). $NC_K$ represents for KRB Seven-Step method, $NC_L$ for Craig Larman’s method, $NC_J$ for Coad and Yourdon’s method, $NC_I$ for Ivar Jacobson’s method.

Table 2: Goal #2 and it’s metric and question

If too many classes are created, the domain model could be more complex. The reason is that large amount of classes could require many more attributes, associations and inheritances as well. If this goal is achieved, we can identify a method that could generate domain model with least complexity.

Goal#3: Identify the best method with respect to the average number of attributes. The method that can generate the lowest average number of attributes is defined as the best one.

Questions: Under the same requirement specification, which method generates a lower average number of attributes?

Metric: 1. The number of attributes in domain model. $NA$ can be obtained by summing up the number of attributes for each class in the domain model. 2. The average number of attributes per class. This metric can be obtained through: $NAA = \frac{NA}{NC}$

Metric Notation: 1. The first metric is denoted as $NA$ (Number of Attributes). $NA_K$ represents for KRB
Seven-Step method, $NA_L$ for Craig Larman’s method, $NA_y$ for Coad and Yourdon’s method, $NA_J$ for Ivar Jacobson’s method.

2. The second metric is denoted as $NAA$ (Average Number of Attributes). $NAA_K$ represents for KRB Seven-Step method, $NAA_L$ for Craig Larman’s method, $NAA_y$ for Coad and Yourdon’s method, $NAA_J$ for Ivar Jacobson’s method.

**Table 3: Goal #3 and it’s metrics and question**

Domain model is the basis for future object oriented design. A class with too many attributes in domain model can be programmed into software class with bigger size. To reduce average size per software class, a method that generates lowest average number of attributes per class could be a good choice.

**Goal#4:**
Identify the best method with respect to the average number of associations. The method that can generate the lowest average number of associations is defined as the best one.

**Questions:**
Under the same requirement specification, which method generates a lower average number of associations?

**Metric:**
1. The number of associations in domain model.
2. The average number of associations per class. This metric can be obtained through:
   
   $$NAS = \frac{NS}{NC}$$

**Metric Notation:**
1. The first metric is denoted as $NS$ (Number of Associations). $NS_K$ represents for KRB Seven-Step method, $NS_L$ for Craig Larman’s method, $NS_y$ for Coad and Yourdon’s method, $NS_J$ for Ivar Jacobson’s method.
2. The second metric is denoted as $NAS$ (Average Number of Associations). $NAS_K$ represents for KRB Seven-Step method, $NAS_L$ for Craig Larman’s method, $NAS_y$ for Coad and Yourdon’s method, $NAS_J$ for Ivar Jacobson’s method.

**Table 4: Goal #4 and it’s metrics and question**

When a class with many associations in domain model is programmed into software class, it could ask for lots of additional work to change the corresponding software class. When this software class needs to be modified, many other software classes could require modifications as well. To reduce average number of associations per class, a method that generates lowest average number of associations could be a good choice.

**Goal#5:**
Identify the best method with respect to the sum of DIT (depth of inheritance) value. A method that generate minimum sum of DIT value is defined as the best one. The DIT value for a class within an inheritance structure is the longest path from this class to the root of the inheritance structure [7].

**Questions:**
Under the same requirement specification, which method generates lower sum of DIT value?

**Metric:**
The sum of DIT value in the domain model. $SD$ can be obtained by summing up the DIT value
for each class in domain model.

**Metric Notation:**
The metric is denoted as $SD$ (Sum of DIT Value). $SD_K$ represents for KRB Seven-Step method, $SD_L$ for Craig Larman’s method, $SD_Y$ for Coad and Yourdon’s method, $SD_I$ for Ivar Jacobson’s method.

<table>
<thead>
<tr>
<th>Table 5: Goal #5 and it’s metric and question</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of inheritance in domain model is an important factor that determines the readability of domain model. Too many inheritances will make domain model hard to read and understand. If the seventh goal is achieved, we can find a method that generate the minimum number of inheritances and employ this method to create a domain model with the best readability. [1]</td>
</tr>
</tbody>
</table>

### 5.2 Case study operation

In this section, this case study’s overall operation process is described. In addition, several threats to the validity of this case study are discussed as well.

#### 5.2.1 Process

The GQM method is used in the course of this case study. Application of GQM method divides measurement into four phases: the planning phase, the definition phase, the data collection phase and the interpretation phase. [5]

In the planning phase, three example projects are selected and corresponding requirement specifications are designed and created. In this thesis, each requirement specification contains several use cases and all use cases are written in the fully dress format, which could display use cases in the most elaborate and detailed way [4]. Two example projects are taken from [1] and [6], and one project is designed especially for the purpose of the case study. During the creation of requirement specifications, several personal assumptions are made. Those assumptions can be referenced in the part 8.3.1.

In the definition phase, goals, questions, metrics are defined and documented. The defined software metrics are good measures of certain features of domain model and acts as the basis for collecting data and identifying correct answers for questions in a measurable way [5].

All of data are collected in the data collection phase. Every method is applied to each example project’s requirement specification to produce corresponding set of candidate classes and domain model. To each example project’s requirement specification, there is four domain models generated and 12 domain models is created for all example projects. The domain model is drawn under assistance of the tool called TAU G2. TAU G2’s information can be acquired from the [8].

During the interpretation phase, the data collected in the data collection phase is processed into measurement results. These results are then analyzed further so that questions can be answered and corresponding goals can be achieved as well. To be objective, all analyses are based on the measurement results of this case study. The conclusions are either relationship between methods or the selection of best method. The conclusions are not aimed for applying all projects; they are aimed for the majority of real-world projects.
5.2.2 Threats to this case study

The conclusions of this thesis are aimed for applying every kind of project. However, only three example projects are experimented in this case study. Due to the limited number and types of projects, the accuracy and reliability of all conclusions could be reduced greatly. This is the biggest threat to the validity of this case study. To minimize the threat, three example projects belonging to the mainstream of industry software are employed.

In this thesis, many designed goals affect each other. However, the fact that many goals affect each other does not make threat to the conclusions of this thesis. In the part of results and analyses, each conclusion is corresponding to one goal instead of combinations of several goals; the influences between goals do not threaten the conclusions.

Only one person is engaged in this case study, this is the second threat to the validity of this case study. Different people may produce variant domain models, and this could cause different comparison results. To obtain more reliable and accurate conclusions, enough people should be engaged in this case study.

There are several other threats as well. For example, the methods may not be understood clearly, some mistakes may exist in the domain models, and measurement results may be documented incorrectly. To minimize these threats, methods, domain models and measurement results have been checked several times.

5.3 Results and analyses

In the first part of this section, measurement results for all three requirement specifications are presented in three tables and each table records measurement results for one requirement specification. In the second part, the measurement results are reorganized into seven tables, and each table records measurement results for one software metric. Furthermore, several statistical techniques are applied to summarize data; the techniques include median value, box-plot. The measurement results are analyzed and goals are achieved in the second part as well.

5.3.1 Measurement results for all requirement specifications

In the following, measurement results are presented in three tables and each table records results for one requirement specification.

<table>
<thead>
<tr>
<th>Number of candidate Classes</th>
<th>Coad and Yourdon’s method</th>
<th>Ivar Jacobson’s method</th>
<th>KRB Seven-Step method</th>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of classes</td>
<td>18</td>
<td>16</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Total number of attributes</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>Average number of attributes</td>
<td>1.38</td>
<td>1.38</td>
<td>1.29</td>
<td>1.6</td>
</tr>
<tr>
<td>Total number of</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>associations</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>22</td>
</tr>
<tr>
<td>-------------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>Average number of associations</td>
<td>1.15</td>
<td>1.15</td>
<td>1.14</td>
<td>1.47</td>
</tr>
<tr>
<td>The sum of DIT value</td>
<td>8</td>
<td>8</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

**Table 6: Measurement results for Weather monitoring station system**

<table>
<thead>
<tr>
<th></th>
<th>Coad and Yourdon’s method</th>
<th>Ivar Jacobson’s method</th>
<th>KRB Seven-Step method</th>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of candidate Classes</td>
<td>12</td>
<td>8</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Number of classes</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>Total number of attributes</td>
<td>31</td>
<td>34</td>
<td>34</td>
<td>49</td>
</tr>
<tr>
<td>Average number of attributes</td>
<td>3.86</td>
<td>3.78</td>
<td>3.4</td>
<td>2.88</td>
</tr>
<tr>
<td>Total number of associations</td>
<td>13</td>
<td>14</td>
<td>16</td>
<td>27</td>
</tr>
<tr>
<td>Average number of associations</td>
<td>1.63</td>
<td>1.56</td>
<td>1.60</td>
<td>1.59</td>
</tr>
<tr>
<td>The sum of DIT value</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>15</td>
</tr>
</tbody>
</table>

**Table 7: Measurement results for Registration and title system**

<table>
<thead>
<tr>
<th></th>
<th>Coad and Yourdon’s method</th>
<th>Ivar Jacobson’s method</th>
<th>KRB Seven-Step method</th>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of candidate Classes</td>
<td>10</td>
<td>8</td>
<td>34</td>
<td>20</td>
</tr>
<tr>
<td>Number of classes</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>Total number of attributes</td>
<td>24</td>
<td>27</td>
<td>27</td>
<td>42</td>
</tr>
<tr>
<td>Average number of attributes</td>
<td>3</td>
<td>3</td>
<td>2.7</td>
<td>2.47</td>
</tr>
<tr>
<td>Total number of associations</td>
<td>10</td>
<td>11</td>
<td>13</td>
<td>19</td>
</tr>
<tr>
<td>Average number of associations</td>
<td>1.25</td>
<td>1.22</td>
<td>1.30</td>
<td>1.12</td>
</tr>
<tr>
<td>The sum of DIT value</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

**Table 8: Measurement results for Manage system for DVD shop**

### 5.3.2 Statistical summary and analyses

In this section, measurement results are rearranged into seven tables, and each table records measurement results for one designed metric. Two statistical techniques are applied to summarize data; the techniques are median value and box-plot. The technique of median value is used to identify average value. Box-plot is used to show how the data are
distributed. During analyzing measurement results for each metric, three inequalities are generated by comparing results. Based on the inequalities, analyses are presented and corresponding goals are achieved.

1. Analyses related with goal#1

a. Statistical summary in table

<table>
<thead>
<tr>
<th>System</th>
<th>Coad and Yourdon’s method</th>
<th>Ivar Jacobson’s method</th>
<th>KRB Seven-Step method</th>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather monitoring station system</td>
<td>18</td>
<td>16</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Registration and Title system</td>
<td>12</td>
<td>8</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Manage System for DVD shop</td>
<td>10</td>
<td>8</td>
<td>34</td>
<td>20</td>
</tr>
<tr>
<td>Median value</td>
<td>12</td>
<td>8</td>
<td>34</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 9: Measurement results for the metric of number of candidate classes

b. Box-plot

![Figure 5: Box-Plot for the number of candidate classes](image)

c. Inequality formula

- Weather monitoring station system: $\text{NCC}_K > \text{NCC}_Y > \text{NCC}_J > \text{NCC}_L$
- Registration and Title system: $\text{NCC}_K > \text{NCC}_L > \text{NCC}_Y > \text{NCC}_J$
- Manage System for DVD shop: $\text{NCC}_K > \text{NCC}_L > \text{NCC}_Y > \text{NCC}_J$

d. Answers to questions and corresponding analyses

From the above three inequalities, we can find that KRB Seven-Step method can generate maximum number of candidate classes. The order between other methods related with the number of candidate class is dependent on the requirement specification. However, on average Craig Larman’s method can produce more candidate classes than Coad and
Yourdon’s method and Ivar Jacobson’s method, and Coad and Yourdon’s method can produce more classes than Ivar Jacobson’s method.

The reason that KRB Seven-Step method may generate maximum number of candidate classes may be that the amount of nouns in use cases is usually very large; this can be reflected from the measurement results of this case study. The large amount of nouns could make KRB Seven-step method generate maximum number of candidate classes. In addition, we cannot say a method with more candidate class categories will generate more candidate classes, that is why the order between Coad and Yourdon’s method, Ivar Jacobson’s method, Craig Larman’s method is dependent on requirement specification.

e. Goal achievement

KRB Seven-Step method can generate maximum number of candidate classes, there seems to be no relationship based on the gathered data between other three methods. However, on average Craig Larman’s method can produce more candidate classes than Coad and Yourdon’s method and Ivar Jacobson’s method, and Coad and Yourdon’s method can produce more classes than Ivar Jacobson’s method.

2. Analyses related with the goal #2

a. Statistical summary in table

<table>
<thead>
<tr>
<th></th>
<th>Coad and Yourdon’s method</th>
<th>Ivar Jacobson’s method</th>
<th>KRB Seven-Step method</th>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather monitoring station system</td>
<td>13</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>Registration and Title system</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>Manage System for DVD shop</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>Median value</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 10: Measurement results for the metric of number of classes

b. Box-Plot

Figure 6: Box-Plot for the number of classes
c. **Inequality formula**

Weather monitoring station system: \( NC_L > NC_K > NC_J = NC_Y \)
Registration and Title system: \( NC_L > NC_K > NC_J > NC_Y \)
Manage System for DVD shop: \( NC_L > NC_K > NC_J > NC_Y \)

**d. Answers to the questions and corresponding analyses**

From the three inequalities above, we can find that Craig Larman’s method can generate maximum number of classes, Coad and Yourdon’s method generate minimum number of classes, and KRB Seven-Step method generates more classes than Ivar Jacobson’s method. This relationship between methods can also be reflected on the median value.

There could be several reasons contributing to this. First, these four methods can generate similar set of classes that are essential to system requirements. In other words, the identification of classes is largely dependent on requirement specification instead of specific method or technique. Secondly, Coad and Yourdon’s method and KRB Seven-Step method can capture more super classes than Ivar Jacobson’s method and Coad and Yourdon’s method, this could cause Coad and Yourdon’s method and KRB Seven-Step method have more classes than another two methods. Furthermore, since Craig Larman’s method converts some attributes into classes, this could cause Craig Larman’s method produce more classes than KRB Seven-Step method as well.

**e. Goal achievement**

Since Coad and Yourdon’s method produce minimum number of classes for three requirement specifications, it is the best method with respect to the metric of number of classes.

3. **Analyses related with goal #3**

a. **Statistical summary in table**

<table>
<thead>
<tr>
<th></th>
<th>Coad and Yourdon’s method</th>
<th>Ivar Jacobson’s method</th>
<th>KRB Seven-Step method</th>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather monitoring station system</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>Registration and Title system</td>
<td>31</td>
<td>34</td>
<td>34</td>
<td>49</td>
</tr>
<tr>
<td>Manage System for DVD shop</td>
<td>24</td>
<td>27</td>
<td>27</td>
<td>42</td>
</tr>
<tr>
<td>Median value</td>
<td>24</td>
<td>27</td>
<td>27</td>
<td>42</td>
</tr>
</tbody>
</table>

**Table 11: Measurement results for the metric of number of attributes**

<table>
<thead>
<tr>
<th></th>
<th>Coad and Yourdon’s method</th>
<th>Ivar Jacobson’s method</th>
<th>KRB Seven-Step method</th>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather monitoring station system</td>
<td>1.38</td>
<td>1.38</td>
<td>1.29</td>
<td>1.6</td>
</tr>
<tr>
<td>Registration and</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 12: Measurement results for the metric of average number of attributes

<table>
<thead>
<tr>
<th>Model</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title system</td>
<td>3.86</td>
<td>3.78</td>
<td>3.4</td>
<td>2.88</td>
</tr>
<tr>
<td>Manage System for DVD shop</td>
<td>3</td>
<td>3</td>
<td>2.7</td>
<td>2.47</td>
</tr>
<tr>
<td>Median value</td>
<td>3</td>
<td>3</td>
<td>2.7</td>
<td>2.47</td>
</tr>
</tbody>
</table>

b. Box-Plot

![Figure 7: Box-Plot for the number of attributes](image)

![Figure 8: Box-Plot for the average number of attributes](image)

c. Inequality formula related with number of attributes

Weather monitoring station system: \( NA_L > NA_k = NA_J = NA_Y \)

Registration and Title system: \( NA_L > NA_k = NA_J > NA_Y \)

Manage System for DVD shop: \( NA_L > NA_k = NA_J > NA_Y \)

d. Inequality formula related with average number of attributes

Weather monitoring station system: \( NAA_L > NAA_k = NAA_J > NAA_Y \)
e. **Answers to the questions and corresponding analyses**

From the three inequalities related with average number of attributes, we can find that the order between all methods related with average number of attributes is dependent on requirement specification. However, on average Coad and Yourdon’s method and Ivar Jacobson’s method produce highest average number of attributes, and Craig Larman’s method produce lowest average number of attributes.

By comparing the domain models in appendix, we can find the common classes in different domain models have same set of attributes. Therefore, we can say the four methods have a similar attribute identification technique. To any common class, the identification of its’ attributes is dependent on specific requirement specification, different methods cannot generate different sets of attributes. In addition, we can find that the relationship between methods with respect to the number of attributes is similar to the relationship regarding to the number of classes, therefore we can say that method generating more classes will not produce less attributes.

We can obtain the value of $NAA$ (Average number of attributes) through the calculation: $NAA = \frac{Na}{NC}$. The relationship between methods with respect to $Na$ (Number of attributes) is similar as the relationship regarding $NC$ (Number of classes) and this similarity could cause the order between all methods regarding the division of $\frac{Na}{NC}$ uncertain and dependent on requirement specification. Therefore, the relationship between methods with respect to the average number of attributes cannot be identified.

f. **Goal achievement**

Since there seems to be no relationship between methods with respect to this metric based on gathered data, the best method cannot be identified. However, on average Craig Larman’s method is the best method with respect to this metric.

4. **Analyses related with goal#4**

a. **Statistical summary in table**

<table>
<thead>
<tr>
<th></th>
<th>Coad and Yourdon’s method</th>
<th>Ivar Jacobson’s method</th>
<th>KRB Seven-Step method</th>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather monitoring</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>22</td>
</tr>
<tr>
<td>station system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registration and</td>
<td>13</td>
<td>14</td>
<td>16</td>
<td>27</td>
</tr>
<tr>
<td>Title system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage System for DVD shop</td>
<td>10</td>
<td>11</td>
<td>13</td>
<td>19</td>
</tr>
<tr>
<td>Median value</td>
<td>13</td>
<td>14</td>
<td>16</td>
<td>22</td>
</tr>
</tbody>
</table>

*Table 13: Measurement results for the metric of number of associations*
Table 14: Measurement results for the metric of average number of associations

<table>
<thead>
<tr>
<th>System</th>
<th>Coad and Yourdon’s method</th>
<th>Ivar Jacobson’s method</th>
<th>KRB Seven-Step method</th>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather monitoring station system</td>
<td>1.15</td>
<td>1.15</td>
<td>1.14</td>
<td>1.47</td>
</tr>
<tr>
<td>Registration and Title system</td>
<td>1.63</td>
<td>1.56</td>
<td>1.60</td>
<td>1.59</td>
</tr>
<tr>
<td>Manage System for DVD shop</td>
<td>1.25</td>
<td>1.22</td>
<td>1.30</td>
<td>1.12</td>
</tr>
<tr>
<td>Median value</td>
<td>1.25</td>
<td>1.22</td>
<td>1.30</td>
<td>1.47</td>
</tr>
</tbody>
</table>

b. Box-Plot

Figure 9: Box-Plot for the number of associations

Figure 10: Box-Plot for the average number of associations
c. Inequality formula related with number of associations
Weather monitoring station system: \( NS_L > NS_K > NS_J = NS_Y \)
Registration and Title system: \( NS_L > NS_K > NS_J > NS_Y \)
Manage System for DVD shop: \( NS_L > NS_K > NS_J > NS_Y \)

d. Inequality formula related with average number of associations

Weather monitoring station system: \( NAS_L > NAS_J = NAS_Y > NAS_K \)
Registration and Title system: \( NAS_Y > NAS_K > NAS_L > NAS_J \)
Manage System for DVD shop: \( NAS_L > NAS_K > NAS_Y > NAS_J \)

e. Answers to the questions and corresponding analyses

From the three inequalities related with average number of associations, we can find that the order between all methods related with the average number of associations is dependent on requirement specification. However, on average Craig Larman’s method generates highest average number of associations, KRB Seven-Step method produce second highest. Coad and Yourdon’s method produces higher average number of associations than Ivar Jacobson’s method.

From the case study, we can find that all methods can generate similar set of associations that are within system responsibilities. Therefore, the identification of associations is largely dependent on requirement specification instead of specific method or technique. Furthermore, from the inequalities related with number of associations, we can find the relationship between all methods with respect to the number of associations is the same as the relationship regarding the number of classes, therefore we can say that method generates more classes will produce more associations, and method generates less classes will produce less associations.

We can obtain the value of \( NAS \) (Average number of associations) through the calculation: \( NAS = \frac{NS}{NC} \). The relationship between methods with respect to \( NS \) (Number of associations) is similar as the relationship regarding \( NC \) (Number of classes) and this similarity could cause the order between all methods regarding the division of \( \frac{NS}{NC} \) uncertain and dependent on requirement specification. Therefore, the relationship between methods with respect to average number of associations cannot be identified.

f. Goal achievement

Since there seems to be no relationship between methods with respect to this metric based on gathered data, the best method cannot be identified. However, on average Ivar Jacobson’s method is the best method with respect to this metric.

5. Analyses related with goal#5

a. Statistical summary in table

<table>
<thead>
<tr>
<th>Method</th>
<th>Coad and Yourdon’s</th>
<th>Ivar Jacobson’s method</th>
<th>KRB Seven-Step method</th>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

26
method

| Weather monitoring station system | 8 | 8 | 13 | 13 |
| Registration and Title system | 4 | 4 | 12 | 15 |
| Manage System for DVD shop | 3 | 3 | 6 | 9 |
| Median value | 4 | 4 | 12 | 13 |

Table 15: Measurement results for the metric of sum of DIT value

b. Box-Plot

![Box-Plot for the sum of DIT value](image)

Figure 11: Box-Plot for the sum of DIT value

c. Inequality formula

Weather monitoring station system: \[ SD_L = SD_K > SD_I = SD_J \]
Registration and Title system: \[ SD_L > SD_K > SD_I = SD_J \]
Manage System for DVD shop: \[ SD_L > SD_K > SD_I = SD_J \]

From the three inequalities above, we can find that KRB Seven-Step method and Craig Larman’s method can produce higher sum of DIT value than another two methods, and Craig Larman’s method can produce higher or equal sum of DIT value than KRB Seven-Step method. Coad and Yourdon’s method and Ivar Jacobson’s method have the same sum of DIT value. This relationship between methods can also be reflected on the median value.

In KRB Seven-Step method and Craig Larman’s method, several classes that have common attributes and associations can generate one super class, while in Coad and Yourdon’s method and Ivar Jacobson’s method, super class can be created in the condition that several classes have common attributes, and classes that have common associations cannot generate corresponding super class. Therefore, KRB Seven-Step method and Craig Larman’s method can produce more super classes than another two methods, correspondingly they can
generate higher sum of DIT value. Coad and Yourdon’s method and Ivar Jacobson’s method have same inheritance technique and similar set of classes, this could make them produce same sum of DIT value. Craig Larman’s method converts some attributes into classes, and this could increase the number of classes greatly, and this could make it generate higher sum of DIT value than KRB Seven-Step method.

e. **Goal achievement**

Since Coad and Yourdon’s method and Ivar Jacobson’s method produce lowest sum of DIT value for all requirement specifications, they can be the best method with respect to the sum of DIT value.
6 CONCLUSIONS

In this thesis, one relationship between four OOA methods regarding the number of candidate classes has been identified. Among those four methods, one method was defined as the best method with respect to each of the following software metrics: the number of classes, the average number of attributes, the average number of associations and the sum of DIT value. For example, Craig Larman’s method could be the best method with respect to the average number of attributes. The identified best methods and relationship can be referenced in the part 5.3.2. Due to limited number of example projects in the case study, the validity of conclusions is restricted and may not be applied for all real-world projects.

Based on this case study, we can find that all methods generate similar set of classes that are essential to system requirements. In other words, the identification of classes is largely dependent on requirement specification instead of certain method or technique, and this can be applied to the identification of attributes and associations as well. There are several important reasons that could affect or determine the identified relationship and selections of best method. The first reason is that all methods produce a quite different set of candidate classes. The differences between sets of candidate classes can be reflected between the sets of actually needed classes. The second reason is that Craig Larman’s method presented several techniques that could increase the number of classes greatly, for example the technique of converting attributes with separate sections into independent classes. The third reason is that KRB Seven-Step method and Craig Larman’s method present different inheritance identification technique than another two methods.

There are further works that need to be done in the future. In the section 5.3.2, many reasons were presented and discussed. However, those reasons may need to be validated further on the condition that enough number of example projects are employed and enough people are engaged in this case study. There could be several reasons that have not been identified yet; further work may need to be done to discover those reasons as well. Furthermore, how the identified relationship and best methods can be applied to Object Oriented Design phase should be investigated as well.
7 REFERENCES

[1]. Peter Coad and Edward Yourdon, *Object-Oriented Analysis (Second Edition)*
ISBN 0-13-629981-4, Published by Prentice-Hall


8  APPENDIX

8.1  Candidate classes for all requirement specifications

8.1.1  Weather monitoring station system

<table>
<thead>
<tr>
<th>Coad and Yourdon’s method</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Ivar Jacobson’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Screen, Wind Speed Screen, Temperature Screen, Wind Speed Information, Temperature Information, Wind Speed Sensor, Temperature Sensor, Clock, Speed generator, Temperature generator, the button of “Wind Speed Information”, the button of “Current Wind Speed”, the button of “The Highest Wind Speed”, the button of “Temperature Information”, the button of “Current Temperature”, the button of “The Highest Temperature”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KRB Seven-Step method</th>
</tr>
</thead>
<tbody>
<tr>
<td>User, Main Screen, Weather Monitoring Station, Wind Speed Information, the button, Wind speed screen, certain item, corresponding information, Wind speed Measurement, time, hour, wind speed sensor, clock, Temperature information, temperature screen, Temperature Measurement, temperature sensor, Weather Monitoring Station system, reported value, Hardware</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wind speed sensor, clock, temperature sensor, weather monitoring station, user, legal event of checking wind speed information, legal event of checking temperature information, wind speed Information, Temperature information, LCD device</td>
</tr>
</tbody>
</table>
### 8.1.2 Registration and title system

<table>
<thead>
<tr>
<th>Coad and Yourdon’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle information, Truck information, motorcycle information, trailers information, travel trailers information, Customer information, title information, clerk information, Clerk, Customer, owner, shop</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ivar Jacobson’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receipt generator, Clerk, Customer information, Truck information, Title information, Motorcycle information, Trailers information, Travel trailer information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KRB Seven-Step method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clerk, customer, truck, customer information, truck information, title information, receipt, owner, business, title, customer number, title number, surrendered title, title date, fee, vehicle number, legal name of clerk, money, payment, legal name of customer, address of customer, telephone of customer, number, model, make, body style, color, cost, number of passengers, mileage, temporary gross weight, motorcycle, motorcycle information, trailers, trailers information, gross weight, travel trailer, travel trailer’s information, body number, length</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Craig Larman’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truck, motorcycle, trailers, travel trailer, receipt, money, Payment, Clerk, customer, owner, truck information, customer information, motorcycle information, trailers information, travel trailer information, title information, Legal event of registration for motorcycle, Legal event of registration for truck, Legal event of registration for travel trailer, Legal event of registration for trailers</td>
</tr>
</tbody>
</table>

### 8.1.3 Manage system for DVD shop

<table>
<thead>
<tr>
<th>Coad and Yourdon’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rent record, customer information, black list, inventory information, People, Clerk,</td>
</tr>
<tr>
<td><strong>inventory staff, manager, customer, Shop</strong></td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
</tbody>
</table>

**Ivar Jacobson’s method**

Rent information, Customer information, Black list, Inventory staff, Clerk, Manager, Inventory information, Receipt generator

**KRB Seven-Step method**

Clerk, customer, customer information, rent information, inventory information, receipt, counter, DVD, business, movie, customer id, customer name, movie id, movie name, the number of movie, start-date, end-date, return date, money, system, legal name, address, telephone, black list, status, items, the number of items, the corresponding place, shop, inventory staff, Manager, corresponding screen, people, changes

**Craig Larman’s method**

Receipt, counter, DVD, money, Shop, Payment, rent, Clerk, manager, inventory staff, customer, Legal event of Customer rents DVD, legal event of customer returns DVD, legal event of customer searches item information, legal event of inventory staff stocks new items, legal event of manager adds some customers into black list, Rent record, customer information, inventory information, black list
8.2  Domain models for all requirement specifications

8.2.1 Weather Monitoring Station System

Model generated by Coad and Yourdon's method
Model generated by Ivar Jacobson’s method
Model generated by KRB Seven-Step method
Model generated by Craig Larman’s method
8.2.2 Registration and title system

Model generated by Coad and Yourdon’s method
Model generated by Ivar Jacobson’s method
Model generated by KRB Seven-Step method
Model generated by Craig Larman’s method
8.2.3 Manage System For DVD Shop

Model generated by Coad and Yourdon’s method
Model generated by Ivar Jacobson’s method
Model generated by KRB Seven-Step method
Model generated by Craig Larman’s method
8.3 Requirements Specifications

8.3.1 Weather Monitoring Station System

Use Case UC1: Check the Wind Speed Information

Primary Actor: A User of the weather monitoring station
Preconditions: The weather monitoring station system behaves normally.
Success Guarantee (Post conditions): The weather monitoring station display correct wind speed information to user.

Main Success Scenario (or Basic Flow):
1. A user arrives at the weather monitoring station.
2. The user touches the main screen and selects the button of “Wind Speed Information”, and wind speed screen appears.
3. The user selects certain item from screen and corresponding information appears.
4. The user reads the information he/she wants to know.
5. The user click the button of “Back to Main Screen” to return to the Main Screen and leave.

Extensions (or Alternative Flows):
3a. The user selects the button of “Current Wind Speed”:
   1. The current measurement of wind speed and current time is presented.
3b. The user selects the button of “The Highest Wind Speed”:
   1. The highest measurement of wind speed within last 24 hours and the time of the reported value are presented.

Technology and Data Variations List:
3a. The Weather Monitoring Station system uses the hardware of wind speed sensor to get the corresponding wind speed information.
3b. The Weather Monitoring Station system uses the hardware of clock to acquire time.

Use Case UC2: Check the Temperature Information

Primary Actor: A User of the weather monitoring station
Preconditions: The weather monitoring station system behaves normally.
Success Guarantee (Post conditions): The weather monitoring station display correct Temperature information to user.

Main Success Scenario (or Basic Flow):
1. A user arrives at the weather monitoring station.
2. The user touches the main screen and selects the button of “Temperature Information”, and temperature screen appears.
3. The user selects certain item from screen and corresponding information appears.
4. The user reads the information he/she wants to know.
5. The user click the button “Back to Main Screen” to return to the Main Screen and leave.

Extensions (or Alternative Flows):
3a. The user selects the button of “Current Temperature”:
1. The current measurement of temperature and current time are presented.
3b. The user selects the button of “The Highest Temperature”:
   1. The highest measurement of temperature within last 24 hours and the time of the reported value are presented.

**Technology and Data Variations List:**
3a. The Weather Monitoring Station system uses the hardware of temperature sensor to get the corresponding temperature information.
3b. The Weather Monitoring Station system uses the hardware of clock to acquire time.

**Assumptions:**
1. The hardware of clock is used to acquire time.
2. The width and length property of screen must be stored in the system.

### 8.3.2 Registrations and Title System

#### Use Case UC1: Register for trucks

<table>
<thead>
<tr>
<th>Primary Actor:</th>
<th>Clerk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions:</td>
<td>Clerk is identified and authenticated.</td>
</tr>
<tr>
<td>Success Guarantee (Post conditions):</td>
<td>In the case of new customer or new truck, corresponding customer or truck information is created and saved successfully. The title information is saved successfully. The receipt is generated.</td>
</tr>
</tbody>
</table>

**Main Success Scenario (or Basic Flow):**
1. An owner arrives at counter to register for his or her new bought truck.
2. Clerk starts a new business.
3. Clerk validates the customer and truck information.
4. Clerk issues a title for the customer and saves the corresponding information. The title information may include customer number, the title number, surrendered title, title date, fee, vehicle number, and the legal name of clerk.
5. System presents the total money.
6. Clerk tells the owner the total and asks for payment.
7. The owner pays and the system handles the payment.
8. The system generates one receipt.
9. The owner leaves with receipt.

**Extensions (or Alternative Flows):**
3a. There is no the owner’s information in the system.
   1. The clerk enters the owner’s information into the system. The owner information may include legal name, customer number, address, and telephone.
3b. There is no the truck’s information in the system.
   1. The clerk enters the truck’s information into the system. The truck information includes number, model, make, body style, color, cost, number of passengers, mileage, and powered by, temporary gross weight.

#### Use Case UC2: Register for motorcycle


Primary Actor: Clerk
Preconditions: Clerk is identified and authenticated.

Success Guarantee (Post conditions): In the case of new customer or new motorcycle, corresponding customer or motorcycle information is created and saved successfully. The title information is saved successfully. The receipt is generated.

Main Success Scenario (or Basic Flow):
1. An owner arrives at counter to register for his or her new bought motorcycle.
2. Clerk starts a new business.
3. Clerk validates the customer and motorcycle information.
4. Clerk issues a title for the customer and saves the corresponding information. The title information may include customer number, the title number, surrendered title, title date, fee, vehicle number, and the legal name of clerk.
5. System presents the total money.
6. Clerk tells the owner the total and asks for payment.
7. The owner pays and the system handles the payment.
8. The system generates one receipt.
9. The owner leaves with receipt.

Extensions (or Alternative Flows):

3a. There is no the owner’s information in the system.
   1. The clerk enters the owner’s information into the system. The owner information may include legal name, customer number, address, and telephone.
3b. There is no the motorcycle’s information in the system.
   1. The clerk enters the motorcycle’s information into the system. The motorcycle information includes number, model, make, body style, color, cost, number of passengers, mileage.

Use Case UC3: Register for trailers

Primary Actor: Clerk
Preconditions: Clerk is identified and authenticated.

Success Guarantee (Post conditions): In the case of new customer or new trailers, corresponding customer or trailers information is created and saved successfully. The title information is saved successfully. The receipt is generated.

Main Success Scenario (or Basic Flow):
1. An owner arrives at counter to register for his or her new bought trailers.
2. Clerk starts a new business.
3. Clerk validates the customer and trailers information.
4. Clerk issues a title for the customer and saves the corresponding information. The title information may include customer number, the title number, surrendered title, title date, fee, vehicle number, and the legal name of clerk.
5. System presents the total money.
6. Clerk tells the owner the total and asks for payment.
7. The owner pays and the system handles the payment.
8. The system generates one receipt.
9. The owner leaves with receipt.

Extensions (or Alternative Flows):

3a. There is no the owner’s information in the system.
1. The clerk enters the owner’s information into the system. The owner information may include legal name, customer number, address, and telephone.

3b. There is no the trailer’s information in the system.

1. The clerk enters the trailer’s information into the system. The trailers information includes number, model, make, body style, color, cost, and gross weight.

Use Case UC4: Register for travel trailer

Primary Actor: Clerk
Preconditions: Clerk is identified and authenticated.
Success Guarantee (Post conditions): In the case of new customer or new travel trailer, corresponding customer or travel trailer’s information is created and saved successfully. The title information is saved successfully. The receipt is generated.

Main Success Scenario (or Basic Flow):
1. An owner arrives at counter to register for his or her new bought travel trailer.
2. Clerk starts a new business.
3. Clerk validates the customer and travel trailer information.
4. Clerk issues a title for the customer and saves the corresponding information. The title information may include customer number, the title number, surrendered title, title date, fee, vehicle number, and the legal name of clerk.
5. System presents the total money.
6. Clerk tells the owner the total and asks for payment.
7. The owner pays and the system handles the payment.
8. The system generates one receipt.
9. The owner leaves with receipt.

Extensions (or Alternative Flows):

3a. There is no the owner’s information in the system.
   1. The clerk enters the owner’s information into the system. The owner information may include legal name, customer number, address, and telephone.
3b. There is no the travel trailer’s information in the system.
   1. The clerk enters the travel trailer’s information into the system. The travel trailer information includes number, model, make, body style, color, cost, number of passengers, mileage, powered by, body number, length.

8.3.3 Manage system for DVD shop

Use Case UC1: customer rents DVD
Preconditions: Clerk is identified and authenticated.
Success Guarantee (Post conditions): In the case of new customer, corresponding customer information is created and saved successfully. The rent information is saved successfully. The inventory information is updated successfully. The receipt is generated.

Main Success Scenario (or Basic Flow):
1. A customer arrives at counter with one or several DVD to rent.
2. The clerk starts a new business.
3. Clerk validates the customer information.
4. To each movie, clerk creates corresponding rent information. The rent information should include customer id, customer name, movie id, movie name, number of movie, start-date, end-date, return date, and the amount of money.
   Clerk repeats the step 4 until all movies have been handled.
5. The system presents the total money.
6. Clerk tells the customer the total and asks for payment.
7. Customer pays and the payment is handled by system.
8. The system saves all of rent information and the inventory information is updated as well.
9. The system generates one receipt.
10. The customer leaves with receipt and DVD.

Extensions (or Alternative Flows):

3a. There is no the customer’s information in the system.
   1. The clerk enters the customer’s information into the system. The customer information may include legal name, customer id, address, and telephone.
3b. The customer is in the black list (black list means that the customer owes DVD to this shop).
   1. The system will refuse this customer’s business.

Use Case UC2: customer returns DVD

Primary Actor: Clerk
Preconditions: Clerk is identified and authenticated.
Success Guarantee (Post conditions): The status of the rent information is changed to “Return” and saved successfully. The inventory information is updated successfully. If the customer, who is in the black list, returns all the items he/she owes, the customer will be deleted from the black list.

Main Success Scenario (or Basic Flow):
1. A customer arrives at counter with one or several DVD to return.
2. The clerk starts a new business.
3. Clerk validates the rent information for each item.
4. Clerk changes the corresponding rent information’s status to “Return”.
   Clerk repeats the step 3,4 until all rent information is changed.
5. The changes of those rent information are saved and the inventory information is updated as well.
6. The customer leaves.

Extensions (or Alternative Flows):

3a. The customer returns all the items that he/she owes.
   1. The system will delete the customer from black list.

Use Case UC3: Stock new items

Primary Actor: Inventory staff
Preconditions: Inventory staff is identified and authenticated.
Success Guarantee (Post conditions): To those new items that have no inventory information in the system, corresponding inventory information is created and saved.
successfully. To those new items that have already inventory information, corresponding information is updated and saved successfully.

**Main Success Scenario (or Basic Flow):**
1. The shop buys some new DVD.
2. The inventory staff starts a new business.
3. The inventory staff creates or updates inventory information for each new item. The inventory information should include movie id, movie name, place, total number, and the number of movies that are still in stock, the number of movies that has been lent out.
4. The inventory staff saves the above inventory information change.
   The inventory staff repeats step 3, 4 until all items are handled.

**Extensions (or Alternative Flows):**

3a. The new item does not have inventory information.
   1. The customer assigns a place to store this item and creates corresponding inventory information.

3b. The new item already has inventory information.
   1. The inventory staff finds the corresponding inventory information and updates it.

---

**Use Case UC4: Add customers into black list**
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Primary Actor:** Manager

**Preconditions:** Manager is identified and authenticated.

**Success Guarantee (Post conditions):** The changes to black list are saved successfully.

**Main Success Scenario (or Basic Flow):**
1. The manager opens corresponding screen to add customers into black list.
2. The system presents all those customers who owe DVD to manager.
3. Manager selects some or all of these people and put them into black list.
4. System saves those above changes made to black list.