A PATTERN LANGUAGE FOR CRC CARDS

Mohamed Fayad[1], Huáscar Sánchez[2], and Haitham Hamza[3]

[1,2]Computer Engineering Dept., College of Engineering, San Jose State University
One Washington Square, San Jose, CA 95192-0180
m.fayad@sjsu.edu[1], hsanchez@email.sjsu.edu[2]

[3]*Computer Science and Engineering Dept., University of Nebraska-Lincoln*
Lincoln, NE 68588, USA
Ph: +1 402 4729492
hhamza@cse.unl.edu[3]

ABSTRACT

The Class Responsibility Collaborator (CRC) cards are index cards that are utilized for mapping candidates classes in predefined design scenarios; e.g. Use Case Scenarios. The objective of CRC cards is to facilitate the design process while insuring an active participation of involved designers. This paper represents the first attempt towards a CRC card pattern language representation via stable patterns as a mean to discover, organize, and utilize CRC cards endured knowledge. Each stable pattern focuses on a distinctive activity and provides a way by which this activity can be conducted efficiently. The pattern language is a continuation of our early effort in improving the effectiveness of CRC cards and their role in the design process.

1. INTRODUCTION:

The notion of CRC-Cards was first introduced in 1989 at the annual OOPSLA conference [2]. The acronym CRC stands for Class, Responsibilities, and Collaboration and, while they are not formally used in UML, they can offer valuable insights during the early stages of development [10]. They are primarily used as a brainstorming technique to rapidly and thoroughly explore design alternatives by identifying the classes and their associations within a system.

Class-Responsibility-Collaborator (CRC) Cards are index cards utilized for mapping candidates classes in predefined design scenarios; e.g. Use Case Scenarios. They provide a simple alternative for a collaborative design environment, where analysts, designers, and developers try to simulate the system behavior; i.e. role-play-driven approach. This process ends up with a set of collaborative classes represented by index cards, along with their roles, which are played by the members of the development team in a pre-animated design scenario [6]. Figure 1 shows the original CRC cards, and illustrates in light gray color current changes incurred over its original format.

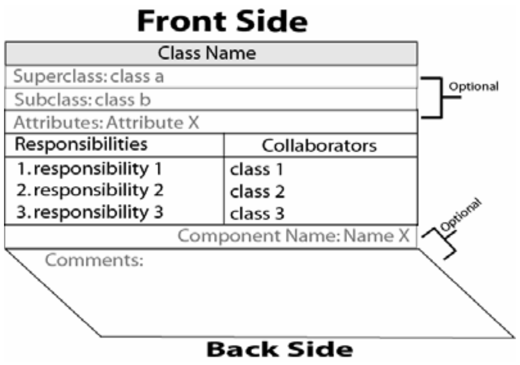

Figure 1: Original CRC card and current changes in its structure

[HS: They recommended including the original CRC card format. I included this one. What do you think?]

Aside from its original purpose that was to teach programmers the Object Oriented Paradigm, CRC Cards have been redefined to become valuable beyond the educational purpose [2]. For their simplicity and flexible form, this tool can be applied to different domains' purposes, such as teaching OOP, a mean for not only documenting and identifying relevant classes of a system, serving as a methodology or as a front-end for other design methods, but also solving modeling problems, and engaging the entire development team through effective brainstorming sessions, etc.

Some of those applications of CRC cards, at such, specify the underlying goals and expectations of CRC Cards use. The processes of accomplishing these goals, especially in software development, throughout iterative sessions are done at ad-hoc. Not precisely knowing when, how, and where to apply them to successfully accomplish the expected goals. This limitation calls software practitioners for a set of suitable guidelines and stable knowledge to answer those former inquiries. Being this the rationale of this work; to propose the first pattern language for CRC cards and communicate this stable knowledge and suitable guidelines. This pattern language will enhance the way we commonly see and use CRC cards; turning it from simple index cards to a valuable knowledge repository. Our idea of this paper is not to provide a specific approach on how to define and deploy a CRC card, instead we provide a straightforward conceptualization and understanding of the CRC Card's domain knowledge, and fundamental processes to deploy one. To do this we relied on the formed synergy of two methodologies: the Software Stability Concepts paradigm [5] and its sub-elements: Stable Analysis and Design Patterns [5], and Pattern Language Methodology [18]. The reader, in this case, would truly visualize the distinct elements that composed the CRC Card's domain knowledge, and how they relate with each other to cope with a determined area of application.

The rest of the paper is organized as follows. Section 2 provides an overview of the essential characteristics of effective CRC Cards. Section 3 focuses on a bird-eye description of the CRC Card implicit goals and Capabilities. Section 4 discusses the rationale of the Pattern Language for CRC Cards and the set of stable patterns involved in the CRC Card usage. Section 5 provides the detailed description of CRC-Cards pattern language. Some conclusions and discussion are presented in Section 6.

2. What Makes an Effective CRC Cards?

CRC Cards regardless of which format is used [19] embodies a particular set of characteristics that transcend across any context of applicability. Each of these characteristics possesses essential semantics that must be taken into consideration when applying CRC cards across domains. The focusing on those semantics, along with the utilization of CRC Card quality factors would be key factors in improving CRC card utilization within software development life cycle.

Regardless of structure's simplification, current CRC-Cards follow the original structure and share same basic elements (i.e. Class name, Responsibility, and Collaborators). Nevertheless, it may include more elements if necessary [19]. In the long run, these current structures might not always attain the essential aspects needed in future development stages (i.e., building system class diagrams, etc).Therefore, for CRC-cards to aid system development, main essential quality factors need to be satisfied. The fulfillment of these quality factors will have a high impact on the CRC Card's characteristics realization. These quality factors are provided herein:

(1) *High Level of Understandability*: Illustrate its sections in an orderly and specific manner; showing an efficient distribution of its elements.
(2) *Accurate Identification of Class Elements*: Assure proper *Identification* of artifact/class and its elements. This will prevent any confusion during class assignation.
(3) *Well-defined Role*: Include a well-defined role, within context, for the artifact/class being developed. This role has to strongly reference the artifact's assigned responsibility. Each class may have multiple roles according to a specific simulated design scenario (Humans for example).
(4) *One Cohesive Responsibility per Class*: Assign a unique, cohesive responsibility, within context, to each class. This responsibility must match the class' defined role. Avoiding overlapping/redundant responsibilities will prevent complex class interactions when applying them in design scenarios.
(5) *Self-Descriptive Services*: Provide a descriptive, straightforward definition of the Services per class. These services will sustain a strong correlation with the class' responsibility. Otherwise, it would be difficult to know which services to invoke to fulfill an artifact's specific job.
(6) *Explicit Notion of the External Collaborators*: Identify the class's collaborators. A class needs to know which artifacts/classes are its collaborators for the achievement of responsibilities.

Based on the Effective CRC Card format, described in [3], we provide a table listing a summary of CRC cards relevant characteristics.

CHARACTERISTIC	SOLUTION
Portable	No computers are required, they can be used anywhere, from the tranquility of your home to a very important meeting.
Reviewable	You can go back and review these index cards anytime after a long period of time without being concern of information deterioration.
Simple	It possesses a simple structure, easy to read, learn and understand by any person without a previous experience on CRC Cards.
Multi-Purpose	Due to its simplicity and its portability, CRC Cards may be utilized in different application domains, e.g. Education, Software Analysis & Design, As a Teaching Technique, etc.
Accessable	The set of CRC index Cards are highly available during the sharing decision process done by analysts, designers, and developers in the early stages of the Software development phase.
Implementable	From the CRC Card blueprint to its Implementation there is a short path to take. Due to a well-definition of the classes within the self-described structure of CRC Cards, developers are able to implement these cards (classes) with ease.
Traceable	These cards can be traced throughout the entire specification of animated scenarios by the explicit exhibition of the classes and their different roles and behavior to which it represents.
Mapping Ability	These index cards represent an exact match and definition of a class, and its elements, in a particular design scenario.
Reusable	After a project is completed, this does not mean that we cannot use our already defined index cards in another project. The classes were defined with a stable and reusable core in mind; therefore, they may be utilized within several applications that share the same domain knowledge. Think about this as a piece of knowledge that may be shared in other contexts of applicability that possesses similar rationale and capabilities, and hence, the effort in coming up with a stable system design would be reduced, and quality enhanced.

Table 1: Explicit Characteristics of CRC Cards and Solutions

3. CRC CARD KNOWLEDGE CLASSIFICATION:

Our primary focus is on the higher level patterns that conforms the underlying goals found in CRC Card utilization: Patterns that help us to develop more understanding of CRC Cards and their effective utilization.

It is worthwhile to mention that product of the association of higher level patterns (Goals) and semi-tangible patterns (Capabilities) we will be able to foresee a generic skeleton for myriad of development scenarios of CRC Cards. Figure 2 illustrates a general view of the CRC Card knowledge stratification in relation to its goals, and capabilities. However, the domain knowledge from which CRC Cards are made up may be enormous, therefore, for simplicity purpose we include few of them.

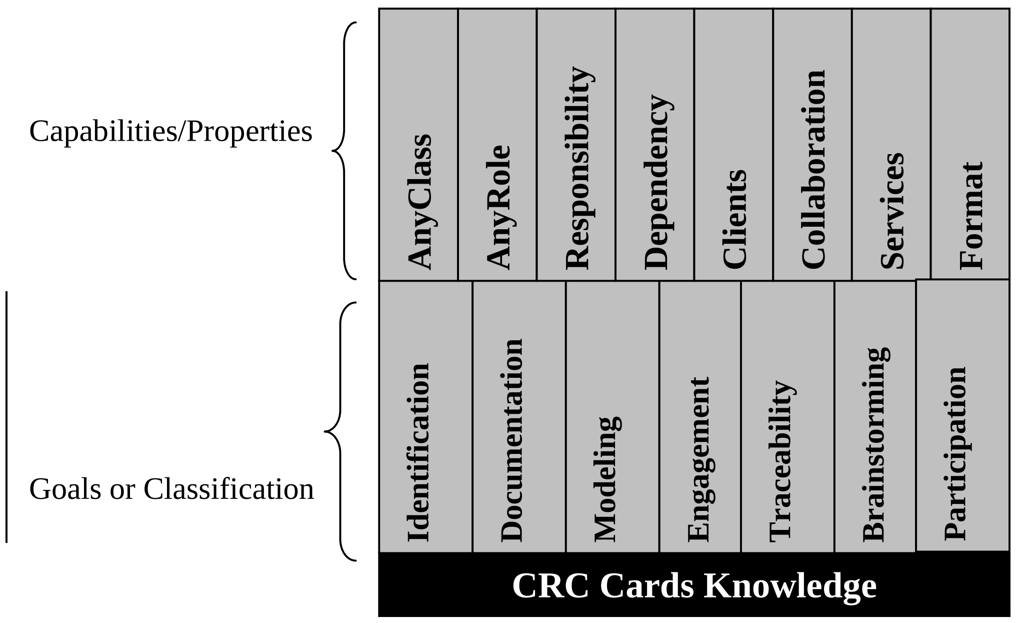

Figure 2: CRC Card Knowledge Classification

In the following section we provide a more specific view of the distinct elements involved in the CRC Card domain knowledge by the means of using and describing a Map representation. This Map representation focuses on the realization of the dissimilar artifacts, quality factors, and how they are associated with each other within the CRC Cards domain boundary. It is worthwhile to mention that each element or artifact represented in this map would come to represent a stable pattern.

4. TOWARDS A PATTERN LANGUAGE FOR CRC CARDS

CRC cards goals and capabilities embody a set of related stable patterns that build CRC cards rationale and usage over a myriad of application contexts. When related stable patterns are interlaced together they form a family of patterns that will cover multiple domains; this family of patterns is called Pattern Languages.

The objective of the overall pattern language is to cover the essential aspects related to the process of conceiving, understanding, writing, and utilizing CRC Cards. Concretely speaking, this pattern language will come to surface the behind the scenes endured knowledge of CRC cards. The process of defining CRC Cards endured knowledge involves four main steps, which can be perceived as the basis for our pattern categorization: 1. *Goals or Classification, 2. Capabilities/Properties of CRC Cards, 3. Development or Scenario Development of CRC Cards, and finally 4. Deployment of CRC Cards* across multiple domains. The outcome of these main steps will be a set of interrelated patterns that interact together to serve a particular purpose, within CRC cards usage and rationale. As a whole, this Pattern Language will embody the core insights as a set of stable patterns and their interactions among them.

If we delve into the Pattern Language definition presented in [17], our Pattern Language for CRC cards is far from being just a decision tree of patterns. On the contrary, it is a network of patterns (not hierarchy of patterns), where each generated interaction among its set of patterns come to represent a distinctive route or path serving a particular purpose or goal. Therefore, the number of distinctive routes that can be orderly navigated to satisfy distinctive purposes can be a very large number.

Before getting started with the family of patterns, let's go through the number of patterns that would be presented in this paper, along with the ones that will be included in future versions. Also, let's mention the undertaken methodology and how patterns will be allocated accordingly to their target purpose via a knowledge map. Currently, our Pattern Language for CRC Cards proposes twenty two patterns covering in certain degree CRC Cards' domain knowledge, where only 4 patterns are documented in this paper. More patterns will be added or documented in future version of this paper. Additionally, as stated above, these patterns would be discovered and organized by means of applying the aforementioned 4 main steps; each one of them addressing a particular objective within the definition of our CRC cards language.

- Goals or Classification:

  This step is concerned with surfacing the implicit goals hidden within the CRC Cards core knowledge. This process requires the capture and fully understanding of the context in where our solution would be laid down. That includes, describing the goals not from its tangible side, but focusing more on its conceptual side. In [18] they are named Enduring Business Themes (EBTs). Examples of the resulted patterns represented within this main step are Documentation goal, Identification goal, and Brainstorming goal, etc.

- Capabilities or Properties

  The second step concentrates in the discovery on those recipes required to fulfill the stated goals and purposes of the CRC Cards. Without those concepts or stable patterns there will be a vague understanding (almost none) on how these goals will be achieved. These stable patterns are known in [18] as Business Objects (BOs). For instance, within our language for CRC Cards, we have: AnyClass, AnyRole, Responsibility, Services, etc.

- Development Scenarios:

  The third step is concerned with: 1- the myriad of development scenarios in where the CRC Cards can be involved. These development scenarios are realized through the distinct routes or paths taken due to the interactions among the involved patterns (EBTs associated with BOs). The product of this association is known in [18] as Architectural Patterns. Each one the complete routes taken will represent a distinct application domain or context in where the CRC Cards would be used. For instance, Teaching Scenario, Groupware, etc. And 2 – how the CRC Cards language would be implemented across dissimilar domains based upon the utilization of tangible artifacts that would conform the domain specific patterns. These patterns are known in [18] as Industrial Objects (IOs). An example of these patterns would be: Book-keeping, etc.

- Deployment:

  The last step, as its name stated, deals not only with how the CRC Cards knowledge would be deployed across different application domains, but also with the representation of the artifacts or patterns that will aid the deployment process. For instance, we have the Blue Print pattern, etc.

The rationale of discovering and stratifying our Pattern Language by means of using a systematic approach, which involves four main steps, and ends up with four categories is to facilitate the findings, execution order, and description of the stable patterns embodying the concepts or building blocks of the CRC Cards domain.

Figure 3 depicts the overall pattern language structure. In the given figure, the main four steps are presented in light gray boxes with circle inside. The light gray boxes represent the generic aspects/recipes or stable patterns that are related to those steps. For instance, the first step is concerned with the analysis of the domain looking for hidden goals waving the domain under discussion. Each aspect is then interconnected with other set of patterns through the CRC Card Pattern Language.

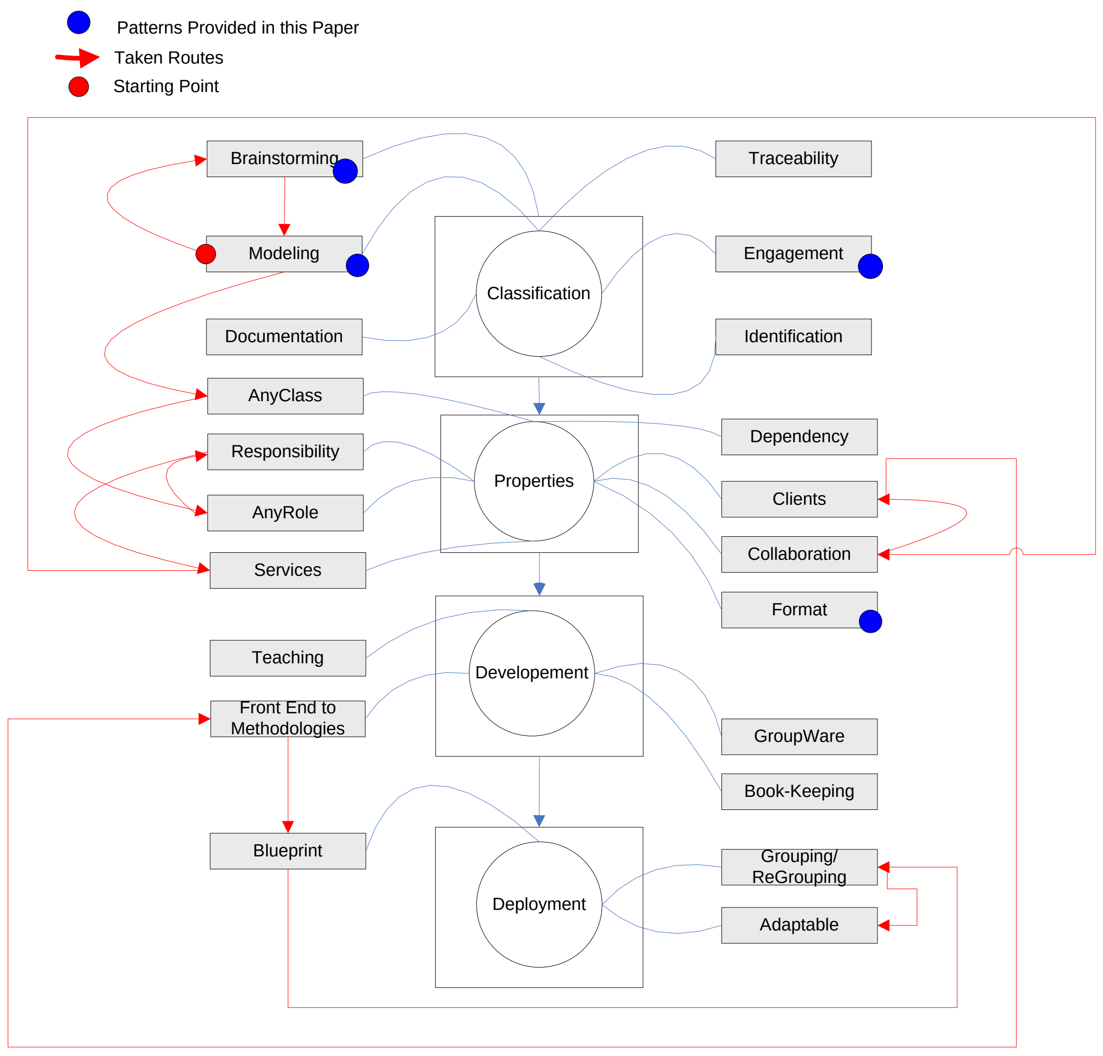

Figure 3: A Pattern language for CRC Cards – Achieving the Modeling Goal Scenario.

In summary, the routes taken when defining a new architecture would provide us the established roadmap to fulfill particular goals expected from the CRC Cards. The outcome of these interconnections represents self-supported aspects of the generated framework that will define the order of employing CRC Cards in dissimilar domains.

4.1 FAMILY OF PATTERNS – BIRD'S EYE VIEW:

The proposed Family of patterns, as stated above, contains twenty two patterns. These patterns as a whole describe the basis for defining, understanding, deploying and embodying a stable knowledge of the Class-Responsibility and Collaborator (CRC) Card across domains. Table 2 references in a concise way the artifacts included in this patterns language. For simplicity purpose only the patterns provided in this paper would be described. Future versions of this paper will include more related patterns to the CRC Cards domain knowledge and these patterns' description.

Category	Pattern	Problem	Solution
Goals or Classification	Documentation		
	Brainstorming	Current implementations of the Brainstorming process are bound to a specific problem domain.	Brainstorming Stable Analysis Pattern
	Engagement	We are always concerned about the quality of the involvement between participants involved in particular activity when interacting with each other.	Engagement Stable Analysis Pattern
	Traceability		
	Identification		
	Modeling	The actual problems range from the overloaded generation of too many responsibilities per class to the lack of specific class roles, which defined the position of a class in a pre-animated scenario in accordance to certain responsibility	Modeling Stable Analysis Pattern
	Participation		
Capabilities Properties	Any Class		
	Any Role		
	Responsibility		
	Dependency		
	Clients		
	Collaboration		
	Service		
	Format	Current CRC-Cards lack some essential qualities that might affect the effectiveness of the developed system that uses them.	Effective Format Pattern
Development	Teaching		
	Front End to Methodologies		
	GroupWare		
	Bookkeeping		
Deployment	Blueprint		
	Adaptable		
	Grouping/Regrouping		

Table 2: Summary of the Pattern Language of the CRC Cards

5. Pattern Language for CRC Cards

A Pattern Language is not a formal language, rather a family of interrelated patterns organized in such a way that facilitates a vocabulary that guides their application when solving standard problems [20]. In our case would be a pattern language for communicating the underlying knowledge of CRC cards' conception, understanding, and application.

In this section, we will describe only the patterns marked with a blue circle in the knowledge map shown in section 4. The rest of the patterns would be described in future versions of this paper.

5.1 CLASSIFICATION Main Step:

This step is concerned with surfacing the implicit goals hidden within the CRC Cards core knowledge. These goals form the basis for stable analysis patterns representation. In this paper we are presenting three stable analysis patterns: Brainstorming, Modeling, and Engagement stable analysis patterns.

PATTERN 1- BRAINSTORMING STABLE ANALYSIS PATTERN

Brainstorming is an informal way of generating ideas or solutions to write about, or points to generate a particular solution based on engaging activities.

Context:
Brainstorming is an informal way of generating ideas or solutions to write about, or points to generate a particular solution. It can be done at any time during the writing process or during certain interactions in particular meetings. In the case of CRC Cards, brainstorming process is intended to generate ideas with respect to the simulation of system behavior. Such process encloses several benefits, such as the treatment of several argumental issues, the achievement of agreement, engaging a particular group of people in the process of identifying candidate classes, their responsibility and collaborations according a distinctive context or predefined scenarios through the utilization of heterogeneous media. This technique is open to a “freestyle” generation of ideas and is not a technique for idea-evaluation. For instance, in a CRC Card session, the modeling team, aided by the use of index cards, will use to identify and understand the requirements for the application they are building. However, since there is not idea-evaluation, the application of this technique brings to surface some tradeoffs, such as a high uncertainty whether the generated idea is correct or not. Nevertheless, it sure helps practitioners to explore a vast set of possible solutions that may reflect the correct system behavior.
Problem:

Brainstorming process can be done at any time, individually or collectively; it can target one particular subject or multiple ones; it can be limited to one particular context or contexts; it can be done synchronously and asynchronously. Currently brainstorming solutions are limited to cope with one context at a time, not allowing certain grade of flexibility to allow practitioners expand their target context scope (s) of determined problem of discourse. Additionally, current brainstorming solutions strive when practitioners deal with several problem domains at the same time, especially when switching from different contexts back and forth, in a randomly order, and trying to keep certain topic’s discussion state where it was previously left by practitioners. Therefore, current brainstorming solutions are unsuitable for an across problem domain application.

In the case of CRC cards use and understanding, brainstorming process seems to be bound to the ability of a moderator to engage practitioners in collaborative environment and the willingness of practitioners to truly be part of the session. For such case, the process of engaging practitioners is detached from the brainstorming process itself. Bringing as a consequence, a poor running session, where the session generated results would have higher degree of failure due to the fragile link of attention experienced between the practitioners running the brainstorming process with respect a topic of discourse.

Solution:

The following model will represent the proposed solution of the Brainstorming Analysis pattern, using the Software Stability Concepts approach. See Figure 3:

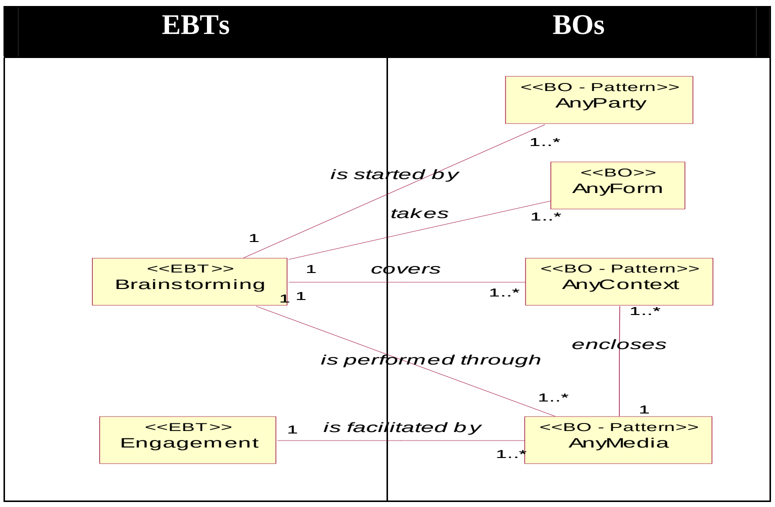

Figure 3: Brainstorming Analysis Pattern stable object model.

PARTICIPANTS:
The participants of the Brainstorming Analysis Pattern are:

CLASSES:
Brainstorming: Represent the Brainstorming process itself. This class contains the characteristics and behavior that initialize the brainstorming process.
Engagement: see Engagement Analysis Pattern.

PATTERNS:
AnyMedia: Represents the media through which the brainstorming process will take place. For instance, one can brainstorm the candidate classes of the system being developed through the utilization of CRC Cards and Use Case Scenarios, utilizing the Role play brainstorming process. Others might use simply the CRC Cards, in a model, to represent all the candidate classes and then brainstorm how these classes interact with each other in a high level representation of the system.

AnyParty: Represents the brainstorming inducers or practitioners. It models all the parties that are involved in the brainstorming process, including the facilitator, who is the person that rules the entire brainstorming process. Party can be a person, organization, or a group with specific orientation and organization.

AnyForm: Represents the forms, and how the brainstorming process can be performed. It models all the forms that may be employed when carrying out a brainstorming process.

Form can be a CRC Card modeling, a Role-playing and Use Case form, or a writing Brainstorming form [12].

AnyContext: Represents the brainstorming topics, problem or subjects to be brainstormed with. It models all the contexts that may be covered to generate possible solutions using a brainstorming process. It includes the scope or boundaries of the context, and the specific points to be discussed. AnyContext can be Class Identification step, Responsibility identification, or just writing an action novel for a local magazine.

Example:

In order to illustrate the use of the Brainstorming pattern in different application areas, one example is presented: CRC cards are one method where Brainstorming is present, by interacting in a Role-playing and Use Case Scenarios to discover, with users, the real-world objects which make up a system. They are meant to assist in mapping the collaborations among classes. Since the purpose of this example is to demonstrate the usage of the proposed pattern, and for simplicity, this example does not present the complete model for the problem. Instead, they focus on the part that involves the brainstorming process.

Example 1: Role-Playing and Use Case Scenario Brainstorming:
Role Playing and Use Case scenario brainstorming process required a list of predifined scenarios, a group of CRC Cards, and the practicioners of the brainstorming session (Analyst, Designers, Developers), including the leader or facilitaror of the session. However, it is almost impossible to make all the practicioners participate openly in those type of sessions. Usually is because they are afraid to give a bad idea or opinion. Here is where the facilitator has to come to shine, he needs to bring all the practioners into a complete engagement by distributing the CRC Cards to the team members, so that team members ‘play’ one of the classes (preferably not so that possible collaborating classes are assigned to the same individual) . This example models a simple solution to interact in an interesting brainstorming process called Role-Playing and Use Case Scenario and generates certain points or ideas that refer to the possible candidate classes, their responsibility and collaborations at a specific scenario. Figure 4 shows the stability model of the brainstorming used in *Role-Playing and Use Case Scenario*. Classes that are not in the original *Brainstorming* pattern are colored in gray.

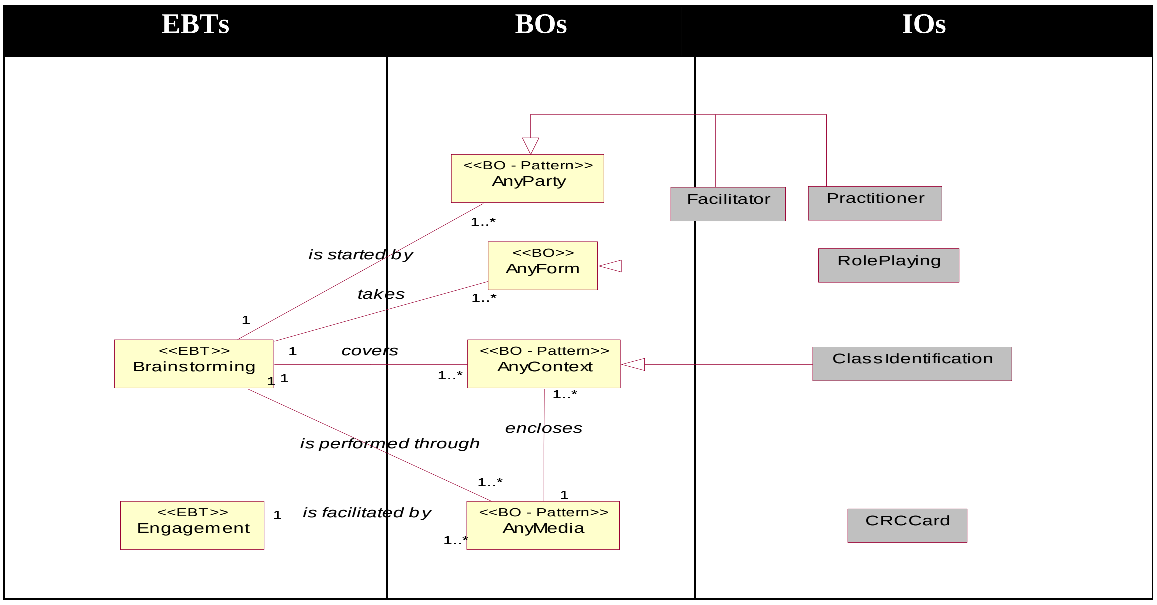

Figure 4: Brainstorming Analysis Pattern, stable object model Applicability

PATTERN 2- ENGAGEMENT STABLE ANALYSIS PATTERN

Engagement represents the process for candidates (participant) to meaningfully involve in a particular activity (s), through interaction with others and worthwhile activities. Such engagement could be accomplished without the use of technology; however, technology may facilitate engagement in ways which are difficult to achieve otherwise [15].

Context:

By engagement process, we meant the sense of concern with and curiosity about a particular activity, in which all participants are immersed in an iterative environment, based on their level of commitment, and disposition regarding a particular set of tasks.

Engagement concern is based upon the idea of creating a collaborative environment for its participants where the act of sharing activities and knowledge is increased. The context may be summarized based on the following elements:

1. Collaborative Teams or Participants
2. Participants' proficiency in certain skills
3. Strong Commitment towards the activity to be performed
4. Be truly involved in particular activity
5. Activities within scope.

Problem:

We are always concerned about the quality of the involvement between participants involved in particular activity when interacting with each other. Those concerns become more noticeable from time to time. This is especially so when we trying to engage participants with different level proficiency toward a particular task. A number of barriers interfere with such 'Effort'. Many involve lack of commitment or interest from participants. These, in turn, may delay or halt the completion of the assigned set of tasks. Other obstacles relate to process of defining the context boundaries of these activities. Therefore, accomplishing engagement may be hinder when proficiency on particular skills and activity's context are not in consensus (not related).

Solution:

The following model represent an abstract representation of the solution that deal with the Engagement Concept:

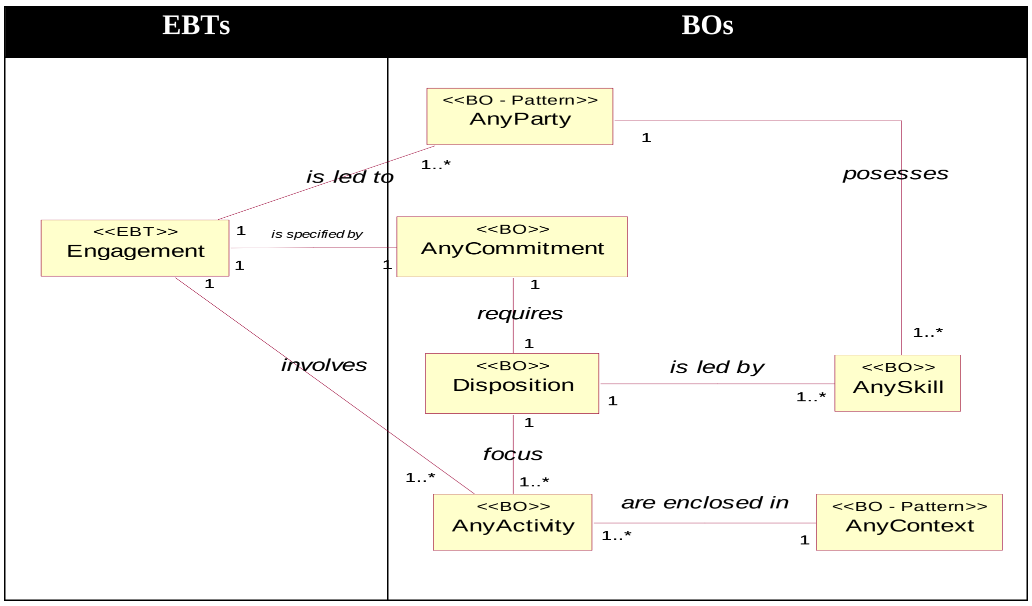

Figure 5: Engagement Analysis Pattern stable object model

Example: Conceptual Map Creation
In order to illustrate the use of the Engagement pattern in different application areas, one example is presented: The creation of a conceptual map [16]. Since the purpose of this example is to demonstrate the usage of the proposed pattern, and for simplicity, this example does not present the complete model for the problem. Instead, they focus on the part that involves the engagement process

Example 1: Creating a Conceptual Map to predict a constructive engagement.

The Creation of a conceptual map [16] should involve reshaping, adding personal links, keywords and conceptual areas. Note that these activities are considered when undertaken by a single user. Therefore user must be familiar with the semantics of creating conceptual maps. Only activities associated with the use of the navigation aid are considered, rather than activities associated with other aspects of the hypermedia content.

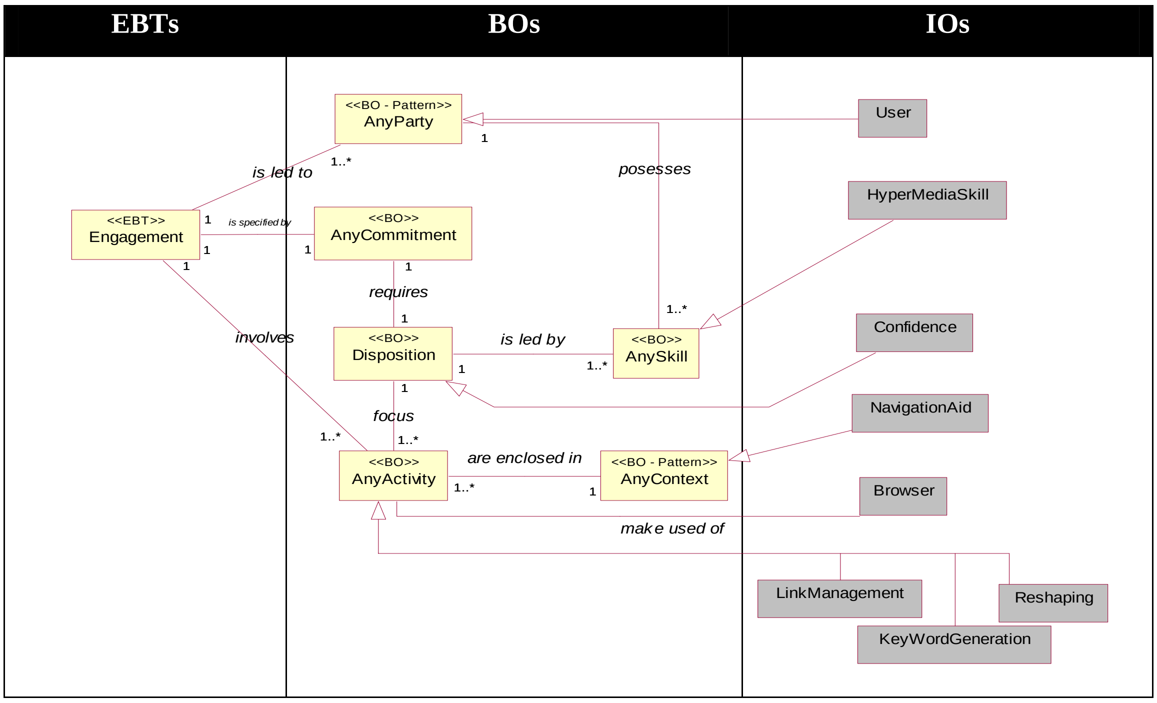

Figure 6: Stable Conceptual Map Creation

PATTERN 3- CRC CARD MODELING STABLE ANALYSIS PATTERN

The intent is to represent a collection of index cards and their interconnection with each other to provide a big picture of the system functionality.

Context:

A CRC card model is low-tech method that implies a collection of index cards arranged in client/server order to picture how the system, being developed, will function. We mean by "client/server order" as bringing interlocked classes or classes collaborating with each other closer in the model (having one or more server classes with their clients). This session is an interactive process, which brings together analysts, designers, developers, and/or any other professional that accedes to participate in this pre-animated exploration of the understanding and identification of the business requirements. As a low-tech, simple yet powerful method for Object-Oriented analysis, this technique provides great opportunities for a better understanding of the system throughout its entire development cycle through a Responsibility-driven analysis.

Usually this CRC Card model is created by a group of domain experts, who are led by the CRC card session moderator or also known as "facilitator" [13]. This moderator is in charge of organizing and performing the CRC card session, providing clear information and background to the participants about the CRC Card techniques and how it should be performed. This moderator is usually assisted by one or two scribes who are in charge or recording the logical perspective on how to fulfill the business requirements product of a constant analysis of the pre-animated scenarios executed in the CRC Card session.

Problem:

During the execution of the CRC Modeling session, domain experts initiate a looping process in which, based on brainstorming and several other methodical processes [3], all the candidate classes, its responsibilities, and collaborations are identified, and then, start filling their respective index cards, creating use case scenarios, and arranging these index card on certain table. However, current CRC Card formats are bound to enclose limited information about the classes being defined. These actual problems range from the overloaded generation of too many responsibilities per class to the lack of specific class roles, which defined the position of a class in a pre-animated scenario in accordance to certain responsibility [3]. These inconsistencies found on current CRC Cards formats will restrain the fluency in which the execution of the CRC modeling session will be held, due to the increment of the times a process will carry out to assure an accurate definition of a classes or an accurate match to the business requirements. Our problem consist of the major issues; the first one is how to express the process of running the CRC Modeling technique using the quality factors [Section 2] already built-in the proposed CRC Card format to better understand the business requirements, and the second one is, how to accurately identify the artifacts that would be modeled; that includes accurately defining the artifacts' enclosed elements. So at the end, practitioners would feel confident of the gained understanding of the system behavior being simulated during a pre-animated session.

Forces:

- Before starting a CRC Card session, the practitioners must have a clear understanding on how to perform this session. Usually, this is done by the Facilitator prior the beginning of the Session. Therefore, some sort of workflow may be useful for practitioners to enhance their understanding of the process.
- How to verify the identified artifacts or candidate classes, along with their enclosed elements, are accurate and widely express the expected behavior from the system being simulated when being modeled as whole.
- How to include the CRC Card quality factors into this workflow? Common workflow for CRC Card session lack of this element. Therefore, based on the proposed CRC Card format [3], those quality factors must be added to the defined process when running a CRC card session.

Solution:

For CRC card modeling there are six steps that need to be followed [13]:

1. Put together the CRC Modeling team.
2. Organize the Modeling room.
3. Do some Brainstorming.
4. Explain the CRC Modeling technique.
5. Iteratively perform the steps of CRC Card Modeling.
6. Perform use case scenario testing.

These steps, throughout the entire process, need to be constantly supervised by the CRC card moderator to make sure that the session is efficiently performed. In some cases, this moderator would allow the entrance of certain observers to the session. Typically, this clearance is for training purposes. These observers, however, cannot actively participate in the CRC Card session.

At the time of identifying candidate classes, their responsibilities, and their collaborators, and then turn them into a CRC Card, we make use of a new look at the CRC Card format described in [3] and the methodical process to fill it. This new format has a lot more to offer than currently used formats [10, 11, and 14], due to the avoidance of all the problems described in [Section 5.2] and provides a clear blueprint of their respective candidate classes.

The following process flow aims to represent the appropriate steps to model candidate classes, previously identified using some brainstorming and methodical process [3], embodied as CRC cards, to see if they match the business requirements of the entire System. This new process flow is an adaptation of the process flow described in [14], but solely focusing on the new CRC card format proposed in [3], avoiding for example the inclusion of many responsibilities per class, and the appearance of distinctive role per class. More information about this new CRC card format, please refer to [3].

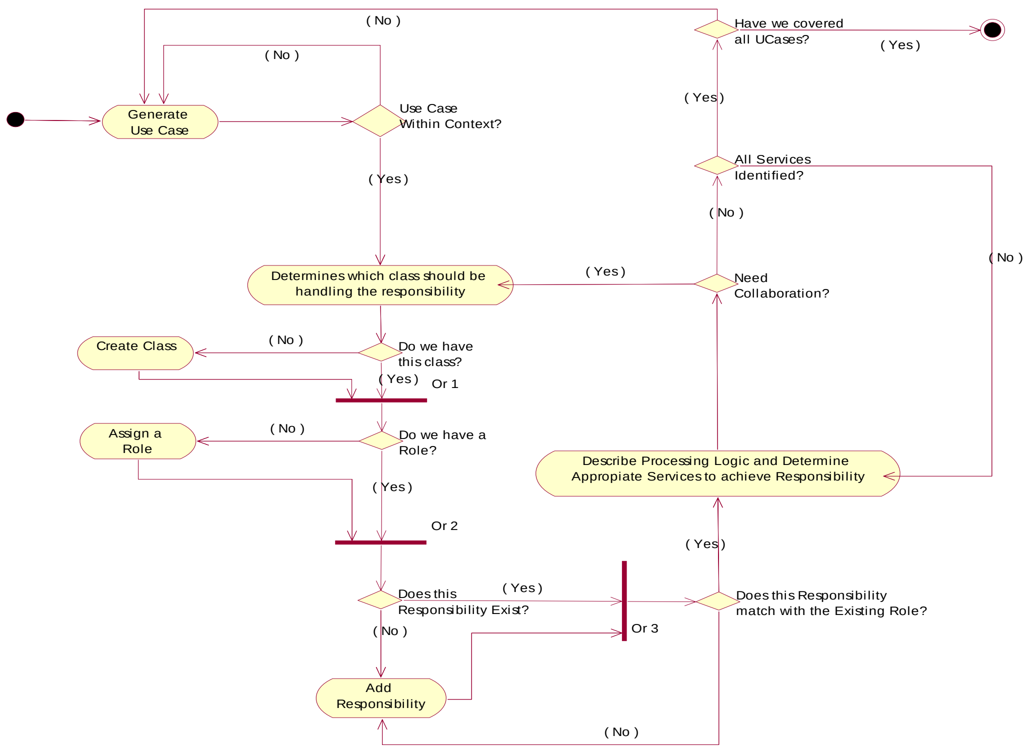

Figure 7: CRC Modeling Process

It is important to note that the success of this modeling process relies on how good and accurate your model mirrors the problem domain from where your application is being developed.

5.2 CAPABILITIES MAIN STEP:

This step emphasizes the discovery on those recipes required to fulfill the stated goals and purposes of the CRC Cards. These recipes are embodied as stable design patterns. In this paper we present only one stable design pattern: Effective CRC Card Format pattern

PATTERN 4 - EFFECTIVE CRC CARD FORMAT PATTERN

Context:
The CRC-Cards technique can be used in either practical software development or in Object Oriented (OO) education. In short, it embodies a higher grade of reusability, proportional to the number of problems it modes. For instance, in development, CRC-Cards can offer valuable insights during the early stages of development. Another benefit of CRC Cards is in teaching Object Concepts in programming language courses. Several reported case studies have demonstrated the effectiveness of CRC Card as a tool for introducing OO programming concepts and for improving the understanding of objects/classes [8].

Problem:
Current CRC-Cards lack some essential qualities that might affect the effectiveness of the developed system that uses them. The main problems in current CRC-Cards can be summarized in the following points:

1. *Possibility of Low Cohesion and High Coupling:* Because there are no limitations on the number of responsibilities allowed for a given class, it is possible to overload a class with too many responsibilities resulting in low cohesion within the model. A cohesive class should ideally have just one responsibility [3, 9]. Excessive responsibilities could also lead to a large number of collaborators required to support them. Too many collaborations between classes will produce high coupling and needlessly increase the complexity of the system. In software design, we strive for just the opposite – high cohesion and the lowest possible coupling.

2. *Macho Classes:* Multiple responsibilities can also result in the creation of macho, classes. A macho class instantiates an object that performs most of the work, leaving all of the minor operations to a set of essentially useless classes [9]. Ideally, the system intelligence should be distributed as evenly as possible across the application and the work shared uniformly. When all of the intelligence is concentrated in one or two classes, it also increases the difficulty of making changes.

3. *Exclusion of Services:* Another problem with these CRC cards is the exclusion of the services provided by the class [3, 4]. By including the services on the CRC card, the classes can be checked for duplicate functionality [9]. By identifying duplicate

functionality, it may be possible to combine or consolidate classes that perform similar functions. In addition, because the responsibility of a class is merely a summary of its operations, explicitly providing the services performed by a class may help verify the responsibility is properly defined.

4. No Clear Role is Defined: The absence of a class role may lead to assigning wrong, useless, or even missing responsibilities. Although the role seems fairly insignificant, it serves a very important purpose. If a class performs more than one role, it is possible that a generalization exists where each role is actually a subclass of some superclass [9]. Humans provide a good example of performing multiple roles; a woman could be a mother, a wife, a daughter, etc. By defining the role, generalizations and specializations can be explored early in the process.

5. Difficulty in Defining Responsibilities: Coming up with class responsibilities can be a difficult task, especially with the absence of a clearly defined role [3]. It is easy to get off track and assign responsibilities that are either ambiguous or irrelevant. It isn't until the developer begins to actually map the CRC cards to various use case scenarios or the class diagram that these extraneous responsibilities are realized.

6. Difficulty in Mapping: Multiple responsibilities can make it difficult to map the classes identified by the CRC Cards to the actual class diagram and use case scenarios. When numerous responsibilities are assigned to one class, the interactions can become complex [3, 9]. This complexity carries over when the CRC cards are used to map use cases. The use cases are provided to determine if the class model provides the necessary functionally to support all possible scenarios. Because the responsibilities of a class are actually a summary of its functionality, mapping can be greatly complicated when the class's operations are tied to multiple responsibilities.

Forces:
For CRC-Cards to enhance the development of systems, main essential quality factors should be satisfied [3]. Yet, satisfying these qualities is not easy. The following summarizes the main points that shows writing an effective CRC-Card are not straightforward:

- A major advantage of the CRC-Cards tool resides in its simplicity to understand. Current CRC-Cards consists of three elements: name of the class, its responsibilities, and its collaborations. However, such simplicity may not convey all the required information needed in the following steps in the development. For example, collaboration section in current CRC-Cards does not provide any information about what kind of collaboration there is. In other words, the card does not specify the services that its class offers to the other classes that collaborate with it. Such information is important in developing the class diagram and in verifying its accuracy. On the other hand, having too much information in a CRC-Card might preclude them from being widely applied. They might become difficult to understand, or to use. This may scarify the simplicity of the technique. Therefore, compromising between completeness and simplicity should be considered when writing CRC-Cards.

- It is important to understand the role of each class in the system in order to identify its responsibilities [9]. A class within the system might have several roles; however, current CRC-Cards do not provide a way to differentiate between the different roles of the same class. How can we handle multiple roles for the same class in a simple way?
- Defining the class responsibility is crucial for developing an accurate class diagram and latter an effective system. A class might have several responsibilities within the system; however, identifying all these responsibilities in one CRC-Card might create great confusion for the developers [3]. This is because different responsibilities for a class can result in different collaborations between this class and other classes in the system. By listing all the responsibilities and collaborations of a class in one CRC-Card, it becomes confusing to match a responsibility of a class to another collaborating class in the system. This complicates the deriving of the system class diagram from the written CRC-Cards.

Solution:
In this section we present an enhanced representation for CRC cards as a solution to some of the problems in current CRC-Cards. The new version will include a clear role for each class which will aid in the discovery of superclasses and their respective subclasses. This class role will also be useful when defining the class responsibility. Each class will be allowed to have only one unique responsibility. If more than one responsibility is identified, additional classes should be formed. Limiting responsibilities will help prevent low cohesion and high coupling as well as reduce the possibility of macho classes. Finally, the revised CRC card will include the services offered by each class. This will help verify the validity of the class responsibility as well as ensure that overlapping functionality is avoided. The proposed CRC-Card format is shown in Figure 8 below.

Class (Role)		
Responsibility	**Collaboration**	
	Client	**Server**

Figure 8: Proposed CRC-Card Format

Creating CRC-cards with the proposed format requires three main steps (See Figure 9):

(1) Class Identification.
(2) Assigning Roles and Responsibilities.
(3) Discovering Collaborators.

Each step is concern with filling one element of the proposed CRC Card at a time. In summary, relevant classes of the system will be identified. Then, proper roles would be assigned to them based on a number of responsibilities found in particular design scenarios. Collaborators or Clients of each class would be discovered to account for the certain inabilities of some classes to fulfill its assigned responsibility [10]. This process will be aided by the explicit declaration of the external services offered by each relevant class.

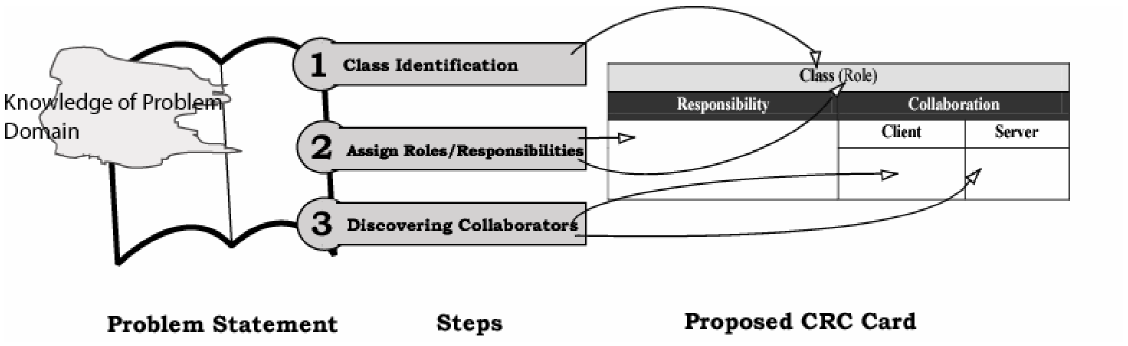

Figure 9: Applying the Proposed CRC-Card

Example:

In this section we show an example of applying the effective CRC-Card on a simple problem.

Problem statement

The services that are linked with the word *Genealogy* have been growing tremendously across the Internet landscape. This idea has been touched by several online businesses, such as Ancestry.com, Msn, etc., but it is still in its infancy. We propose a simple Family Tree design. This system will offer a central storage device, where all the information of current members will be stored. Each member will have full control of his/her information portrayed in the system.

The system will provide consistent updating, searching and tracking mechanisms to facilitate an easy interaction between the system and the members throughout the entire Family Tree. An efficient user-friendly navigation mechanism will be presented as well. This will guarantee full access to all the features of the system. Members of the system will be able to share this experience by inviting new users to either start up their own family tree, and/or enroll in a member's family tree. The latter would happen in the case that these potential members are relatives of an already enrolled member. Regarding guests of the system, there will be a section exclusively for them, allowing them to visit current Family trees with certain limitations. They can also create their own family tree if they desire.

Step 1: Identifying Classes

The Class identification process does not vary in both approaches. Both approaches use similar techniques [10, 11] along with the overall knowledge on the subject from analysts, and designers. Similar naming conventions are applied, except they vary when dealing with compound nouns. Compound nouns are treated as one word, and no spaces are allowed between these two words. This is sometimes called "camel-casing". For method declarations the accepted notation is just as it is for classes *except* that the first letter is in lowercase. The following is an example of a possible class in the presented problem:

FamilyTree (Role)		
Responsibility	**Collaboration**	
	Client	**Server**

Figure 10: A Class representation using CRC Card

Step 2: Assigning Roles and Responsibilities

The proposed CRC Card format offers much more to the process of discovering responsibilities than the current one. The presence of a well-defined role makes things easier for the analyst because each role is tightly bound to a unique responsibility [3, 9]. Therefore, the analyst can map the distinct responsibilities per class based on particular scenarios. Other techniques may be used to assist this process [10].

A class that contains multiple responsibilities will be partitioned into several classes [3]. No naming rules are required in this step. However, for understanding purpose, analysts need to define clear and cohesive responsibilities. The following is an example of assigning Responsibilities and Roles:

FamilyTree (FamilyTree)		
Responsibility	**Collaboration**	
Illustrate the bond of a group of people	**Client**	**Server**

Figure 11: Assigning a Role and unique Responsibility

Step 3: Discovering Collaborators.

The goal of this step is the definition of a set of methods that will help to achieve a responsibility. The most common techniques used for services identification is the application of a grammatical parse over the problem statement, looking for verbs and/or verbs phrases [9, 10]. These verbs or verb phrases usually corresponds to the methods used to fulfill certain functionality or unique responsibility of a particular class. They will be explicitly nested on the right compartment of the Collaboration section of the proposed CRC Card. This compartment is name *Server*. They obey certain naming rules to assure a proper definition. The Identifying Classes section refers to these naming rules.

The inclusion of these services in the proposed CRC Card will enhance the ability for responsibility identification, making it a very straightforward process. Along with this inclusion of services, several classes that communicate with one particular class will be placed on the left compartment of the Collaboration section. This section is called *Clients*. These classes are called clients because they collaborate with a particular class, by requesting services from it. These requests are performed based on a message-wise action. These classes (Clients) are usually identified at the moment of establishing an interaction between classes and its surroundings in a particular design scenario (e.g. Use Case Scenario) [9]. This step is not just a common step. Having exhibited two more sections of this CRC Card, we have granted the accomplishment of several quality factors at the same time (e.g. Self-descriptive Services, Explicit notion of Collaborators) [3]. This result will also increase the level of understandability for analysts and designers towards the proposed CRC Card structure.

Coming up with classes (Clients) is a repeatable process done by analysts/developers and it will be completed only when the analysts/designers feel that they have covered all the distinct design scenarios. The following is an example of filling out the Collaboration section of the proposed CRC Card.

FamilyTree (FamilyTree)		
Responsibility	**Collaboration**	
To show relationships between people	**Client**	**Server**

	Member
	Family
		initiateTree()
		connectFamily()
		joinToTree()
		search()

Figure 12: Collaborators and Services Identification

6. CONCLUSION:

Our objectives for defining this pattern language for CRC Cards were concentrated in the offering of a suitable language for writing, and applying CRC Cards across domains. This language definition was aided by the addressing of CRC Card’s domain knowledge, and their fundamental deployment processes from a conceptual and purpose-driven perspective. This conceptualization and understanding was laid out through “four main steps stratification" and a set of stable patterns. The rationale of this stratification was to facilitate the findings, execution order, and description of the stable patterns embodying the concepts or building blocks of the CRC Cards domain. The resulted work was possible due to the synergy of two methodologies: the Software Stability Concepts paradigm [5] and its sub-elements: Stable Analysis and Design Patterns [5], and Pattern Language Methodology [18]. Patterns left without description would be address in future versions of the paper.

REFERENCES:

[1] Harold Halbleib, "Software Design Using CRC Cards", Real Time Magazine 99-1, www.realtime-info.com, 1999.

[2] David M. Rubin, "Introduction to CRC Cards", Methodologies and Practices, Softstart Research, Inc.

[3] Mohamed Fayad, Valerie Stanton, Huascar Sanchez, and Haitham Hamza "A Closer Look at Class Responsibility Collaborator (CRC) Cards", www.activeframework.com.

[4] M.E. Fayad, H.S. Hamza, and H.A. Sánchez. A Pattern for an Effective Class Responsibility Collaborator (CRC) Cards, The 2003 IEEE International Conference on Information Reuse and Integration, Las Vegas, NV, October 2003.

[5] Mohamed Fayad, Haitham Hamza, "Software Stability Background", www.activeframework.com.

[6] Robert Biddle, James Noble, and Ewan Tempero, "Role-Play and Use Case Cards for Requirements Review", Proceedings of the 20th Australian Conference on Information System.

[7] Beck, K., and Cunningham "A Laboratory for Teaching Object-Oriented Thinking", OOPSLA' 89, Conference Proceedings, (1989).

[8] Jürgen Börstler, Thomas Johansson, and Marie Nordström , "Teaching OO Concepts—A Case Study Using CRC-CARDS and BLUEJ", 32nd ASEE/IEEE Frontiers in Education Conference, November 6 - 9, 2002, Boston, MA.

[9] M.E. Fayad. Object-Oriented Analysis and Design Lecture notes, Full 2000 to Spring 2003.

[10] Roger S. Pressman, "Software Engineering – A Practitioner's Approach," McGraw Hill Publishing Company, 2001.

[11] Leszek A. Maciaszek, "Requirements Analysis and System Design – Developing Information System with UML," Addison-Wesley Publishing Company, 2001.

[12] Will Durfee, "Brainstorming Basis," 1998.

[13] Scott W. Ambler, “CRC Modeling: Bridging the Communication Gap between Developers and Users,” An AmbySoft Inc. White Paper, 1998.

[14] Scott W. Ambler, “The Object Primer: The Application Developer’s Guide to Object Orientation,” An AmbySoft Inc.

[15] Greg Kearsley, Ben Shneiderman, “Engagement Theory: A framework for technology-based teaching and learning ,” http://home.sprynet.com/~gkearsley/engage.htm.

[16] Ursula Armitage, “Can Navigation Aids Support Constructive Engagement with Hypermedia?,” Centre for HCI Design, City University, Northampton Square, London EC1V OHB.

[17] James O. Coplien, “Software Patterns”, BellSouth Laboratories, The Hillside Group.

[18] M.E. Fayad, H.A. Sanchez, Ram Goverdhan. A Goal-Driven Software Development Life Cycle, www.activeframeworks.com , in progress.

[19] Harold Halbleib, “Software Design using CRC Cards”, Real-Time Magazine, 99-1, http://www.realtime-info.com.

[20] Douglas C. Schmidt, Mohamed E. Fayad, and Ralph E. Johnson, “Software Patterns”, Communications of ACM, October 1996/Vol. 39. No. 10.