FP-International Journal of Computer Science Research
Volume-1, Issue-1, Pages 1-12, 2014

CRYPTANALYTIC SYSTEMS USING RATIONAL UNIFIED MODELLING TECHNIQUE

Nithin.N
Assistant Professor
Department of Computer Science
Manipal Institute of technology
Manipal (Karnataka)
nithin.n@manipal.edu

Moulshree Kumar
BE Final Year
Department of Computer Science
Manipal Institute of technology
Manipal (Karnataka)
moulshreekumar91@gmail.com

Sanyogita Sao
BE Final Year
Department of Computer Science
Manipal Institute of technology
Manipal (Karnataka)
sanyo1810@gmail.com

***Abstract*— the complexity of cryptanalytic attack is increasing multi-fold and hence efficient systems need to be developed to secure the computer environment. Efficient cryptanalysis is of utmost importance to understand and analyse the working of a cryptographic system, which would further help in developing more secure computer systems. This paper presents how Object Oriented Analysis and Design can be used to create an Artificial Intelligence on Cryptanalysis. It includes three of the important topics in the current times- Object Oriented Analysis and Design, Computer Security, and Artificial Intelligence. It shows how a Rational Unified Process can be applied for creating an intelligent system, which carries out cryptanalysis for a simple single substitution cipher using the UML model. This can be considered as a base for creating more complex cryptanalytic attacks.**

Keywords- Cryptanalysis, OOA, Artificial Intelligence, API

1. INTRODUCTION

Cryptanalysis is the field of cryptography which deals with deciphering the message without having the actual key. Cryptanalysis is the method of "Breaking the Cipher" i.e. finding vulnerability in the cipher which can be exploited with a complexity less than brute force [3]. Cryptanalysis is highly essential for evaluating the level of security provided by the encryption technique. Since the cryptanalysis technique may change with each situation, it is highly essential to create an intelligent system to cover all aspects of cryptanalysis, in order to be able to rely on computer systems.

Software Development Life Cycle (SDLC) provides a foundation of activities or steps which are followed by software designers and developers for software development. SDLC helps to create a basic structure, organize, plan and regulate the process of developing a software system.

SDLC consists of six phases – Requirement gathering and analysis, Design, Implementation or coding, Testing, Deployment and Maintenance [4].

Requirement gathering and analysis phase is very crucial as it is the foundation of software design and development. Software engineers conduct several meetings with the stakeholders in order to understand their requirements.
Analysis on target clients, range of inputs to the system, desired outputs and cost of system are considered. After these considerations the actual feasibility of the plan is analysed. As an output of this phase a Requirement Specification document is created which is a guide for all future phases.

In the design phase the requirement specification document is studied upon to come up with a software design which meets all the software and hardware requirements. The software design specification is then forwarded to the next phase.

Software design specification is received by the team of implementation and coding phase. The actual implementation and coding based on the design is done in this phase. The various tasks are modularised in order to meet the target within the given time limit and to achieve higher efficiency.

This phase is followed by testing which is one of the most important steps. The code is tested against the requirements to ensure that all requirements have been achieved and the software is working as expected. Several tests such as unit testing, integration testing, system testing and acceptance testing are done.

In the deployment phase the product is sold or distributed to the customers. Installation of the product and providing customer support is a part of this phase. Continuous evaluation needs to be done during deployment. Maintenance phase includes rectifying the faults in the product and enhancing its features according to the customer feedbacks.

This paper will highlight the design phase of SDLC. Design phase is a process which consists of several steps, wherein depictions of the framework of data and program, properties of interface, and procedural intricacies are generated from information requirements collected in the earlier phase. Each
system contains of several components. The design phase represents the structure of data and program components that are required to develop a computer system. It considers the architectural style that the system will follow the characteristics and structures of the components that constitute the system, and the interactions and relationships between these architectural components of the system [14].

There are types of SDLC methods which have been developed and evolved through the years. Each of these methods has its merits and demerits, and is appropriate for a different type of project based on various factors like its technical aspects, organization and team style and dynamics, and other project considerations.

We will now discuss the various design models in detail [20**,** 16].

Waterfall model is considered to be a traditional method. In this model the project is divided into a sequence of phases. Planning, budgeting, dates and schedule, time management, implementing the system accordingly is stressed upon greatly at every step. A lot of documentation and formal reviews are constantly conducted both by the client and the management team. Approvals and reviews are generally done at the end of a phase. Waterfall model is apt for developing less complex applications where progress is measurable at every step and resources are not wasted. Measurability, debugging and reliability at every step lead to an efficient functional system. Its main disadvantage is that it is not iterative and it is not possible to carry out modification once the implementation is done. Testing is done at the final stage which makes making any sort of modifications cumbersome. Projects where requirements are clearly defined beforehand can be developed using this method, so that minimal changes are made at the time of testing. Real time and large projects should turn to other models for implementation.

Another model is Prototyping, which is implemented as a supplementing development strategy with other development models like spiral model and Rapid Application Development (RAD). Project development is iterative wherein at every step feedback is taken and changes are made accordingly. Most of the prototypes are designed for getting a basic idea of the project and its requirements. A prototype can also evolve into a working system. Periodic reviews from client and team improves communication and hence development is constructive. In contrast to waterfall model prototypes can be tested at each step with the reviews received from the clients. However, evolving prototype into an actual model can lead to missing out on major features of the working model. The client may confuse the prototype to be an actual model where in reality it may not be properly functional. Prototypes should be used in processes where the client is not clear with the requirements and can get a clear picture seeing the prototype.

In Incremental Model all the steps of a waterfall model are performed at every increment. Client feedback is considered in short spans of time which leads to positive communication between the software team and the client. This leads to effective changes in the system at every step. This design model leads to a more accurate final implementation due to the continuous review and testing of the system at each increment. It is difficult to maintain proper coordination between different modules of the project since some parts of the project are completed before the others. Technical requirements can also be neglected at some stages. This project is highly effective for large projects as it can be developed successfully through increments in spite of fluctuating requirements.

In a Spiral Model the same set of steps are repeated at each cycle for every component of the product. This is done at every level of elaboration, from an overall documentation of requirements to the actual implementation of every detail of the product **[2]**. The main aim of this model is to bring about changes in the product easily as per the changing requirements. The budget has to be flexible in order to implement this model as changes have to be accommodated at every round. Each trip around the spiral consists of four basic tasks. These include determining objectives, alternatives and constraints of the iteration, followed by evaluation of available alternatives giving special attention to the possibility of different kinds of risks. After this the output of the iteration is developed and verified. The last task involves planning of the next iteration. Every cycle is started with an identification of stakeholders and their requirements, and is ended with review and feedback. Spiral Model is considered to be

one of the best methods of development as takes into consideration all the requirements as well as risks involved at every step. Spiral model may take a longer time for completion as compared to other models as the spiral process is never ending and every activity is implemented at every cycle.

In Rapid Application Development (RAD) the main aim is to produce the results based upon the need of the application, while technological perfection is of lesser importance. The main emphasis is on the completion of the product within the given span of time .The project should be able to be modularized into sets of task so that different teams can be allotted to complete different modules within the given deadline. After the modules are finished they are integrated together which results in the final product. This makes the integration phase highly important as coordination between different modules can be a difficult task. Documentation for analysis is vital as projects are built in a short span of time and with minimal utilization of resources. Reusability is stressed upon during the implementation phase. This model may compromise on the system in order to meet project deadlines and the system may also lack measurability. A large number of dedicated engineers are required in order to successfully complete the product using this method.
Another important aspect of software development are the architectural styles. Architectural Styles can be defined as a detailed manner in which various components are integrated together to form a system. The following are the different types of architectural styles –

Call and return architecture consists of Main Program Architecture where the main program invokes various other components and Remote Procedure Call Architecture where the components of the main program are present amongst other computers on a network.
Another architectural style is Object Oriented Architecture which emphasizes on encapsulation of data and operation needed in components. Message passing is used to communicate between the different components. The key features of this architecture are data abstraction, inheritance and polymorphism.
In a Layered Architecture a number of layers are present, each with a specific predefined set of functions. The outer layer components perform user interface operations whereas the inner layers perform operating system interfacing. The intermediate layers are usually used to perform utility services.
A Service Oriented Architecture is an architectural method and style that provides services to communicate with users that need certain services across a network of remote and distant domains of services and technology

Domain Driven Design (DDD) is used for developing innovative domain models and domain systems using various business codes and methods. The domain specifications have to be followed to develop the domain and domain logic. Another category of architectural styles are 2-Tier and 3-Tier architectures. The 2-tier architectures consist of two parts, the client and the database. On the other hand 3-tier architectures consist of three layers which are client, software or protocol and database section. The client requests for services, whereas the protocol processes the services queried by the client. The database layer is the section from where the result of processing is sent back to the client. The Client and Server Architecture is one of the most common forms of architecture. The client sends a query or request to the server, which processes the request and sends back the result to the client. The client is then used as an interface to display the content sent by the server. This system is made efficient by proper communication between the clients and servers.

Object Oriented Approach (OOA) has remained to be a topic of discussion since the 1990's. Seeing the positive transformation it has brought in Software Development, it can easily be said that it is here to stay [2]. Object Oriented Approach has been incorporated in all fields and at all levels. It provides a high level of abstraction and focuses on the user's view and real world entities[18].
Hence, living in the era of OOA, its importance cannot be ignored. The power of OOA lies in combining the principles of encapsulation, abstraction, hierarchy, modularity, concurrency, typing, and persistence in a way which enhances the process, as well as the result of creating a software application. OOA concentrates on achieving the desired output rather than the procedure itself. OOA&D came into the limelight when the importance of faster and cheaper delivery of software increased. Reusability and extension are the key characteristics which help in achieving this aim.
Therefore, OOA&D helps in decomposition of complex problems into simpler independent modules. We have shown detailed steps of developing the artificial intelligence system for single substitution encryption in an object-oriented approach, using Unified Modelling Language (UML) for representations.
UML is considered to be one of the most powerful and important notational modelling languages [15], which is used to express the software development problems of interactive systems visually or diagrammatically. The UML framework helps representing the system in an object oriented manner. The system is usually divided into two views – Static View and Dynamic View. UML covers both these views with various types of diagrams.
Static View, which is the structural view of the system, is described using objects, attributes of the objects,

relationships between the objects and the operations done on the objects. These characteristics are included in the class diagrams and composite structure diagrams. On the other hand, Dynamic View and Functional View are considered to be the behavioural view of the system. These views show the collaborations among objects and the changes to the internal states of objects. This aspect is covered using sequence diagrams, activity diagrams and state machine diagrams [12]. UML leads to high reliability, robustness and improved manageability.

The paper starts with the background knowledge on cryptography, blackboard model, rational unified process and UML, which is required to understand the application being developed. This is followed by the actual development of the application using the SDLC phases mentioned in the start of this section. The analysis and design phases of SDLC are stressed upon to highlight the importance of UML in developing the application. The paper also explains how UML helps increase the reusability and extensibility of the application. The paper ends by describing the scope of using this approach to develop further applications in the field of cryptography or other areas.

2. BACKROUND AND RELATED WORK

A. *Substitution Algorithm in Cryptography*

Cryptography is the branch of science and modern security which implements various methods through which information between two communicating parties is kept confidential and safe. The messages shared are vulnerable to numerous attacks during transit; therefore cryptography is finding its use in every field nowadays where data and information is shared between any two parties.

The most basic and simplest method of cryptography is the mechanism of Substitution **[6]**. This was traditionally a very common method for coding and encoding of plaintext. It involves mapping of plaintext alphabets to corresponding cipher text alphabets. The plaintext alphabets are replaced by cipher text alphabets according to a decided rule or regular system. It is done in a way such that the third party who wants to attack or analyse the transmission of data is not able to decipher the original text. Examples of cryptographic algorithms using substitution algorithms are Caesar Cipher, Hill Cipher and Play fair Cipher.

B. *Blackboard Model in Artificial Intelligence*

Blackboard model is an architectural model used in numerous applications including those of artificial intelligence. It helps simplify complex problems by modularizing the problem and dividing it into steps controlled by predefined rules. The blackboard system is divided into primarily in three parts- Blackboard, Knowledge Sources and Controller.

The blackboard is a collection of data and assumptions. Every new assumption contributed by the knowledge sources are stored and maintained on the blackboard so that all knowledge sources can analyse the data clearly without assumptions and data contradicting each other.

Knowledge Sources are the information sources which help in creating a range of assumptions. These assumptions are then analysed to reach the final solution.

The controller provides the control mechanism between all the knowledge sources and the blackboard. It maintains hypothesizing of assumptions and tests the current states of these assumptions made by knowledge sources.

C. *Rational Unified Process (RUP)*

The Rational Unified Process (RUP)[9] is an iterative software development process framework. It systematically assigns tasks and responsibilities within a development organization. RUP was designed with the main goal of ensuring high-quality production of the software which focuses on the needs of its end-users, without compromising on the schedule and budget [16].

The RUP team is not restricted to the software analysts and developers, but is broadened to include the customers, partners, as well as the consultant organization. This helps in continuously evolving the product or process according to the changing situations, practices and requirements, thereby ensuring that the end product is the optimum one. To increase team productivity, RUP provides all the members of the team an access to a common database of guidelines, templates and tool mentors for the development activities. Therefore all the members working for all the phases of the software development have a common language, process, and target, leading to higher synergy of the team. RUP reduces the documentation as it focuses on development and maintenance of models. RUP and UML work simultaneously, as RUP assists in understanding how UML can be used effectively. RUP is a highly automated process which uses several tools to achieve this aim. The Rational Unified Process is supported by tools, which automate large parts of the process. They are used to create and maintain the various models of the software engineering process: visual modelling, programming, testing, etc. They are extremely important for maintaining the documentation

related with change and configuration management during each iteration. The Rational Unified Process can be built into small development teams as well as large development organizations due to its easy configurability. The Unified Process is based can act as a common fundamental unit for a large range of processes due to its elementary and clear process architecture. It can also be adjusted to different type of situations and processes due to its scope of detailing. To meet the needs of a particular organization a development kit is usually provided to configure the process accordingly. Therefore using the Unified approach gives several key advantages.

D. *Previous work on applications using UML*

The first application is the use of UML Use Cases in developing GUI Components. Use cases help in determining the initial requirements of the system and thus increase the ease of implementation.UML use case model can be used in designing GUI components through the use of use cases and activity diagrams describing the use cases. Furthermore code fragments can be generated based on these specifications which can be considered as GUI prototypes. The use case models help the designers to identify the requirements of the system and to study its high level functionality. In this paper a methodology for graphical user interface design is presented using the UML use case model. Given a use case diagram representing the actors and use cases of a system, and a set of activity diagrams describing each use case, the technique allows generation of a prototype of each user interface together with a set of GUI components. The technique handles the generalization relationships on use cases, in such a way that they are interpreted from the point of view of the GUI design **[**8**]**.

Another application is Automatic Source Code Generation using UML. Code generation using UML specifications concentrates on writing code constructs in target programming language. UML specifications contain calls to specific parameterized snippets leading to templates while retaining semantic description of model's requirements. The main aim of developing a code generator is the automatic generation of repetitive parts while the programmer can concentrate on specific code. It is helpful in enhancing the code quality and decreasing code length [10].
Software Components Testing using UML is another important application. Software component testability is a key characteristic that enables improvising software's quality and stability in the long run. Classically UML models are used for component design and implementation but may not prove to be efficient for model based testing [21].

An application of UML in Real Time Embedded Systems has also been considered. In the present scenario everything is directly or indirectly controlled by technical systems and machines. These systems are usually controlled by real time embedded systems. Embedded systems automate work and hence make day to day work easier and time saving. UML is very useful in modelling the real-time embedded systems. The graphical notation developed by UML can be used with object-oriented systems. As a modelling language it is used with embedded systems for developing numerous applications [1]
UML finds its application in the implantation of Relational Database as well. UML being a standard language for designing models and developing object oriented software applications can be extended to convert the relational dependencies present in relational database to UML constructs which are then implemented to build a software design.[11]
UML Modelling can be used in the compression of files. Compression is the technique of reducing the size of data to a size which is effective in storing and apt for representation. UML modelling can be applied to this field of science too by successfully creating software that can compress any size and any type of data [5].
UML Modelling of an API for Cloud Computing application development can be done. The application programming interface (API) is a major part of various branches of cloud computing including storage and is needed for efficient working of cloud services. UML can be used for development such models of API for cloud computing [7]. UML implementation can be used in Pattern Specification Techniques. Design patterns are used in software architecture and engineering for resolving commonly occurring design problems by using patterns that can be directly implemented into machine code. UML can be applied for designing and specifying these design patterns effectively [17**].**

As we can see that UML has a wide range of applications. We have also used UML for our application as it provides a great mechanism for visualization and documentation of software designs. Another advantage of UML is that it is independent of the programming language and can be implemented across a variety of platforms. It also makes maintaining and extending the software design easier. The static and dynamic behaviour of the software can be captured accurately using this approach.

3. REQUIREMENT GATHERING PHASE