A Systematic Framework for Structured Object Oriented Security Requirements Analysis

Sojan Markose, Xiaoqing (Frank) Liu*, and Bruce McMillin*
Computer Science Department
University of Missouri – Rolla
1870 Miner Circle
Rolla, MO 65409-0350
fliu@umr.edu

ABSTRACT
There are several approaches to elicit, analyze and specify security requirements ranging from formal mathematical models for proof of certain security properties to informal methods which are easily understood. Applicability of formal security models is limited since they are complex and it is time consuming to develop. On the other hand, informal security requirements analysis methods are not integrated with conceptual models in requirements analysis and they provide no process for analyzing both internal and external threats in a structured manner. This paper discusses a structured object oriented security requirements analysis methodology for the elicitation and analysis of security requirements. It is capable of identifying hierarchically both external and internal threats posed by both external and internal actors of a system level by level. It is illustrated and validated by security requirements analysis for an online banking system and an advanced power grid control system.

Categories and Subject Descriptors
D.2.1 **[Software Engineering]:** Requirements/Specifications – *elicitation methods, methodologies.*

General Terms
Security, Verification.

Keywords
Security requirements, use cases, misuse cases, structured object-oriented analysis, security goal, and security requirements.

1. INTRODUCTION
Non functional aspects of a software system such as security, safety, and reliability should be considered and incorporated in the software system along with the functional requirements throughout the software development process [12]. Among the nonfunctional requirements, security requirements are of great importance since security failure of a software system may incur important political, financial and military loses. However security needs of a software system are often considered only after the implementation begins. We need to analyze security needs early in the process and integrate countermeasures to security threats into the system. There is also a need for a method which can identify security requirements at multiple levels of a system to deal with both external and internal threats rather than specifying the security requirements for the system as a whole. The existing methods for security requirements analysis do not address the issue of analyzing both internal and external threats. They only deal with the external mis-users of the system.

Studies are going on across the globe to come up with effective methods for eliciting security requirements at the early stages of system development. Most studies are centered on extending use cases to analyze security requirements. Specifying security requirements depends on the complexity of the application [16]. A systematic process for a structured object oriented security requirements analysis and specification is needed since objects represent the assets of a system. A Structured object oriented methodology can be used to model security requirements. This paper i) studies a few noticeable research on extending use cases to elicit and analyze security requirements and ii) introduces a new methodology where security requirements elicitation and analysis is done in a structured manner using HOOMT[10] and misuse cases.

2. RELATED WORK
Security requirements are usually specified to prevent any activities that may pose a threat to either the stakeholders or the system itself. Researchers believe that activities that pose a threat to the software system can be effectively described using use cases by extending it to represent misuses and mis-users. These extended use cases are called misuse cases and are helpful in documenting negative scenarios which can be used to improve security by preventing them [3, 4, 5, 6]. The following represent some of the related research work on using misuse cases to analyze security requirements.

Sindre et al. developed a method to elicit security requirements using misuse cases [3, 4, 5, 6]. Use cases have actors who initiate them likewise, misuse cases also have mis-actors who initiate them. In this method use cases and misuse cases are specified in a single diagram. In addition to the "include" and "extends" relations between use cases in UML, new relations called "prevents" and "detects" were introduced in this method. The "prevents" relates a use case and a misuse case where the use case prevents the activation of the misuse case. The "detects" relates a use case and a misuse case where the use case detects the activation of the misuse case.

Ian Alexander developed a method for security requirements analysis using misuse cases in the context of trade-off analysis [7]. An 'exception' is considered to be an undesired event that

could cause a system to fail and misuse case analysis is the best way to find possible 'exceptions'. In this method to analyze a trade-off between the use cases and misuse cases two relationships; "threatens" and "mitigates" were introduced. Misuse case A threatens use case B if achieving the goal of A reduces the system's ability to achieve the goal of B. Use case A mitigates misuse case B if it reduces B's effects on the use case that it threatens. Two more relations "aggravates" and "conflicts with" are also introduced. Use or Misuse case A aggravates misuse case B if it increases either the probability of success or the seriousness of the damage that B threatens. Use case A conflicts with use case B if achieving A's goals makes achieving B's goal more difficult, and vice versa for B's effect on A.

John McDermott and Chris Fox proposed an abuse case model to capture and analyze security requirements. They define abuse case as 'a specification of a type of complete interaction between a system and one or more actors, where the results of the interaction are harmful to the system, one of the actors, or one of the stake holders in the system' [8]. They describe abuse cases using use case diagrams and use case descriptions. Use cases and abuse cases were distinguished by keeping them separate and by labeling the diagrams. Actors in abuse cases are described in detail because their characteristics such as resources, skills and objectives are important in understanding the abuse cases. In abuse case modeling, if an actor of a use case also acts maliciously, a new actor should be defined to represent that misbehavior. The authors also pointed out that since it is not sure where flaws will occur, an abuse case describes a family of undesirable interactions [9]. Finally abuse cases are usually developed through the exploitation of requirements oversights, design flaws and implementation flaws. The authors proposed to develop the abuse case model one step behind the use case model. Each component of the abuse case model is developed based on the corresponding component of the use case model. The five steps in a abuse case modeling process are i) identify the actors: identify all possible actors who might attempt harmful use of the system, ii) identify the abuse cases, iii) define abuse cases, iv) check granularity: abuse cases should be neither too detailed nor too abstract and abuse cases should have a uniform granularity of detail in its cases, and v) check completeness and minimality.

The above misuse case based approaches have provided a solid guideline for security requirements analysis. However, several important issues remain to be resolved, such as the integration of analysis of misuse cases and negative scenarios with conceptual models which describes assets of a system, such as object models; and the analysis of both external and internal threats in a structured way.

A goal driven framework was developed to analyze non-functional requirements [13]. An approach has been proposed to integrate it with conceptual models, such as object-oriented models in UML [12].

Sam Supakkul and Lawrence Chung propose a use case and goal-driven framework for integrating nonfunctional requirements with functional requirements [15]. In this framework nonfunctional requirements are represented as 'softgoals' which are to be 'satisficed'. To determine satisficeability, design alternatives or decisions are considered, design trade offs are analyzed, design rationale is recorded and design choices are made.

The above goal driven approaches have provided a general framework for nonfunctional requirements analysis. However, they have not addressed specific needs for security requirements analysis, such as the analysis of misuse cases, internal and external threats, and countermeasures.

The Sandia SCADA reference model [11] provides guidance in determining security requirements through a systematic approach. This framework is attractive in that vulnerabilities common to many systems are documented ranging from management practices and physical security to applications policies. These vulnerabilities are at a high level and are not decomposed into subsystem threats.

3. OUR APPROACH

Our objective is to develop a structured object-oriented security requirements analysis methodology to elicit and analyze security requirements. It improves the existing methods introduced in the previous sections by developing a structured object-oriented security requirements analysis process to identify both internal and external threats level by level through the system structure which is modeled by a high order object model based on a High Order Oriented Modeling Technique (HOOMT) [10].

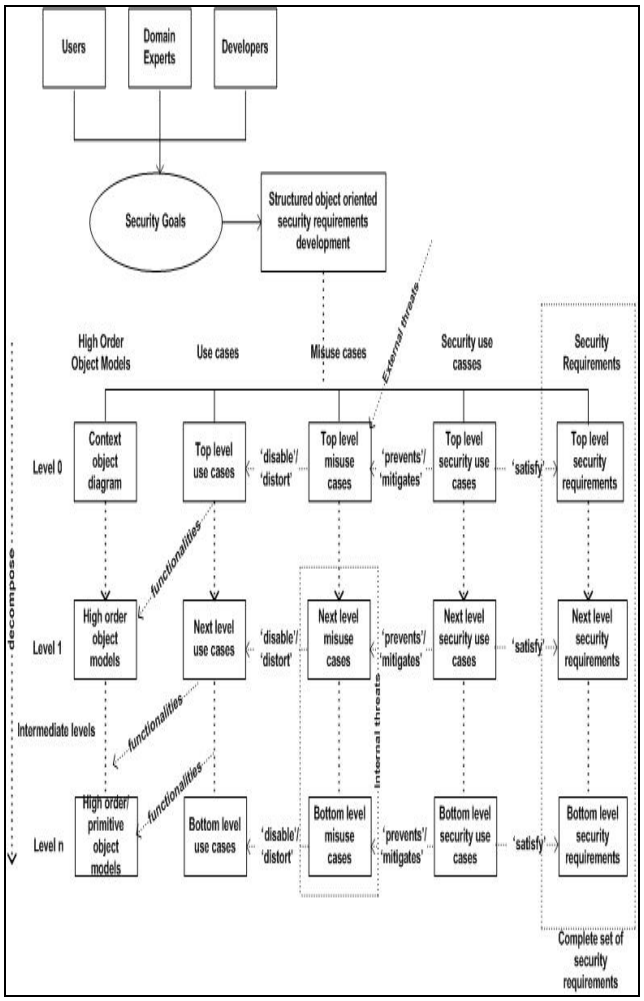

Figure 1: Structured object oriented security requirements process

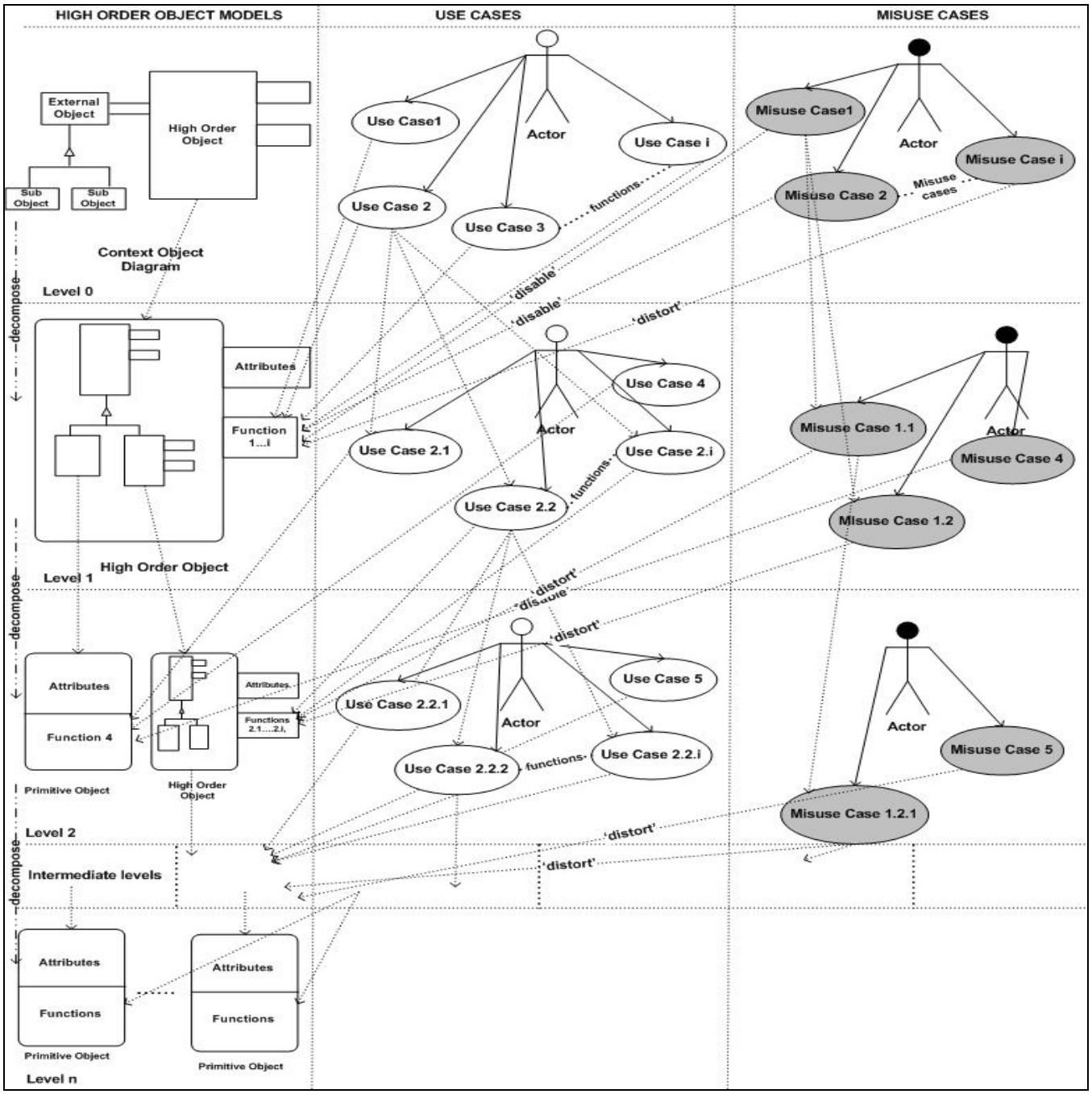

Figure 2: A detailed process for the hierarchical development and analysis of misuse cases

The HOOMT provides a structured object-oriented software design methodology which is based on hierarchical model development. The integration of structured methods with object-oriented methods provides the uniformity and reusability of the object-oriented approach with the hierarchical decomposition of objects, their functions, and their dynamic behaviors which are provided by the structured method.

In the structured object oriented security requirements process model the target system is decomposed using HOOMT. Figure 1 illustrates a structured object oriented security requirements process. In the first step a context object diagram is developed which shows the interactions between the high order system object and the external objects. In the second step, at each level of decomposition, use cases are specified to identify the main functionalities of an object. At the third step, misuse cases and mis-users which can cause harm to the functionalities represented by use cases are identified. At all levels, mis-users can be either external actors or internal actors. These use cases and misuse cases are related using two relations 'disable' and 'distort'. The 'disable' relates a use case and a misuse cases where the misuse cases completely disables a functionality represented by a use case. The 'distort' relates a use case to a misuse case where the

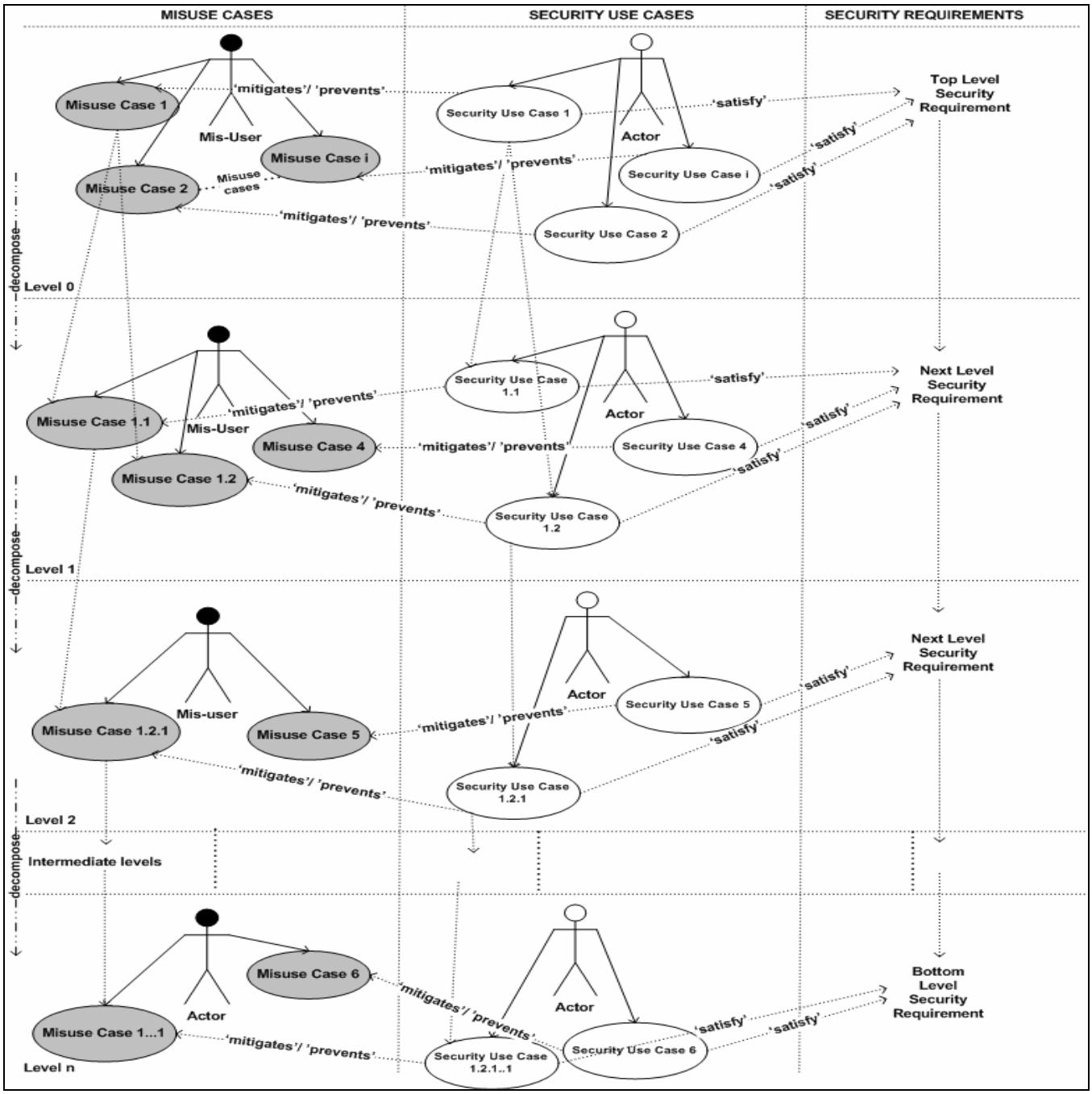

Figure 3: A detailed process for the hierarchical development and analysis of security requirements

misuse case distorts or change the functionality represented by the use case. At the fourth step security use cases are derived which act as counter measures to the misuse cases. These security use cases are related to the misuse cases using 'prevents' and 'mitigates' relations'. The 'prevents' relates a security use case and a misuse case where a security use case completely stops a negative scenario represented by a misuse case from happening. The 'mitigates' relates a security use case and a misuse case where a security use cases reduces the threat of a misuse case. Finally at step five security requirements are derived from security use cases. The security use cases and security requirements are related using a relation called 'satisfy'. The 'satisfy' relates a security use case and a security requirement where the security use case satisfies the derived security requirement. At each subsequent level of decomposition all the above five steps are repeated until a stage is reached where all the objects are primitive and further decomposition is either impossible or unnecessary. At the end of this structured object oriented security requirements process, we obtain a set of security requirements which are capable of securing the target system at each level of abstraction. A detailed process for the hierarchical development and analysis of misuse cases based on high order

object models and use case diagrams is shown in Figure 2. At the top level a context object diagram is developed which shows the interactions between the high order system object and its external objects. At this level a use case diagram is developed which represents the major functionalities of the system. These use cases becomes the 'Functions' component of the next level high order object model of the system object. Based on this use case diagram, a system level misuse case diagram is developed. These misuse cases are a threat to the functionalities represented by the 'Functions' of the system object. The relations 'disable' and 'distort' represent the relationship between misuse cases and the functions in the high order object and primitive object models. At each level of decomposition together with high order object models, new use cases are derived from higher level use cases. Use cases unique to that particular level are also identified. Misuse cases corresponding to use cases at each level of decomposition are also derived hierarchically which are then related to the 'Functions' of the high order objects and primitive objects.. Misuse cases at each level can be initiated by either external actors or internal actors. The decomposition process continues until a stage is reached where the objects are primitive and corresponding use cases and misuse cases are fully explored. Figure 3 represents a detailed process for the hierarchical development and analysis of security requirements based on misuse case diagrams and security use case diagrams. Starting from the top level, security use case diagrams are developed corresponding to the misuse case diagrams. These security use cases are related to the misuse cases using 'mitigates' and 'prevents' relations. At each hierarchical level security requirements are derived from security use cases. These security use cases are related to the security requirements using the 'satisfies' relation. The decomposition process is continued until a stage is reached where each misuse case at all levels is prevented or mitigated using security use cases and finally corresponding security requirements are derived.

4. APPLICATION EXAMPLES

4.1 Online Banking System (OBS)

An Online Banking System (OBS) is selected to demonstrate the proposed structured object oriented security requirements process model. A high order object model for the target system is built hierarchically by first developing a context object diagram.

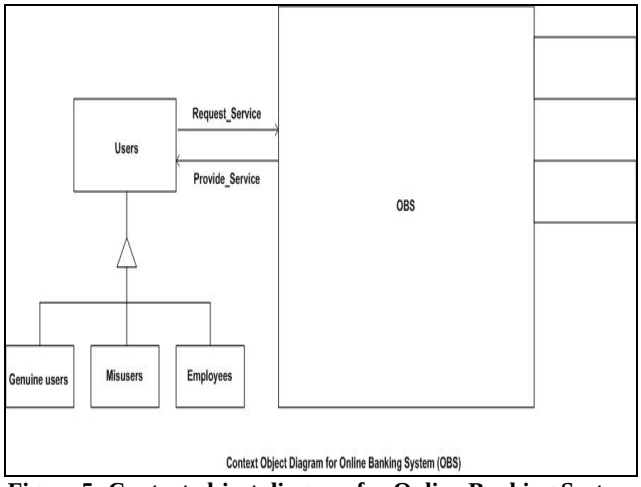

Figure 5: Context object diagram for Online Banking System

The context object diagram shows the interactions between the high order system object which in this target system is the 'OBS' and external object 'Users'. The context object diagram is then further decomposed into a set of components and their relationships in the following High Order Object Diagrams. The 'OBS' system object is decomposed into its high order objects such as 'Accounts', 'Owners', 'Transaction' and 'Services'.

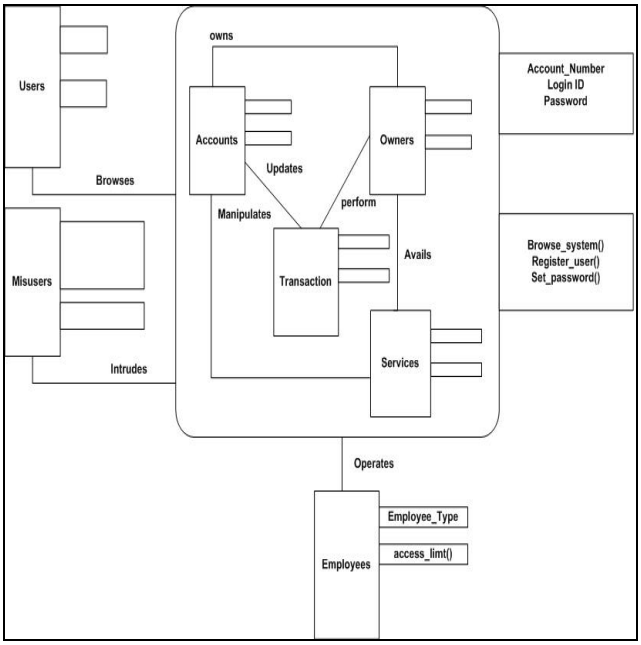

Figure 6: Decomposition of 'OBS' system object

The external object 'Users' is a generalization of sub objects 'Genuine users', 'Mis-users' and 'Employees'. Figure 6 and Figure 7 represent the High Order Object Diagram for the system object 'OBS' and its corresponding 'Use case-Misuse case' diagram.

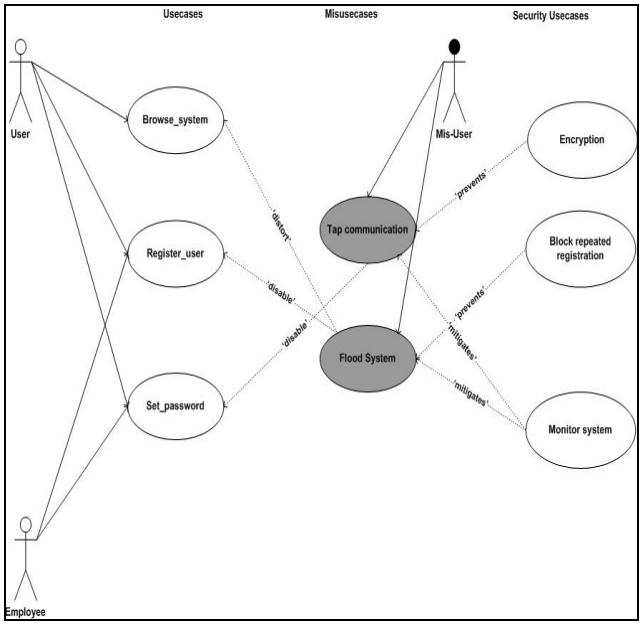

Figure 7: Use case-Misuse case diagram for 'OBS' system object

The OBS system object comprises of three significant use cases such as 'Browse_system', 'Register_user' and 'Set_password'. Two actors who initiate these use cases are also identified. They are 'User' and 'Employee'. Based on the use cases and actors, the possible mis-users and misuse cases at that level of hierarchy are identified. 'Tap communication' and 'Flood system' are the misuse cases initiated by the 'Mis-user'. Based on the identified misuse cases, counter measures are derived which are represented as security use cases in Figure 7.

Finally security requirements are developed from the security use cases. The following figure shows the top level security requirements for the target system.

Top Level Security Requirements:-

SR 1.1: The system must display user/ account activities only to respective users and employees.

SR 1.2: The system must monitor all user activities.

SR 1.3: The system must encrypt confidential information.

SR 1.4: The system must block repeated registration

Figure 8: Security requirements for OBS system object

At the second level of decomposition we first select the 'Accounts' high order object of the 'OBS' system object to decompose. Figure 9 represents its high order object model. At this level of decomposition we find out that 'Accounts' object have two sub classes 'Checking' and 'Saving' and its relationship with other sub objects at the same level and external objects. The 'Employee' actor is decomposed into 'Employee_Operator' and 'Employee_Manager' actors depending on the access rights to the target system. Figure 10 represent the 'Use case-Misuse case' diagram for the second level of decomposition. There are four important use cases initiated by the actors 'User' and 'Employee' for the high order object 'Accounts'.

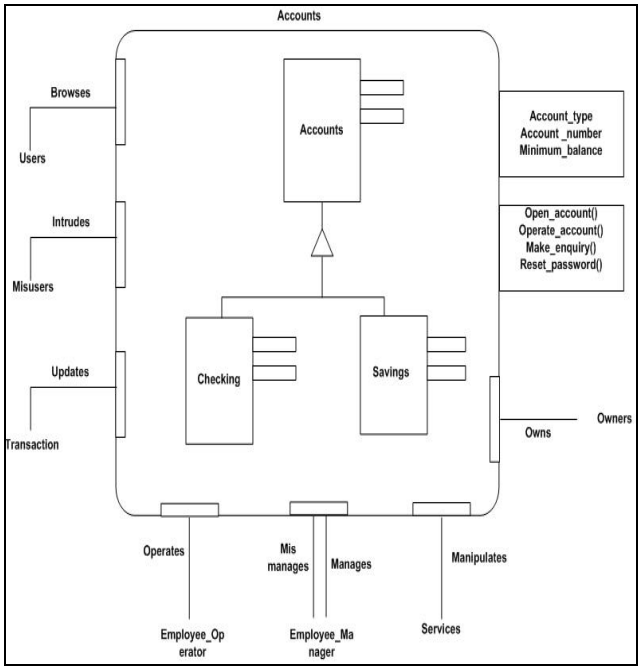

Figure 9: High Order Object Model for 'Accounts' object

The use cases include 'Open_account', 'Operate_account', 'Make_enquiry' and 'Reset_password'. Based on these use cases possible misuse cases such as 'Steal valid means of user identification and authentication', 'Disable password', and 'Transfer money' are identified. Based on the identified misuse cases, counter measures are derived which are represented as security use cases in Figure 10.

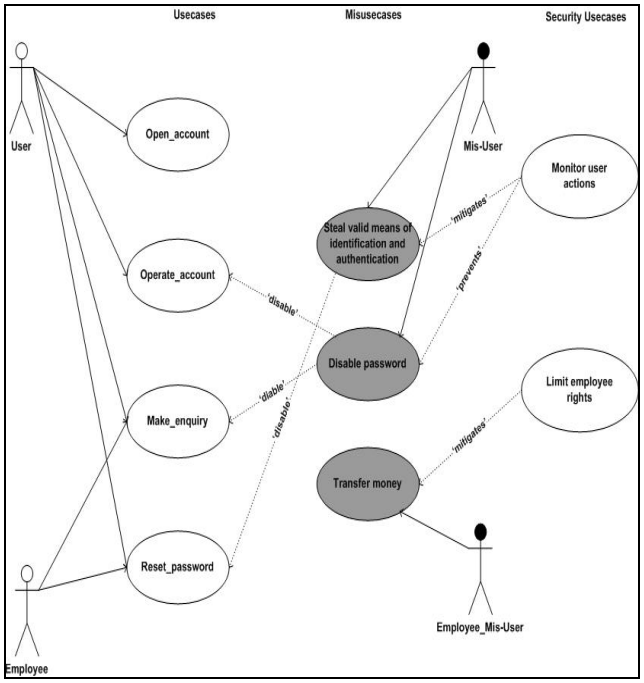

Figure 10: Use case-Misuse case diagram for 'Accounts' object

At the second level of decomposition along with 'Accounts' high order object, there are two primitive objects such as 'Services' and 'Owners', of the 'OBS' system object.

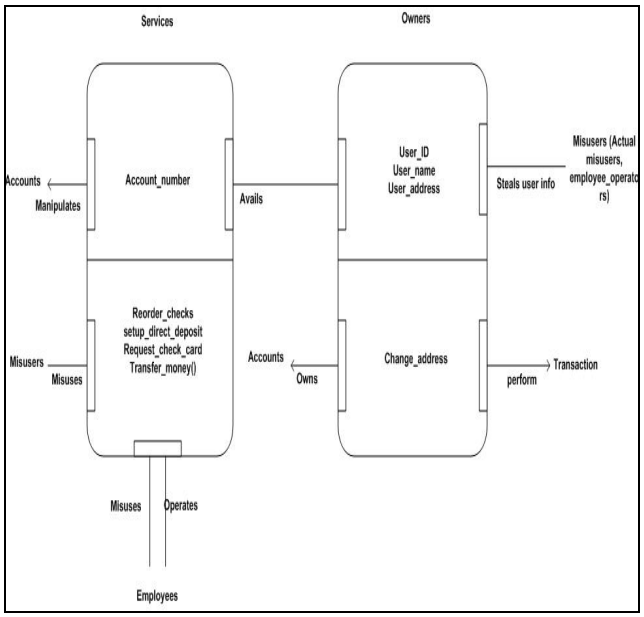

Figure 11: Primitive Object Model for 'Services' and 'Owners' object

Figure 11 represents the primitive object model of the 'Services' and 'Owners' primitive objects. It also shows the interrelationships between the objects and the actors. These sub objects are set as the primitive objects for this sample online banking system and they are not further decomposed. The following figure represents the 'Use case-Misuse case' diagram for the second level of decomposition.

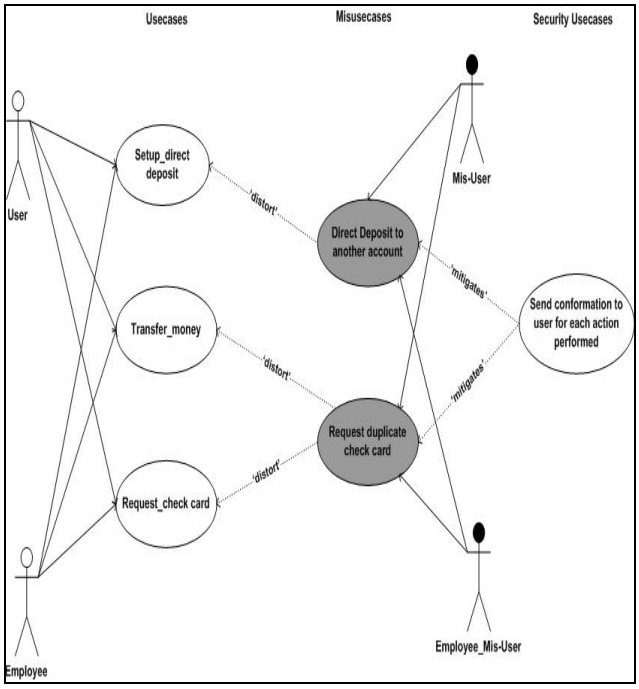

Figure 12: Use case-Misuse case diagram for 'Services' and 'Owners' object

The 'Services' and 'Owners' primitive objects consist of three important use cases initiated by the actors 'User' and 'Employee'. The use cases include 'Setup_direct_deposit', 'Transfer_money', and 'Request_check_card'. Based on these use cases possible misuse cases such as 'Direct deposit to another account', and 'Request duplicate check card' are identified. At this primitive level the use cases can also become misuse cases when initiated by a mis-user. Based on the identified misuse cases, counter measures are derived which are represented as security use cases in Figure 12. Finally Security requirements are developed from the security use cases. Figure 13 shows the security requirements for the target system at the second level of decomposition for the high order and primitive objects.

Level 2 Security Requirements:-

SR 2.1: The system must display valid means of user identification and authentication to respective users only

SR 2.2: The system must monitor all user activities.

SR 2.3: The system must encrypt confidential information.

SR 2.4: The system must limit employee access based on their clearance level

SR 2.4: The system must send confirmation message to the user for each action performed.

Figure 13: Security requirements for 'Accounts' high order object, 'Services' and 'Owners' primitive objects

The 'Structured Security Requirements Process Model' is successfully applied to the sample online banking system. As a result we are able to identify the different objects within the system, the different actors such as users and mis-users of the system and the possible misuse cases that can cause harm to the target system. Finally based on the misuse cases we identified few countermeasures which are represented as the security use cases. These security use cases are further transformed into security requirements. With the help of this sample online banking system case study, it is clear that the proposed 'Structured Object Oriented Security Requirements Process Model' can be implemented on other software systems and real time systems involving both software and hardware components.

4.2 FACTS Power System

Control of power networks is a tedious task because of its sheer size. They are vulnerable to contingencies like line failure. A combination of such contingencies may lead to a cascading power failure. The family of "Flexible Alternating Current Transmission System" (FACTS) devices shows promise for use as network-embedded controllers [1, 2]. There is ongoing research to incorporate a number of FACTS devices into a power grid network to act as a distributed, fault-tolerant, and real-time constrained control system. The 'Structured Object Oriented Security Requirements Process' is implemented on the FACTS power system to identify the misuse cases and mis-users that can cause harm to the power system and finally derive security requirements.

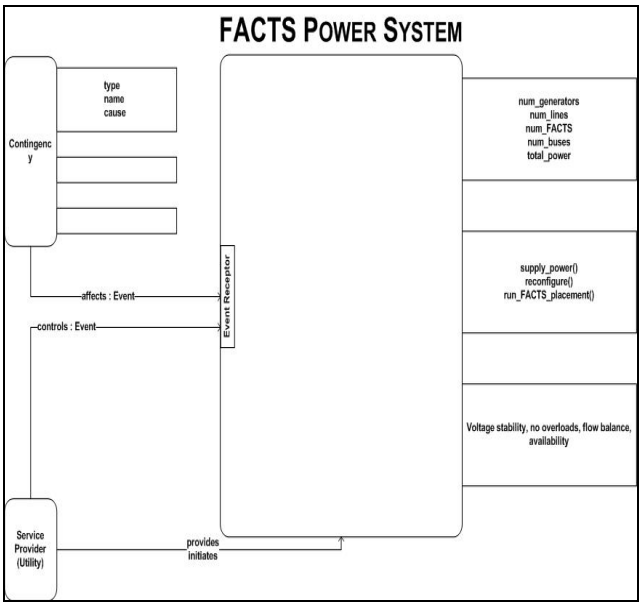

Figure 14: Context object diagram for Facts Power System

The high order object model for the target system was previously developed as part of a research on object-oriented co-analysis/co-design of the FACTS power system [14]. Figure 14 shows the context object diagram for the FACTS power system. The context object diagram shows the interactions between the high order system object which in this target system is the 'FACTS Power System.' and external objects 'Contingency' and 'Service Provider'. The context object diagram is then further decomposed into a set of components and their relationships in the following High Order Object Diagrams. The 'FACTS Power System'

system object is decomposed into its high order objects such as 'Facts device', 'Placement' and 'Power transmission system'.

The following figures, Figure 15 and Figure 16 represent the high order object diagram for the system object 'FACTS Power system' and the 'Use case-Misuse case' diagram of the corresponding level of hierarchy respectively.

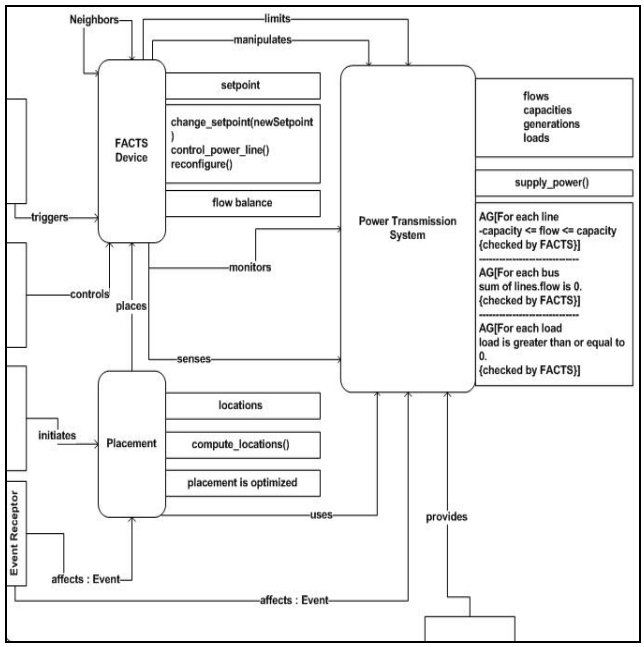

Figure 15: Decomposition of 'Facts Power System' object

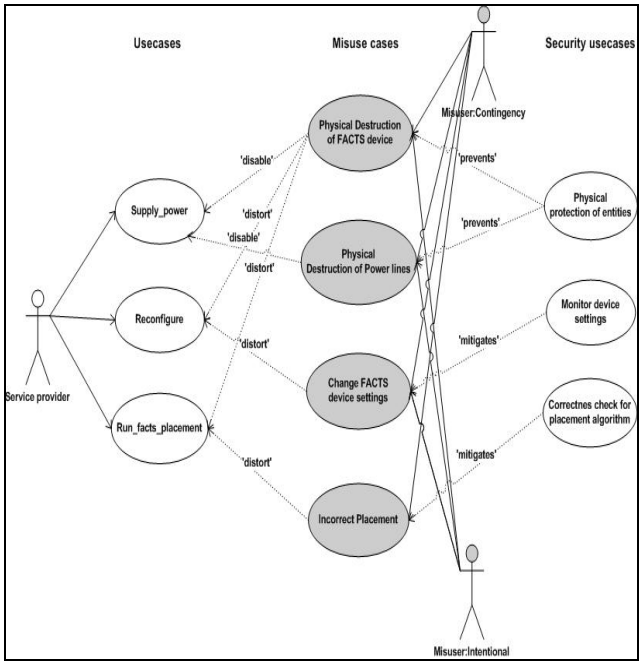

Figure 16: Use case-Misuse case diagram for 'Facts Power System' object

At the 'FACTS Power System' object level four significant misuse cases are identified such as 'Physical destruction of facts device', Physical destruction of power lines', 'Change facts device settings' and 'Incorrect placement'. These misuse cases are initiated by the mis-users 'Contingency' and 'Intentional'. Based on the identified misuse cases, counter measures are derived which are represented as security use cases in Figure 16.

Finally security requirements are developed from the security use cases. The following figure shows the top level security requirements for the target system.

Top Level Security Requirements:-

SR 1.1: The system must provide physical protection to all physical entities

SR 1.2: The system must monitor device settings

SR 1.3: The system must perform correctness check on placement algorithm

Figure 17: Security requirements for 'Facts Power System' object

At the second level 'UPFC Facts device' object is decomposed into following four sub objects, 'DSP board', 'Embedded computer', 'Interface board' and 'UPFC Power electronics'. Figure 18 represents the high order object model for 'UPFC Facts device object'. It shows the relationship between the four sub objects.

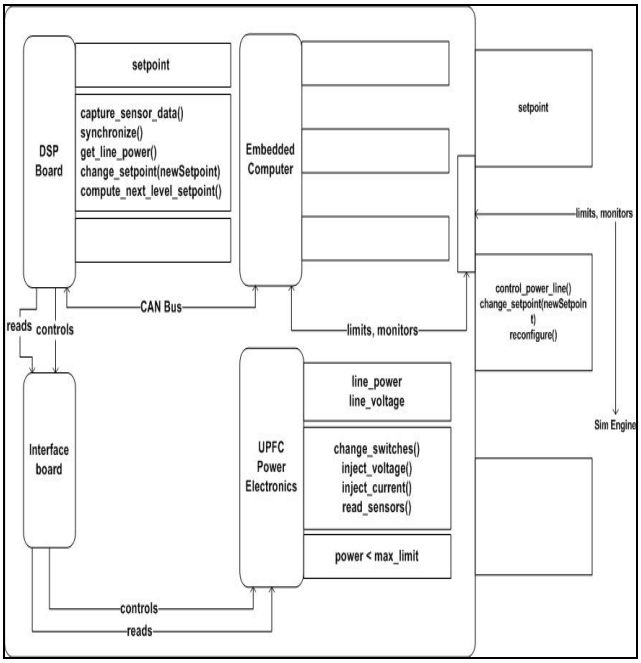

Figure 18: Decomposition of 'UPFC Facts Device' object

The 'UPFC Facts Device' object have three major use cases such as 'Control_power_line', Change_set_point', and 'Read _sensor_data'. The mis-users 'Contingency' and 'Intentional' initiates three misuse cases such as 'change set point of long term control', 'change control point in dynamic control', 'Change sensor data in DSP board'. Based on the identified misuse cases, counter measures are derived which are represented as security use cases in Figure 19. Finally Security requirements are developed from the security use cases. Figure 20 shows the security requirements for the 'Facts UPFC Device' object level. At the third level 'Embedded computer' object is decomposed into 'Dynamic control' and 'Long term control' sub objects.

Figure 21 represents the high order object model for the 'Embedded computer' object.

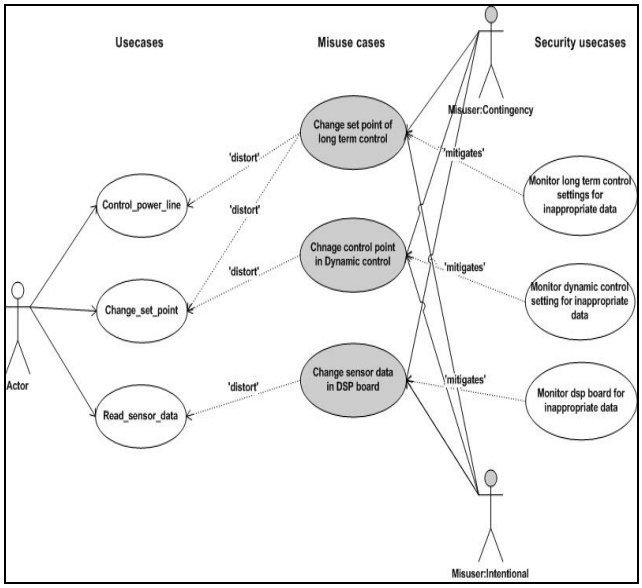

Figure 19: Use case-Misuse case diagram for 'UPFC Facts Device' object

Level 2 Security Requirements:-

SR 1.1: The system must monitor long term control for inappropriate data

SR 1.2: The system must monitor dynamic control for inappropriate data

SR 1.3: The system must monitor dsp board for inappropriate data

Figure 20: Security requirements for 'UPFC Facts Device' object

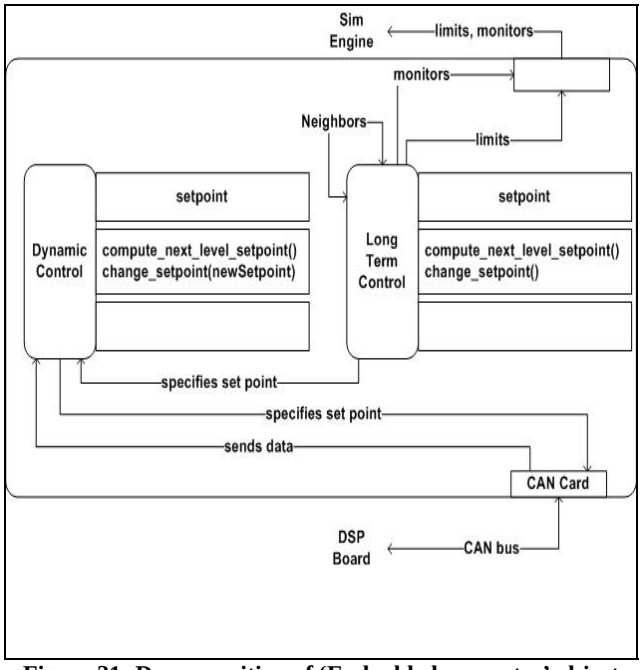

Figure 21: Decomposition of 'Embedded computer' object

One of the major use cases in 'Embedded computer' object is 'compute_next_level_setpoint'. The mis-users 'Contingency' and 'Intentional' initiates three misuse cases such as 'Randomly lose flow messages', 'Alter all flow messages', Randomly invert accept/reject messages', and 'Increase edge flow'. Based on the identified misuse cases, counter measures are derived which are represented as security use cases in Figure 22. Finally Security requirements are developed from the security use cases. Figure 23 shows the security requirements for the 'Facts UPFC Device' object level.

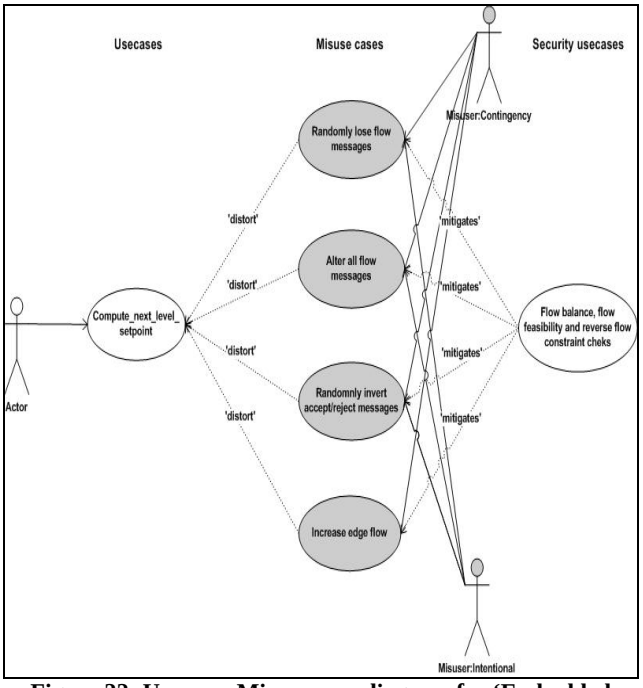

Figure 22: Use case-Misuse case diagram for 'Embedded computer' object

Level 3 Security Requirements:-

SR 1.1: The system shall perform assertion checking or constraint checking on the state of the system.

Figure 23: Security requirements for 'Embedded computer' object

5. CONCLUSION

Security requirements are of great importance these days as every software system is under threat. Although existing methods have provided a useful guideline for security requirements analysis, several important issues remain to be addressed including integration of security requirements analysis methods with conceptual models such as object models of system assets, and analysis of both internal and external threats in a structured manner. This paper successfully addresses these issues by proposing a structured object oriented security requirements process in which security requirements for the entire system is derived level by level in a structured manner. This methodology is capable of analyzing security requirements by identifying threats posed by both external and internal actors of a system. Object oriented nature of this methodology helps in identifying the assets of a system which are basically the system objects. The proposed methodology is successfully demonstrated using a sample online

banking system to identify the misuse cases and derive security requirements at each level of hierarchy. To test the applicability, the proposed methodology is also applied to a real time FACTS power system. As a result misuse cases at three different levels of hierarchy could be identified. Counter measures for these misuse cases are identified as security use cases. Finally security requirements are derived based on the security use cases. Security requirements derived at each level of hierarchy when grouped together represents the security requirements specification for the entire system.

6. ACKNOWLEDGEMENTS

*This work is supported in part by NSF MRI grant CNS-0420869

7. REFERENCES

[1] Austin Armbruster, Mike Gosnell, Bruce McMillin, Mariesa Crow. "Power Transmission Control Using Distributed Max Flow". Proceedings of the 29th IEEE Annual International Computers Software and Applications Conference. Edinburgh, U.K, June 2005.

[2] B. McMillin, M. L. Crow. "Fault Tolerance and Security for Power Transmission System Configuration with FACTS Devices," Proceedings of the 32rd Annual North American Power Symposium, vol. 1, Waterloo, Ontario, October 2000.

[3] Guttorm Sindre, Andreas L. Opdahl, "Eliciting Security Requirements by Misuse Cases". Proceedings of the 37th International Conference on Technology of Object oriented languages and systems. Sydney, November 2000.

[4] Guttorm Sindre, Andreas L. Opdahl, "Capturing Security Requirements through Misuse Cases". Proceedings of the 14th annual Norwegian informatics conference. Norway, 2001.

[5] Guttorm Sindre, Andreas L. Opdahl, "Templates for Misuse Case Descriptions", Proceedings of the 7th international workshop on requirements engineering:Foundation for software quality. Switzerland. June 2002.

[6] Guttorm Sindre, Donald G. Firesmith, Andreas L. Opdahl, "A Reuse-Based Approach to determining Security Requirements". Proceedings of the 9th International workshop on requirements engineering:Foundation for software quality. Austria, June 2003.

[7] Ian Alexander, "Misuse Cases: Use cases with Hostile Intent", Software IEEE. January 2003.

[8] John McDermott, Chris Fox, "Using Abuse Case Models for security Requirements Analysis". 15th Annual computer security applications conference. Arizona. December 1999.

[9] John McDermott, "Abuse-Case-Based Assurance Arguments". 17th Annual computer security applications conference. New Orleans, December 2001.

[10] X. F. Liu, H. Lin, and L. Dong. "High Order Object Oriented Modeling Technique for Structured Object-Oriented Analysis." International Journal of Computer and Information Science (IJCIS), June 2001.

[11] D. Kilman, J. Stamp. "Frame Work for SCADA Security Policy". Technical Report SAND 2005-1002C. Sandia National Laboratories Albuquerque, NM 87185-0785.

[12] L. Marcio, J. Cesar Samprio do Prado Leite, "Nonfunctional Requirements: From Elicitation to Conceptual Models", IEEE transactions of software engineering. May 2004.

[13] J. Mylopoulos, L. Chung, B. A. Nixon. "Representing and Using Nonfunctional Requirements". IEEE Transactions on Software Engineering, 1992.

[14] M. Ryan, S. Markose, Y. Cheng, X. F. Liu, B. McMillin. "Structure Object Oriented Co-analysis/Co-design of Hardware/Software for the FACTS Power System". Proceedings of the 29th IEEE Annual International Computers Software and Applications Conference. Edinburgh, U.K, June 2005.

[15] S. Supakkul, L. Chung. "Integrating FRs and NFRs: A Use Case and Goal Driven Approach". Proceedings of the 2nd International Conference on Software Engineering Research, Management & Applications (SERA'04), Los Angeles, May 5 - 7, 2004.

[16] T. C. Ting. "Modeling Security Requirements for Applications". Proceedings of the 8th annual conference on Object-oriented programming systems, languages, and applications, Washington, D.C, 1993