ISSN No. 0976-5697

Volume 3, No. 3, May-June 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Tree Based Approach to Use CRC Cards Efficiently in Software Project Management

Vrushali Yogesh Karkare[1] and Dr. Nileshsingh V. Thakur[2]
Dep.of Computer Science and Engineering
Shri Ramdeobaba College of Engineering and Management
Nagpur, India
vrushaliy.karkare@gmail.com,[1]
thakurnisvis@rediffmail.com[2]

Abstract: In software engineering, CRC (Class-Responsibility-Collaborator) cards are used for object oriented analysis, design and modelling. These CRC cards are used to understand the objects, its responsibilities, that is coupling between the classes and sometimes it is used to adjust the load on the classes. After the designs of these CRC cards, they are used for implementation. This paper presents an approach based on tree for efficient use of CRC cards in software project management. This paper suggests that the CRC cards are not only use for implementation but also for project management, like load management with team members, project scheduling and tracking using a data structure graph and trees. Efficacy of the proposed approach is justified through the analytical results.

Keywords: Software engineering, CRC cards, Software project management, Tree, Graph

I. INTRODUCTION

In software industries, the software development assignments are generally, not only time bound activities but also constrained by some other parameters, for example, cost of the project; complexity of the problem; real time or stand alone system application, etc. Software engineering principles are used in software development. Software engineering [1] is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software. It is the application of engineering to software because it integrates significant mathematics, computer science and practices whose origins are in engineering. It is also defined as a systematic approach to the analysis, design, assessment, implementation, testing, maintenance, and reengineering of software.

One of the important concepts of software engineering is object-oriented software engineering- is an object modeling language and methodology. It also uses other design products similar to those used by object-modeling technique. Object-oriented analysis (OOA) applies object-modeling techniques to analyze the functional requirements for a system. Object-oriented design (OOD) elaborates the analysis models to produce implementation specifications. Class-based modeling is used in object-oriented analysis and design (OOAD). It includes identify-analysis classes by examining the problem statement, use a grammatical parse to isolate potential classes, identify the attributes of each class, and identify operations that manipulate the attributes.

Class-responsibility-collaboration (CRC) cards [1] are used for OOA and OOAD. CRC cards are a way to describe the responsibilities and collaborations between objects in an application. CRC cards are used as a brainstorming technique designed to walk-through use cases in order to create objects, responsibilities, and discover collaborators. Individual CRC cards are used to represent objects. The class of the object can be written at the top of the card, responsibilities listed down the left side, collaborating classes are listed to the right of each responsibility. CRC cards facilitate the software development work. Presently, most of the industries dealing with the object-oriented software development are using the CRC cards for systematic development of software.

This paper presents an approach based on tree [2] evaluation of the CRC card to use it more efficiently for project management. Presented approach move around the cost evaluation of the classes and the relevant methods, based on this, the priorities of the development of the modules can be decided and accordingly the allotment of the modules to the concerned developer team or member.

This paper is organized as follows: section 2 discusses the basic fundamentals of tree, CRC cards, and scheduling. Section 3 consists of the discussion on proposed approach which is based on tree to use CRC cards efficiently for project management. Section 4 discusses the application of proposed approach to the application problem where the analytical result is presented. Discussion on the analytical results is presented in section 5. Finally, section 6 discusses the conclusion and future scope followed by the relevant references.

II. TREE, CRC CARDS, AND SCHEDULING

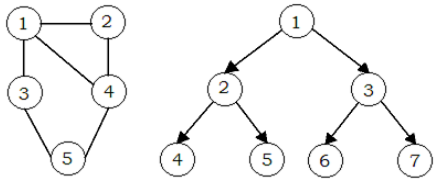
This section presents the discussion on the basics of tree data structure, fundamentals related to CRC cards, and project scheduling and tracking.

A. Tree Data Structure:

A graph [3] data structure consists of a finite (and possibly mutable) set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices. The nodes may be part of the graph structure, or may be external entities represented by integer indices or references. A graph data structure may also

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

associate to each edge, some edge value, such as a symbolic label or a numeric attribute (cost, capacity, length, etc.). Example of graph with five nodes is shown in Fig. 1 (a). A tree is data structure derived from graph and is a way of representing the hierarchical nature of a structure in a graphical forest. Example of tree with seven nodes is shown in Fig. 1 (b). Graphs and trees have a wide range of applications in the field of computer science. Scientifically it can also be used for many other non computer fields for the simulation and modeling of designs.



(a) (b)

Figure 1: (a) Graph and (b) Tree

B. Fundamentals of CRC Cards:

CRC cards were introduced by Kent Beck and Ward Cunningham in their paper "A Laboratory for Teaching Object-Oriented Thinking" released in OOPLSA'89. Their original purpose was to teach programmers the object-oriented paradigm. Although CRC cards were created for teaching, they have proven useful for much more. They have become an accepted method for analysis and design.

A CRC card corresponds to a class. It describes the properties that certain kinds of objects of interest in the problem/ application domain have in common. An object can be a tangible thing, person, place, event, or (abstract) concept. A class should have a single and well-defined purpose that can be described briefly and clearly. It should be named by a noun, noun phrase, or adjective that adequately describes the abstraction. The class name is written across the top of the CRC-card. A short description of the purpose of the class is written on the back of the card.

A responsibility is something the objects of a class take care of; a service provided for other objects. A responsibility can be either to know something or to do something. To do something an object usually uses its (local) knowledge. If this knowledge is not sufficient for the purpose, the class can require help from other objects i.e. its collaborators. The responsibilities of a class are written along the left side of the card.

The collaborators describe which kinds of objects can be asked for help to fulfill a specific responsibility. An object of a collaborator class can for example provide further information, or actually take over parts of the original responsibility (by means of the collaborators own responsibilities). Collaborators are written along the right side of the card in the same row as the corresponding responsibility.

The goal of the CRC-card approach is to develop, discuss and evaluate object-oriented models. In the role-play activities, objects are treated as living entities that fulfill certain responsibilities in the context of the system to be developed. The general format of CRC card and example of CRC card for medical application is shown in Fig. 3 (a) and Fig. 3 (b) respectively.

Class: Class Name Description:	
Responsibilities	*Collaborators*

(a)

Class: Patient Description: A person receives or received medical care	
Responsibilities	*Collaborators*
Make Appointment	Appointments
Get Last Visit	
Change Status	
Provide Medical History	Medical History

(b)

Figure 3: (a) General format of CRC Card and (b) Example of CRC Card

C. Project Scheduling and Tracking:

A project schedule [1] is required to ensure that required project commitments are met. A schedule is required to track progress toward achieving these commitments.

In order to make a schedule, the following tasks must be completed:

a. Identify manageable activities and tasks by decomposing the process and the product.
b. Determine which tasks are dependent on the completion of others (which activities must occur in sequence and which can occur concurrently).
c. Allocate each task a number of work-units (often person-days), a start date and a completion date.
d. Define responsibilities for the tasks (allocate them to a person or persons).
e. Define outcomes of the tasks (deliverables) and milestones for the schedule.
f. Review the proposed tasks, their effort allocation and start and end dates with the people involved to ensure there are no conflicts and over allocation.

III. PROPOSED APPROACH

A CRC cards are used for various purposes. They are used to understand the system as a whole in an object oriented manner. In presented approach, the CRC cards are considered for the purpose of task distribution to the team members and scheduling and tracking the project progress. Proposed

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

approach is based on the tree formation from the graph of the CRC card. The decision of the task allotment is based on the weights or costs associated with the relevant classes and methods. General steps of the approach are given in next subsection.

A. Steps of An Approach:

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

Step 1: Get CRC Cards.
Step 2: Create graph of components with their weights, here weight of component is the number of classes in that component.
Step 3: Get Breadth First Search (BFS) [4, 5] of this graph.
Step 4: Create tree of this graph, if the tree is not binary, then convert it to binary.
Step 5: Get post-order of this graph, this post-order is the sequence of implementation of modules considering their dependency.
Step 6: Consider first component in this sequence and create graph of its classes with their weights, here weight is the number of methods in that class.
Step 7: In the same way, get BFS of this graph, Get binary tree and get post-order.
Step 8: This post-order is the sequence of implementation of classes.
Step 9: Consider this sequence and weights of classes and accordingly allot the task of implementation of classes to team members.
Step 10: Repeat the process for other modules.

IV. APPLICATION OF PROPOSED APPROACH

A. Example:

Consider the example of implementation of five classes for certain software project assignment. CRC Cards for five classes are shown in Fig. 4 with their responsibilities and collaborators.

Class : 101	
Responsibilities	Collaborators
101a	
101b	102
101c	
101d	103

Class : 102	
Responsibilities	Collaborators
102a	
102b	104
102c	103

Class : 103	
Responsibilities	Collaborators
103a	
103b	
103c	105

Class : 104	
Responsibilities	Collaborators
104a	
104b	

Class : 105	
Responsibilities	Collaborators
105a	
105b	

Figure 4: CRC Cards for five classes, their responsibilities and collaborators

B. Explanation:

To finish the task of implementation of class 101, the responsibilities 101b and 101d have to be accomplished first. To accomplish these responsibilities, it needs the help of classes 102 and 103. So these classes must be completed first. Again to finish work on class 102, its responsibilities 102b and 102c must be implemented first. They are depending on classes 103 and 104 respectively.

If the task of all classes is given to five team members, five classes are distributed. That is one team member is given a single class. Now person 1 can implement responsibility 101a and 101c, but has to wait till person 2 finishes class 102 and person 3 finishes class 103. Also person 2 has to wait till person 3 finish.

Formulation of graph is possible by using CRC cards given in Fig. 4. As a CRC card depends on other classes to accomplish some of its responsibilities, there is a dependency between the CRC cards and this dependency should be followed at the time of implementation of the designs. To formulate a graph by using CRC cards, the classes, responsibilities and collaborators all will become nodes. Now, consider classes as a node, then its responsibilities become its children and the responsibilities which are having collaborators, these collaborator classes will become children of these responsibilities node. The formulated graph should be a directed graph to show the dependency between the nodes. It is like, class depends on its responsibilities and some responsibilities depend on collaborators. Again these collaborators are the classes which are having their own responsibilities and collaborators. So the graph grows further in the same manner as, the responsibilities are children on class and some of the children are having collaborators as their child.

In formulation of graph, it seems that it will become a tree, as it shows the dependencies. But as per the definition of tree, a node can have at most one parent and in this case it is observed that 101d depends on 103 and also 102c depends on 103. And this scenario is very well possible in formulation of a graph for certain application problem. Now, node 103 has two parents 101d and 102c, which is violating the definition of tree. So it will not form a tree, it is simply a directed graph. Fig. 5 shows the directed graph created by using CRC cards of example given in Fig. 4.

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

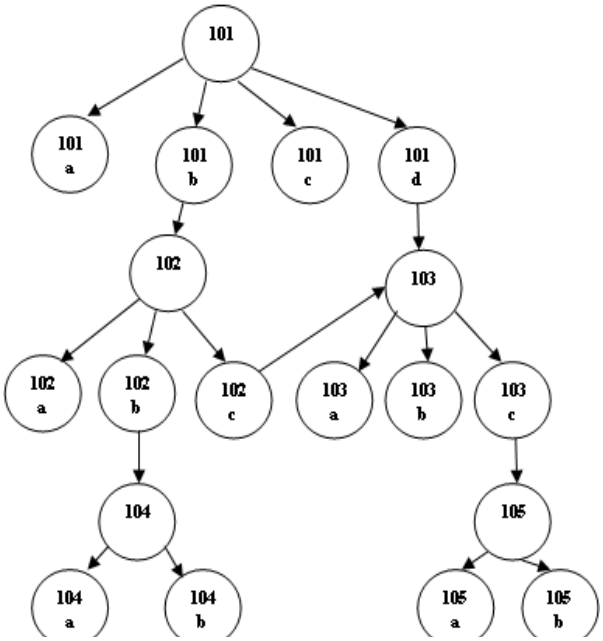


Figure 5: A Graph formed using CRC Cards

In the directed graph shown in Fig. 5, the node 103 has 2 parents, namely 102c and 101d. This is because both nodes 102c and 101d depend on node 103 for some functionality. So this graph is not a tree.

Now, the graph in Fig. 5 seems to be complex and the complexity cane be reduced by considering only classes of CRC cards and putting a node weight to the class nodes. Fig. 6 shows the simplified graph with node weights. Node weight is the number of responsibilities a class has to perform, for example, a class 102 has to perform 4 responsibilities, so weight of this class is 4. Classes 102 and 103 have to perform 3 responsibilities so node weight is 3. Likewise, for class 104 and 105 weight is 2.

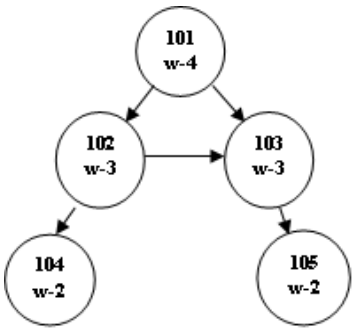


Figure 6: Simplified graph with node weight

To convert a graph shown in Fig. 6 into tree, the closed area should be removed. To remove the closed area, take a breadth first search (BFS) of the graph. Then, BFS of the graph is- 101, 102, 103, 104, and 105. Constructed tree using this BFS is shown in Fig. 7. This tree is constructed considering the order of nodes in BFS and not considering the edges which are forming the closed regions that is the edge which is forming another parent to any particular node should be removed.

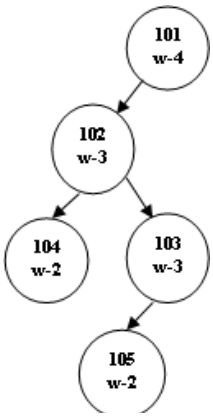


Figure 7: Tree formed from graph

The post-order of tree shown in Fig. 7 is- Post-order = 104, 105, 103, 102, and 101. By observing the order got in post-order, and comparing it with the original graph, then, it follows that the obtained order in post-order can be used while implementation and it will be beneficial.

As five persons are working on five classes, then person 1 working on class 101 has to wait till person 3, working on class 103, completes his/her work. So instead of allotting five classes to five persons, allot only first two classes as per the post-order to two persons. After finishing work on classes 104 and 105, allot classes 103 and 102 again to these two persons, and after completing work on 103 and 102, finally class 101 is allotted and finished.

V. DISCUSSION

With the use of proposed approach, less man power and no waiting time is needed to complete the work. Whenever any class is considered for implementation then it is started and finished as its dependencies are already completed and so all the responsibilities can be completed straightforward without any waiting.

The proposed approach can be used in Rapid Application Development, where time required for completion of project is very crucial parameter. To finish the project fast, then there should not be any dependencies and waiting time because of dependencies. If work is allotted just by considering the number of classes, then this is not fair, because some classes are having many responsibilities and some classes are having less. By using the node weights, the tasks among team members can be easily distributed. Maximum number of responsibilities a class can have depends on the company policy and designs. For example, consider given 5 classes then maximum responsibility is available in class 101, so this class should be given to single person with no other load. Classes 104 and 105 are having fewer loads with total load of 4, so that can be given to single person. So by using the proposed approach, the load management and team management can be performed actively and fairly.

Scheduling and tracking can be performed by using the sequence in post-order. It is obvious that if the implementation of class 101 is finished then the implementation of all the classes is over and it can be treated as 100 % completed. If

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

implementation up to class 103 is finished then it can be claimed that 60 % work is finished.

VI. CONCLUSION AND FUTURE SCOPE

Generally the CRC cards are designed for class based modeling and are used for understanding the classes and use them for implementation. Presented paper focuses on the use of CRC cards not only for the understanding and implementation but also for project management. This project management can be done in terms of load management to team members, project tracking and scheduling. The proposed approach is very much useful for rapid application development type of projects and the projects having less team size. Since many times the team members only wait for others to finish because of dependency, by using the proposed approach, this waiting time can be reduced or there can be no waiting time at all. In future, some metrics can be developed for the performance evaluation of team members. Apart from this, the simulation of the proposed approach can be carried out for the real software development problems.

VII. REFERENCES

[1]. R. Pressman, "Software Engineering: A Practitioner's Approach", 7th edition, McGraw-Hill, 2009.

[2]. Pai, "Data Structures and Algorithms: Concepts, Techniques and Applications", Tata McGraw-Hill Education, 2008.

[3]. V. Aho, J. D. Ullman, and J. E. Hopcroft, "Data Structures and Algorithms", Addison Wesley, 1983.

[4]. Brassard and P. Bratley, "Fundamentals of Algorithms", PHI Learning, 1996.

[5]. Brassard and P. Bratley, "Algorithmics: Theory and Practice", Prentice Hall, 1988.

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html