I J C T A, 9(2) 2016, pp. 477-495
© International Science Press

FORBID Feature Oriented Role Based Access Control Permission Definition

K. Shantha Kumari[1], Tamizhmani[2] and Abitha[2]

SUMMARY

In multi-user information systems present in business enterprises, users/ employees are permitted / restricted to access enterprise' system resources. These access permissions are assigned to the users based upon the Role(s) they hold in the enterprise and are referred as Role based access control permissions [RBAC Permissions]. Existing Research in secure software engineering have identified that when these RBAC permissions definitions are analysed and included at the design phase of software development, the implementation of overall information system's security is improvised. Hence, many design abstractions such as classes, aspects, features etc. are found to be used for abstracting RBAC permissions in the existing research works. Either RBAC permissions are defined using a single access control operation or with multiple operations that constitutes the granularity of RBAC permission. Similarly, a RBAC permission may be specified with or without the details of associated roles / resource sets on which it is applicable- which details constitutes for the level of abstraction of RBAC permission. In addition, there is a possibility that RBAC permission may be enforced statically or dynamically which constitutes its type of enforcement.

RBAC Permissions may evolve in terms of its level of granularity, level of abstraction and type of enforcement, due to the changes in organization's business process, new market demands, growing customer's expectations, increasing security threats etc. A suitable modeling paradigm and process based on it is required to handle the evolutions in RBAC permissions unambiguously and systematically. As a result of exhaustive literature review, modeling paradigm 'Feature' is found to fulfill the above given requirement when it is ameliorated and in this paper, a ameliorated Feature is proposed to model the RBAC permissions.

Keywords: Features, Modeling, RBAC Permissions

I. INTRODUCTION

Role based access control (RBAC) is defined as an access control method which is implemented in an organization to ensure that the information is accessed by 'User(s)' based only on the responsibilities that they hold in the organization. A 'Role' abstracts these responsibilities which are defined for any job profile in the organization e.g. Finance Manager is a role which is responsible for handling the financial dealings. 'Permission(s)' are provided to Roles in order to access various system 'Object(s)' (files, tables, programs etc.) based on their responsibilities [1]. These RBAC permissions actually represent the required access control 'Operations' (read, append, modify etc.) to be imposed on the roles to access the system resources [2, 3] to execute the role's responsibilities.

In existing research works, a RBAC permission is defined and modelled with respect to its functional description related to Roles and Objects ignoring its procedural description viz. number of operations used to implement the permission (Granularity), their implementation details (Abstraction) and their implementation mode (Enforcement). By ignoring the RBAC permission's procedural description, there is

[1] Associate Professor, [2] Post Graduate Scholars
Department of Computer Science and Engineering, Rajiv Gandhi College of Engineering and Technology, Pondicherry

a high probable chance for misinterpreting and ambiguously modelling the variants of RBAC permissions as explained below:

1. A RBAC permission for e.g. “Read” may be specified to read a table. However, this permission can be implemented either to read a table wholly or read a particular column/row/cell alone.
2. A RBAC permission, for e.g. “Edit” may be defined either as composite permission with functional aggregates – “cut”, “copy” and “paste” or with its implementation variants as –”Edit a file”, “Edit a source program” etc.

Existing modeling paradigms ambiguously interpret these permission definitions as same and this would lead to erroneous implementation.In order to avoid such ambiguity and misinterpretations, the need for suitable modeling paradigms to model all variants of RBAC permissions unambiguously and completely stands emphasized.

Towards this objective, a detailed literature review on prominent modeling paradigms for modelling access control permissions was carried out in [4]. Major modeling paradigms which are used for this purpose include classes, class templates, UML profiles, UML patterns, Aspects andFeatures. The outcome of the review brought out their limitations in modelling access control permissions. These paradigms majorly model access control permissions at a higher abstraction level ignoring their procedural description. In addition, the abstractions are comparatively analysed to know their capabilities in modelling RBAC permissions (including their granular definition, abstraction details and enforcement). From this analysis, the modeling paradigm ‘Feature’ is found to be more capable than other abstractions in modelling RBAC permissions. However, the modeling paradigms ‘Feature’ with its present definition[4] is unable to suitably model every kind of RBAC permissions with respect to granularity, abstraction and enforcement. Hence, there is need to ameliorate its conceptual definition and this research work identifies and enforces these ameliorations, to render the Features suitable to model the RBAC permission variants unambiguously and completely.

II. FEATURE & FEATURE MODELS

A. Feature

In software product line engineering, a Feature is a functionality or an implementation characteristic that is important to a client. Furthermore, Features indicate capabilities of the system; these capabilities fulfill both the functional and non-functional requirements of the software [5]. The Feature definitionis given in Figure 1 and the attribute explanation is same as [6, 7, 8]

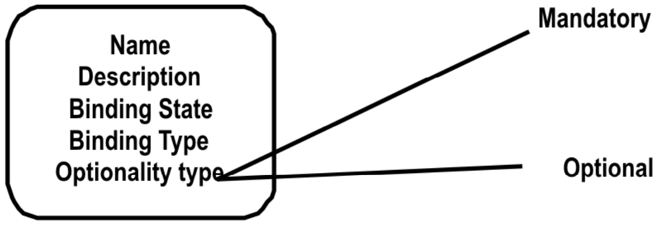


Figure 1: Existing Feature Definition

B. Feature Model

The Features, when identified and hierarchically organized as a model- Feature model serves as a domain analysis tool [9]. The Feature models hold three types of information: Features, Feature relationships, and Feature constraints [4].

Feature relationships: The link between a Parent Feature (source) and its Children Feature (destination) is called a feature relationship. Various Feature relationships are as follows:

1. Mandatory feature relationship (All or None) - The mandatory children features have to be selected whenever its parent is called for new product configuration.
2. Optional feature relationship (Zero or Many) -The optional children features may or may not be selected whenever its parent is called for new product configuration.
3. XOR feature group (Only one) - Only one child feature should be selected from a mutually exclusive children feature group, whenever its parent is called for new product configuration.
4. OR feature group (One or Many) - None or One or many children features can be selected from this feature group, whenever its parent is called for new product configuration.

Feature dependencies: These dependencies define which features can or cannot coexist and together in the same product. There are two Feature dependencies are given: Requires and Excludes.

1. Requires: Presence of a given Feature requires the inclusion of another Feature
2. Excludes: Presence of a given Feature requires the exclusion of another Feature

"Feature model" always includes all Features of the domain that may not be currently mapped to any system requirement, but may be eventually realized due to some future evolutionary changes or customizations [10]. A simple example for Feature model adapted from [5]is given in Figure 2.

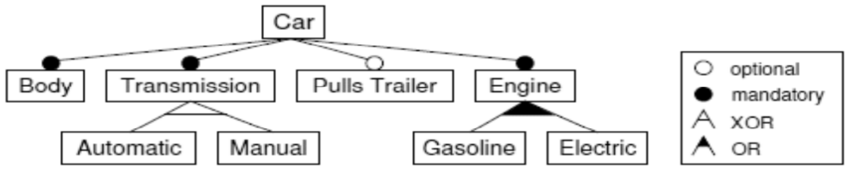


Figure 2: Illustration of a Feature model based on [11]

In the above given example, Body, Transmission, Engine are mandatory features and Pulls Trailer is an optional feature. Automatic and Manual form the XOR group children of Transmission. Similarly, Gasoline and Electric form the OR group children of Engine. Whenever the Transmission of Car is Manual- it "requires" Gasoline Feature too.

(C) Feature and RBAC Permissions

As RBAC permissions are meant to satisfy the user's requirements and also indicate the system functionality, they can be easily abstracted as Features. The mandatory and optional Features help to model RBAC permissions. The variability among various RBAC permissions are explored by arranging them as a Feature model. This would help to model future evolutionary changes in RBAC permissions too.

(D) Limitations in Existing Feature Definition

The detailed literature review provided in [12] helped to identify that "Features" are the most suitable design level abstraction to model RBAC permissions appropriately. Further study on Features and Feature oriented approaches in [13] brought out certain limitations with respect to their definition, relationships and constraints in modelling the RBAC Permissions. This section is devoted in elaborating those limitations. In addition, the required ameliorations in conceptual Feature definition, Feature relationships and their constraints are also identified.

Limitation 1: Features are defined always with their Optionality Type”. However, specifying the optionality type at the time of ‘Feature’ definition will have the following drawbacks:

- It will limit the capabilities of the Feature to evolve into many possible variants [14] during implementation.
- Changing the optionality type of a Feature from ‘mandatory’ to ‘optional’ is not possible. Hence it will limit the capability of the Feature to undergo any change in Future

Ameliorations identified

- *Exclusion of ‘ Optionality type’ Feature attribute*
- *Introduction of ‘granularity [simple / composite]’ and ‘category[Concrete / Abstract]’ attribute*

Limitation 2: In Feature definition [6, 11] approach only the ‘decomposition’ relationship is used to model the relationships between any two Features. However, Features can be related by means of other different relationships also. The following example illustrates the need for exploring the other relationships in Feature definition: **Example:** The whole Feature – ‘Edit’ in figure 3is constituted with the corresponding functional parts i.e. ‘Cut’, ‘Copy’ and ‘Paste’.

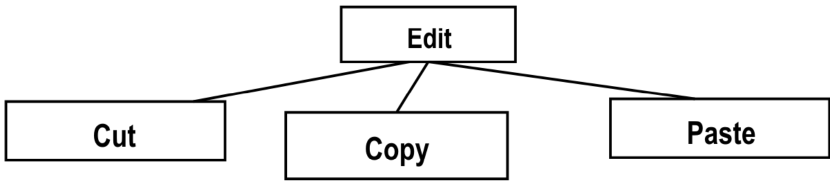


Figure 3: Composition (whole-part) relationship of Edit Feature

Alternatively, the Feature “edit” can be specialized to edit a cell or column or a row. This is illustrated in figure 5.

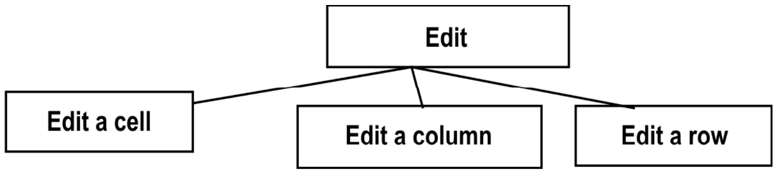


Figure 3: Specialization (Variants) of Edit Feature

AMELIORATION IDENTIFIED

- ***Inclusion of ‘Specialization’ relationship***

Limitation 3: Existing Feature definition[6] use only whole-part aggregation relationship that allows the AND grouping among Features i.e. ‘All or none’ group. For e.g. consider the previously described ‘Edit’ Feature. Either the whole Feature “Edit” inclusive of its every functional part (sub-Features) is selected or it is not selected for policy configuration. However, the Features can also be related by means of *XOR (1 in ‘n’ sub-Features) and OR group (‘m’ in ‘n’ sub-Features) in* Whole-part composition relationship.

Amelioration identified

- ***Inclusion of XOR and OR groups***

Limitation 4: The existing Feature definition [6]uses two cross hierarchical Feature constraints- “requires” or “excludes”. These constraints are enough to relate any two whole Features that would be either required

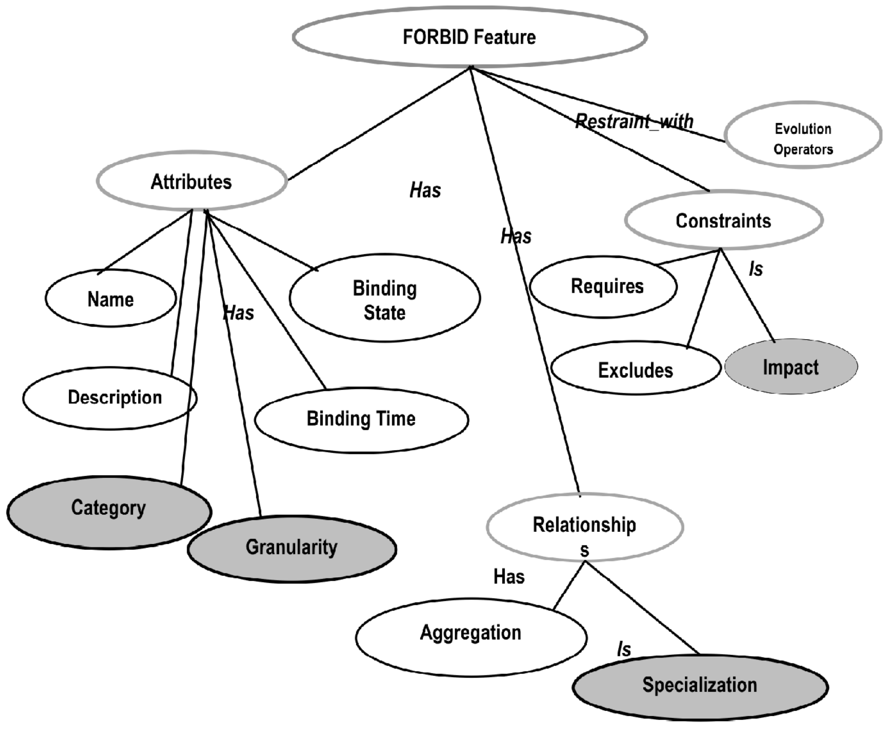


Figure 5: FORBID Feature Ontology

or excluded completely. However, there is a need for a new constraint to model the impact of one feature's selection on another.

Amelioration identified

- ***Introduction of "impacts" constraint***

Limitation 5: The existing Feature definition used in [6, 11] does not provide any process for the identification of Features from the access control requirements, methodical construction of Features model and systematic handling of evolutions.

Ameliorations identified

- ***Introduction of a Feature oriented process for Feature identification, Feature model construction and Feature evolution handling***

Limitation 6: In order to handle all possible RBAC Permission evolutions unambiguously and completely i.e. changes with respect to levels of granularity, levels of abstraction and types of enforcement, there is a need for suitable evolution operators in the Feature definition.

Amelioration identified

- The Feature evolution operators that are required to model the evolutionary changes in FORBID features are ADD, DELETE, MERGE, SPLIT and CHANGE. These evolution operators are inspired from the works available in [15], [16], [17], [18], [19].

Thus, the Feature given in meta-model [11] is now augmented with new Feature attributes, relationships and constraint. The ameliorated feature is called as FORBID feature [**F**eature **O**riented **R**ole Based Access Control Perm**i**ssion **D**efinition].

III. FORBID FEATURE

The ontology of the conceptual definition of the FORBID feature that resulted by augmenting the identified ameliorations are as follows:

The new attributes, relationships, and constraints are highlighted in figure 5. Mathematical definition of every component in FORBID Feature follows below. The mathematical definition is represented using Propositional logic notations [20].

(A) FORBID Feature Attributes

The FORBID feature attributes, which are obtained after applying the identified ameliorations, are illustrated in figure 6.

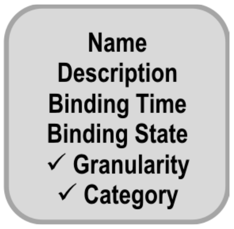


Figure 6: FORBID Feature Attributes

FORBID feature is defined as follows:

Definition 1: *A FORBID Feature* ***(f)*** *is defined with following attributes:*

- *Name - Mnemonic representation of an RBAC permission. This attribute is used to manage the naming and versioning of the Features.*
- *Description - Brief summary of RBAC permission.*
- *Binding time - {Reuse-Time, Compile-Time, install-time, load-time...}*
- *Binding state - {Bound, removed , undecided}*
- *Granularity - {Simple, Composite}-Simple Features are the atomic units of functionality and composite Features are built using the simple Features.*
- *Category - {Concrete, Abstract} - Concrete Features represent the concrete logical or physical characteristic of the software system.Abstract Features do not have real implementation, but it is a generalization of more specific concrete or abstract Features* [4] .

By combining the values of the newly included attributes, it is possible to obtain different FORBID features that can model different kinds of RBAC permissions as given in Table 1.

Table 1
FORBID Features and their mathematical definitions

No	*FORBID Feature Type*	*Mathematical definition*
1	Simple concrete FORBID Feature	f_{sc}
2	Simple abstract FORBID Feature - • Obtained due to specialization of multiple simple concrete FORBID features	$f_{sa} \xrightarrow[Is\ A]{} \{f_{sa_i} : (\mathrm{P} = f_{sa}) \wedge (\mathrm{Ch_i} = f_{sc_i}, \mathrm{i} = 1 \ldots \mathrm{n})\}$
3	Composite concrete FORBID Feature • Obtained due to aggregation of multiple simple and composite concrete FORBID features	$f_{cc} \xrightarrow[Has\ A]{} \{f_{cc_i} : (\mathrm{P} = f_{cc}) \wedge ((\mathrm{Ch_i} = f_{sc_i}) \vee (\mathrm{Ch_i} = f_{cc_i}))\ \mathrm{i} = 1 \ldots \mathrm{n})\}$
4	Composite abstract FORBID Feature • Obtained due to specialization of multiple simple and composite concrete FORBID features	

f – FORBID feature with attributes**P** – Represents any Non-terminal RBACP permission feature**n** - represents the number of children of P**{Ch$_1$ ... Ch$_n$}**– represents the set of children in RBACP permission feature P.

(B) FORBID Feature Relationships

The following three new FORBID feature relationships are introduced:

1. XOR and OR group Aggregation and
2. Generalization / Specialization.
3. Interaction relationships

The FORBID feature relationships are given in figure 7:

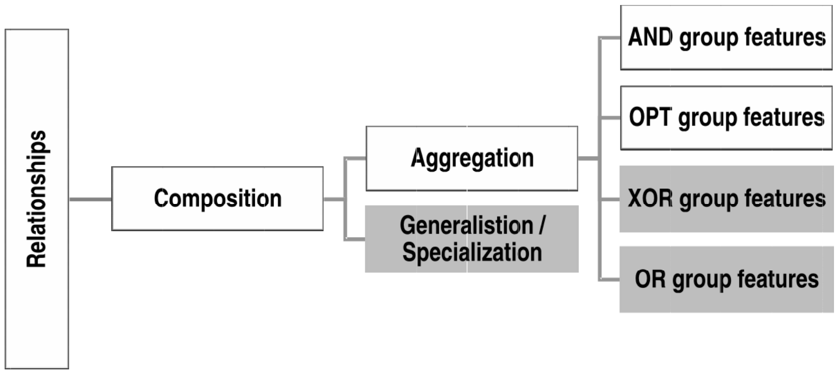


Figure 7: FORBID Feature Relationships

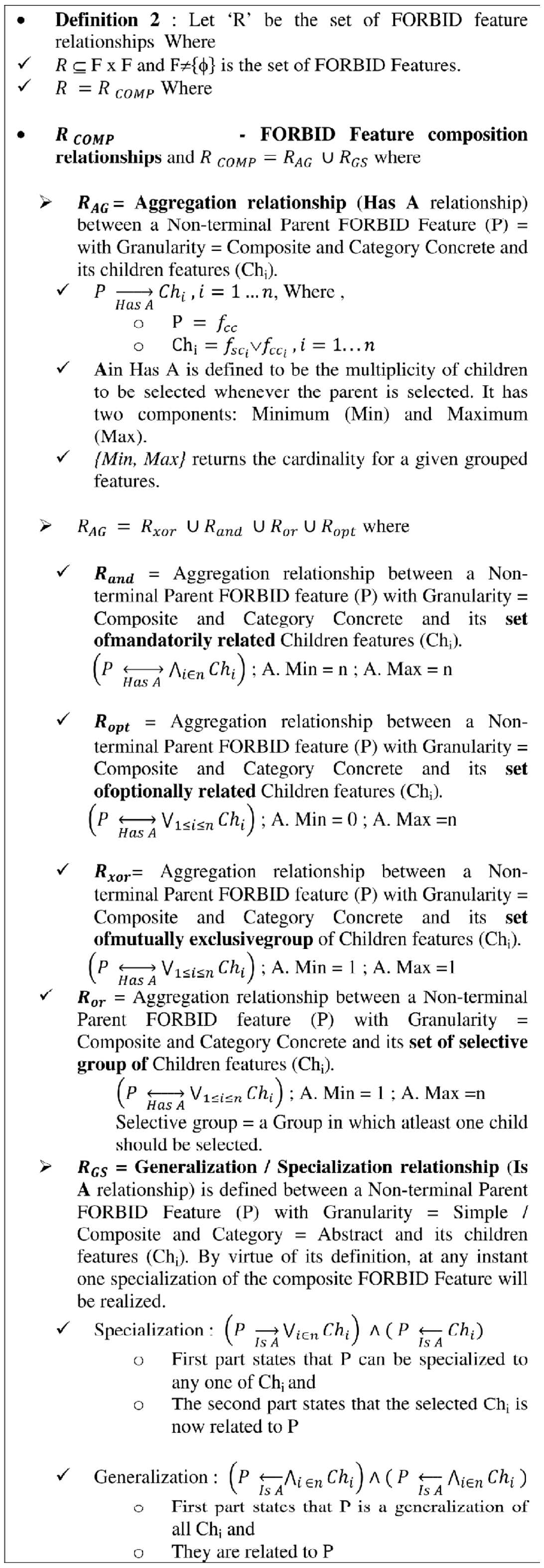
The explanations for the FORBID feature relationships are as follows:

Composition Relationships - These relationships focus on the hierarchical compositions between Composite FORBID Features and its constituent Simple FORBID Features. The various types of composition relationships are given as follows:

➢ ***Aggregation (Functional Composition) Relationships*** - By this relationship, FORBID features are composed as aggregation of its functional components. This relationship is same as that of the 'decomposition' relationship in existing Feature definition used in [6]. This relationship defines a composite FORBID feature (Whole) with logically grouped (AND/OPT/XOR/OR) simple FORBID features (parts) that can provide for specific or complete selection among them. This relationship requires the parent Feature to be Composite in granularity

- ***Generalization / Specialization (Variability Composition) Relationships*** **-**With this relationship, a FORBID feature is a generalization of specialized multiple FORBID features (variants). This relationship requires the parent Feature to have category as 'abstract'.

The FORBID feature relationships are defined as follows:



- **Definition 2** : Let 'R' be the set of FORBID feature relationships Where
  - ✓ $R \subseteq F \times F$ and $F \neq \{\phi\}$ is the set of FORBID Features.
  - ✓ $R = R_{COMP}$ Where
- $\boldsymbol{R_{COMP}}$ - **FORBID Feature composition relationships** and $R_{COMP} = R_{AG} \cup R_{GS}$ where
  - ➢ $\boldsymbol{R_{AG}}$**= Aggregation relationship (Has A** relationship) between a Non-terminal Parent FORBID Feature (P) = with Granularity = Composite and Category Concrete and its children features (Ch_i).
    - ✓ $P \xrightarrow[Has\ A]{} Ch_i , i = 1 \ldots n$, Where ,
      - ○ $P = f_{cc}$
      - ○ $Ch_i = f_{sc_i} \vee f_{cc_i} , i = 1 \ldots n$
    - ✓ Ain Has A is defined to be the multiplicity of children to be selected whenever the parent is selected. It has two components: Minimum (Min) and Maximum (Max).
    - ✓ *{Min, Max}* returns the cardinality for a given grouped features.
  - ➢ $R_{AG} = R_{xor} \cup R_{and} \cup R_{or} \cup R_{opt}$ where
    - ✓ $\boldsymbol{R_{and}}$ = Aggregation relationship between a Non-terminal Parent FORBID feature (P) with Granularity = Composite and Category Concrete and its **set ofmandatorily related** Children features (Ch_i).
      $\left(P \xleftrightarrow[Has\ A]{} \wedge_{i \in n} Ch_i\right)$; A. Min = n ; A. Max = n
    - ✓ $\boldsymbol{R_{opt}}$ = Aggregation relationship between a Non-terminal Parent FORBID feature (P) with Granularity = Composite and Category Concrete and its **set ofoptionally related** Children features (Ch_i).
      $\left(P \xleftrightarrow[Has\ A]{} \vee_{1 \le i \le n} Ch_i\right)$; A. Min = 0 ; A. Max =n
    - ✓ $\boldsymbol{R_{xor}}$= Aggregation relationship between a Non-terminal Parent FORBID feature (P) with Granularity = Composite and Category Concrete and its **set ofmutually exclusivegroup** of Children features (Ch_i).
      $\left(P \xleftrightarrow[Has\ A]{} \vee_{1 \le i \le n} Ch_i\right)$; A. Min = 1 ; A. Max =1
  - ✓ $\boldsymbol{R_{or}}$ = Aggregation relationship between a Non-terminal Parent FORBID feature (P) with Granularity = Composite and Category Concrete and its **set of selective group of** Children features (Ch_i).
    $\left(P \xleftrightarrow[Has\ A]{} \vee_{1 \le i \le n} Ch_i\right)$; A. Min = 1 ; A. Max =n
    Selective group = a Group in which atleast one child should be selected.
  - ➢ $\boldsymbol{R_{GS}}$ **= Generalization / Specialization relationship (Is A** relationship) is defined between a Non-terminal Parent FORBID Feature (P) with Granularity = Simple / Composite and Category = Abstract and its children features (Ch_i). By virtue of its definition, at any instant one specialization of the composite FORBID Feature will be realized.
    - ✓ Specialization : $\left(P \xrightarrow[Is\ A]{} \vee_{i \in n} Ch_i\right) \wedge (P \xleftarrow[Is\ A]{} Ch_i)$
      - ○ First part states that P can be specialized to any one of Ch_i and
      - ○ The second part states that the selected Ch_i is now related to P
    - ✓ Generalization : $\left(P \xleftarrow[Is\ A]{} \wedge_{i \in n} Ch_i\right) \wedge (P \xleftarrow[Is\ A]{} \wedge_{i \in n} Ch_i)$
      - ○ First part states that P is a generalization of all Ch_i and
      - ○ They are related to P

AND, OPT, XOR and OR FORBID feature groups [EG_{and}, FG_{opt}, FG_{xor} and FG_{or}] are obtained with the help of R_{and}, R_{opt}, R_{xor} and R_{or} relationships.

(C) FORBID Feature Constraints

The constraints are applied to FORBID features present in FORBID model. It provides a method for verifying the stakeholders' customized preferences. Various FORBID Feature constraints including the highlighted new constraint is given in Figure 8:

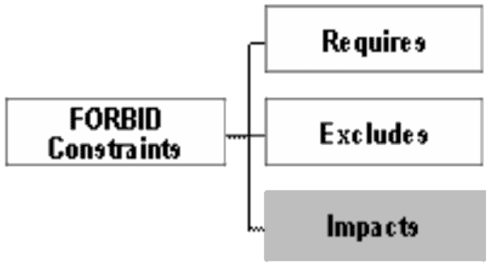


Figure 8: FORBID-feature Constraints

Constraints are defined as follows:

Definition 3: Let 'C' be a set of FORBID feature constraints and $C \subseteq F \times F$

- $f \in F$ and $f = NTF \cup TF$; Where
  - $F \neq \{\phi\}$ is the set of FORBID Features in FORBID Model
  - NTF = Non terminal features
  - TF = Terminal features
- $(f_1, f_2) \in C \rightarrow (f_1, f_2 \in TF) \land ((P, f_1) \land (P, f_2) \notin R_{COMP}))$
- $C = C_r \cup C_e \cup C_i$; where
- $C_r =$ Requires constraint, which is defined between two terminal features (f_1, f_2) and states that whenever the FORBID feature $f_1$is selected, then f_2 must also be selected.
  - $C_r = (f_1, f_2) \rightarrow (f_1 \xrightarrow[requires]{} f_2)$
- C_e = Excludes constraint, which is defined between two terminal features (f_1, f_2) and states that whenever the FORBID feature $f_1$is selected, then f_2 must not be selected.
  - $C_e = (f_1, f_2) \rightarrow (f_1 \xrightarrow[excludes]{} f_2)$
- C_i = impacts constraint, which is defined between two terminal features (f_1, f_2) and states that the selection of FORBID feature f_1 has an impact on f_2 and f_1 notifiesf_2.
  - $C_i = (f_1, f_2) \rightarrow (f_1 \xrightarrow[impacts]{} f_2) \land \left(f_1 \xrightarrow[notifies]{} f_2\right)$

(D) FORBID Evolution Operators

The Evolution operators that could model the FORBID feature level evolutions as well the FORBID model level evolutions are illustrated in Figure 9. The FORBID feature level operators are based upon the evolution operators given in [15], [16],[18]and the FORBID model level operators are based on the evolution operations described in [6].FORBID Feature level evolution operators helps the FORBID Features to evolve in various ways- *Add, Delete*, *Merge*, *Split* and *Change*. Similarly, Feature model level operations help the Feature model to evolve by '*adding*' and '*deleting*' the features in the Feature model. Both kinds of evolutions need to be addressed to handle the RBACP evolutionary changes completely. Its definition is as follows:

Definition 4: An evolutionary change in RBAC Permission **EVOPER** $_{IS}$ defined as

$$\mathbf{EVO_{PER}} = \mathbf{EVO_{PERFL}} \cup \mathbf{EVO_{PERML}}$$, where

- $\mathbf{EVO_{PERFL}}$ **=Universal set of** RBAC Permissions **evolutions** to be effected at FORBID feature level ; $\mathbf{EVO_{PERFL}} = \{ \mathbf{ev_{OPERFL}} \}$

- $\mathbf{EVO_{PERML}}$ **= Universal set of** RBAC Permissions **evolutions** to be effected at FORBID model level ;$\mathbf{EVO_{PERML}} = \{ \mathbf{ev_{OPERML}} \}$

Definition 4a : An evolutionary change to the RBAC Permission, $\mathbf{ev_{OPERFL}}$is defined as :
$\mathbf{ev_{OPERFL}}$ **= {Name of the change, arguments, Pre-conditions, Post-conditions} Where**

1. Name of the change : Name of the evolutionary changes.
2. Arguments:$(F \cup R \cup C)$, relevant to the FORBID Feature. A change could have one or more arguments.
3. Pre-conditions: It is a set of assertions relevant to FORBID Feature that must be satisfied in order to apply the change to any primitive elements of **FORBID Feature**.
4. Post-conditions: It is a set of assertions relevant to FORBID Feature that must be satisfied after implementing the change to any primitive elements of **FORBID Feature**.

Definition 4b : An evolutionary change to the RBAC Permission, $\mathbf{ev_{OPERML}}$is defined as :
$\mathbf{ev_{OPERML}}$ **= {Name of the change, arguments, Pre-conditions, Post-conditions} Where**

1. Name of the change : Name of the evolutionary changes
2. Arguments:$(F \cup R \cup C)$,relevant to the Feature Model. A change could have one or more arguments.
3. Pre-conditions: It is a set of assertions relevant to FORBID Feature that must be satisfied in order to apply the change to any primitive elements of **FORBID Model**.
4. Post-conditions: It is a set of assertions relevant to FORBID Feature that must be satisfied after implementing the change to any primitive elements of **FORBID Model**.

The different types of evolution operators are illustrated in figure 9.

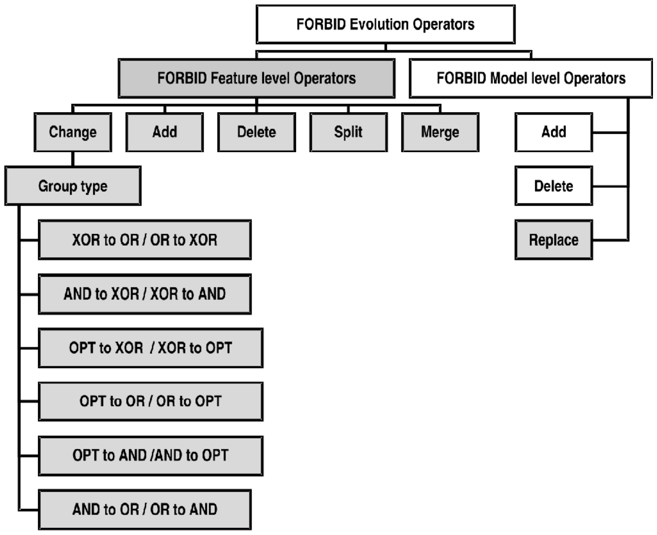


Figure 9: FORBID-feature Evolution Operators

Thus, by ameliorating the existing feature definition's attributes, relationships and constraints it is possible by the FORBID feature definition to characterize and model all types of RBACP permissions unambiguously and completely.

IV. EVALUATION ANALYSIS

Comparing with the existing Feature Definition, FORBID Definition is supposed to model all different types of RBAC permissions and also could handle the Evolutions. The following table shows the significance achieved by FORBID Feature

Table 2
Number of features used to model rbacp permissions

S.No	*Parameters*	*Existing Feature definition*	*FORBID Definition[Section 2]*
1	Total number of identified RBAC permissions	1 (Concrete)	*6[Simple/Composite/Concrete/ Abstract/Static/Dynamic]*
2	Number of Composition relationships	1[Aggregation]	2[Aggregation/Specialization]
3	Number of Feature Groups	2 [AND/OR]	4[AND/OR/XOR/OPT]
4	Number of Constraints	2 [Requires/Excludes]	3 [Requires/Excludes /Impacts]
5	Number of Feature level Evolution operators	0	5 [Add/delete/chage/merge/split]
6	Number of Feature model level evolution operators	2	2

From the above table, it is evident that FORBID feature can model RBAC Features completely and unambiguously when compared with the existing Feature Definition. Moreover, the proposed FORBID Feature is enabled to handle the evolutions at Feature and Feature model level, while existing definition cannot handle the Feature level evolutions.

V. EXAMPLE CASE STUDY

A small theoretical example from banking domain is used for this explanation i.e. **Access control requirements of "Accounting operations" related software functionality belonging to Banking domain**. The example is defined as follows: *Customers apply for account in the Bank to the clerk and these requisitions are stored automatically in Customer registration offline DB. Then the Customer personal profiles are created by the clerks and maintained by them with proper approval from manager at Customer personal profile offline DB. It can be modified by the clerks in offline mode. Customer profiles can also be created and modified by the clerks in online mode at Customer personal profile online DB, but this requires for the prior registration of the customers in the bank. The registration process is carried out by the clerk and he uses the Customer registration profile online DB. Both creation and modification of customer personal profiles involve multiple operations in them. Furthermore, all the modifications in the customer profile will come into effect only after approval from the manager. Customers can raise request for either debit or credit transactions at an instant in their account. These requests are carried out as financial transactions are by the cashier and are recorded in Customer Account DB. Any suspicious credit and over draft debits are subjected to prior approval of the manager.*

The FORBID features are defined from the RBAC permission requirements that have to be identified from the case study. Towards this, the various permissions (verb/action) are identified and the roles / resource sets associated with them are also identified. Table 3 depicts the same.

Table 3
RBACP permissions
Accounting-Operations

Roles	*RBACP permissions*	*Resource-Sets*
Customer	**Apply** account opening offline	Customer registration offline DB
Clerk	**Create** customer personal profile Offline	Customer personal profile offline DB
Clerk	**Create** customer personal profile Online	Customer personal profile online DB
Clerk	**Modify** customer personal profile Offline	Customer personal profile offline DB
Clerk	**Modify** customer personal profile Online	Customer personal profile online DB
Clerk	**Maintain** the customer personal profile	-
Clerk	**Register** for creating customer profile online	Customer registration online DB
Customers	**Request** for Debit or Credit	Customer Account DB
Cashier	**Debit** from Customer account	Customer Account DB
Cashier	**Credit** to Customer account	Customer Account DB
Manager	**Approve** Customer Profile	Customer personal profile offline DB Customer personal profile offline DB
Manager	**Approve** Customer Transact- Suspicious credit	Customer Account DB
Manager	**Approve** Customer Transact –Overdraft debit	Customer Account DB

DB – Database

These permissions (Features) are characterized and classified appropriately as simple / composite permissions (depending on the number of access control operations) and concrete / abstract permissions (depending on the details of roles, operations and objects) as could be captured from the case study. Then the interrelationships and constraints between these various FORBID features are defined.

(A) Interpretation of RBAC permissions

Applying the FORBID feature definition on the RBAC permissions captured from the case study, the following types of FORBID Features could be identified.

1. Composite Concrete Permissions

✓ *Apply* (RBAC permission) is performed by the customer (Role). At present, the customer applies for account opening by filling an application form and submitting the same to the bank personal. This operation involves two sub-operations viz. filling and submitting. The request for the account opening is stored in Customer registration offline DB (resource-set). *Apply* can be classified as follows:

   a. Composite – as *Apply* consists of “filling” and “Submitting” of account opening application.

   b. Concrete – as the *Apply* permission has specific description of its function. Also, the permission definition includes the Customer (role) and the Customer registration offline DB (resource set) on which the permission is provided.

   Hence *Apply* is classified **Composite Concrete**.

✓ The above said explanation holds the same for the following RBAC permissions listed in table 3 and they all are classified as **Composite concrete [Multiple access control operations and resource-sets]**. These permissions are tabulated in table 4 as follows:

Table 4
Composite Concrete RBACP permissions

RBACP permissions	*Access control operations in the RBACP permissions*
Create customer personal profile Offline	**1. Verify** the submitted profile
	2. Prepare the offline profile
	3. Submit to Manager
Create customer personal profile Online	**1. Verify** the submitted profile
	2. Prepare the online profile
	3. Submit to Manager
Modify customer personal profile Offline	**1. Add** new data
	2. Delete existing data
	3. Change existing data
Modify customer personal profile Online	**1. Add** new data
	2. Delete existing data
	3. Change existing data
Approve Customer Profile	**1. Verify** submitted profile data
	2. Permit for Account opening
Approve Customer Transact- Suspicious credit	**1. Verify** Customer transaction history
	2. Permit suspicious credit
Approve Customer Transact –Overdraft debit	**1. Verify** Customer transaction history
	2. Permit Overdraft debit

2. *Simple Concrete RBAC permissions*

✓ *Register* (The RBAC permission) is performed by Clerk (role). With this permission, the clerk can register a customer's request for having access to their account online. The request for registration is stored in Customer registration DB (resource-set). Register can be classified as follows:

a. Simple - as *Register* involves only one operation – registering customer's request for online account.

→ Concrete – as *Register* has specific description of its function. Also, the permission is assigned to Customer (role) and performed on a Customer registration online DB (resource set).

b. Similar to *Register,* certain other permissions of the case study can also be classified as **simple concrete.** These permissions are associated with appropriate roles and the resource sets. The table 5 details the same.

Table 5
Simple Concrete RBAC permissions

Role	*RBACP permission*	*Access control operation*	*Resource-set*
Customers	**Request** for Debit or Credit	Request	Customer Account DB
Cashier	**Debit** from Customer account	Debit	Customer Account DB
Cashier	**Credit** to Customer account	Credit	Customer Account DB

3. *Simple Abstract RBAC Permission*

• The FORBID feature *Maintain* is performed by Clerk (role) to maintain the integrity of the customer profiles. The customer profiles are stored in two resource sets viz. customer personal profile online DB and customer personal profile offline DB. *Maintain* can be classified as follows:

→ Simple - as *Maintain* involves only one operation

→ Abstract - as the specification of function of Maintain permission is not provided with clear details of the operations involved i.e. whether the customer profile has to be maintained either by periodic update or only when there is a need, or on which resource set.

Hence, *Maintain* is defined as an abstract feature.

The summary of all the above identified RBAC permissions of the case study are summarized in table 6 along with appropriate classifications as explained in previous discussions.

Table 6
Classification of FORBID features

S.No	*FORBID features*	*Granularity*	*Abstraction*
1	**Apply** for account opening	Composite	Concrete
2	**Create** customer personal profile Offline	Composite	Concrete
3	**Create** customer personal profile Online	Composite	Concrete
4	**Modify** customer personal profile Offline	Composite	Concrete
5	**Modify** customer personal profile Online	Composite	Concrete
6	**Approve** Customer Profile	Composite	Concrete
7	**Approve** Suspicious credit	Composite	Concrete
8	**Approve** Overdraft debit	Composite	Concrete
9	**Register** to create customer profile online	Simple	Concrete
10	**Request** to do debit or credit	Simple	Concrete
11	**Debit** from Customer account	Simple	Concrete
12	**Credit** to Customer account	Simple	Concrete
13	**Maintain** the customer personal profile	Simple	Abstract

Thus, the FORBID feature definition has the provision to characterize and classify the RBAC permissions as appropriate FORBID features from the functional requirements, stakeholder preferences and any other domain input unambiguously and completely.

(B) Identification of FORBID feature relationships

With respect to the above classified FORBID features, the following are the relationships that could be identified:

- ***IS-A Relationship:*** The access control operations – *Create customer personal profile Offline and Create customer personal profile Online* are composite concrete FORBID features with analogous operations on different resource-sets. This makes them as a candidate for "Generalization" relationship i.e. *create customer personal profile Offline* and *Create customer personal profile Online* can be generalized as an **abstract** feature – **"Create"** and it is attributed with two specializations viz. *Create customer personal profile Offline* and *Create customer personal profile Online.* **Create**is characterized as **composite** since both the specializations are composite in granularity. Thus, the RBAC permission **"Create"** is classified to be a **composite abstract FORBID feature**.

  ✓ The above said explanation suits for the access control operations – *Modify customer personal profile Offline, Modify customer personal profile Online.* "Modify" is defined as composite abstract FORBID feature that has two variants – "offline mode" and "online mode" which are said to be composite concrete.

- ✓ In a similar fashion, the different types of "**Approve"** operations are also analysed and are classified as follows **– Composite abstract"Approve"** with various specializations viz. approve customer profile, Approve Suspicious credit and Approve overdraft debit.
- ✓ ***XOR Feature Group:*** The FORBID feature – *Debit from Customer account and Credit to Customer account* can only be used in a mutually exclusive manner on a customer account. Hence, these features can be grouped in XOR aggregate relationships.
- ✓ ***Has –A Relationship:*** The operations "Apply", "Maintain" "Register" and "Request" are the aggregate children of "Accounting operations" software functionality.

Table 7 provides the relationships between the identified FORBID features.

Table 7
FORBID feature relationships

FORBID features		*Relationship with immediate parent*
Parent	*Children*	
Create	**Create** customer personal profile Offline	Is-A
	Create customer personal profile Online	
Modify	**Modify** customer personal profile Offline	Is-A
	Modify customer personal profile Online	
Approve	**Approve** Customer Profile	Is-A
	Approve Suspicious credit	
	Approve Overdraft Debit	
Accounting operations(Root)	**Debit** from Customer account	XOR feature group
	Credit to Customer account	
Accounting operations(Root)	**Apply** for Account opening	Has-A
	Maintain the customer personal profile	
	Register to create customer profile online	
	Request for Debit or Credit	

(C) Identification of FORBID feature constraints

Several constraints are defined based on the case study on the above classified FORBID features. They are tabulated in table 8. In this table, on either side of the 'constraints', the children FORBID features which are related via the constraint are listed and the parent Feature are listed alongside their children Features.

Table 8
FORBID feature constraints

FORBID features		*Constraints*	*FORBID features*	
Parent	*Children*		*Children*	*Parent*
Create	Create customer personal profile Offline	***Requires***	Apply	Accounting operations (root)
		Requires	Approve customer profile	Approve
	Create customer personal profile Online	***Requires***	Register	Accounting operations (root)
		Requires	Approve customer profile	Approve
Modify	Modify customer personal profile Offline	***Requires***	Approve customer profile	Approve

	Modify customer personal profile Online	***Requires***	Approve customer profile	Approve
Approve	Approve Customer Profile	-	-	
	Approve Suspicious credit	***Impacts***	Debit / Credit	Accounting operations (root)
	Approve Overdraft Debit	***Impacts***	Debit / Credit	Accounting operations (root)
Accounting (Root)	Debit from Customer account	-	-	
	Credit to Customer account	-	-	
Accounting (Root)	Apply for Account opening	-	-	
	Maintain customer personal profile		-	-
	Register to create customer profile online		-	-
	Request for Debit or Credit	-	-	

Thus, the FORBID features, their relationships and the constraints are wholly defined for the given case study.

VI. QUANTITATIVE ANALYSIS WITH RESPECT TO THE CASE STUDY

The case study provides a clear insight of the FORBID feature definition. However it is vital to prove that FORBID Feature definition satisfies the research objectives. In order to accomplish this the existing Feature definition [6] is also applied on the given case study and compared with the FORBID Feature definition. Both of the definition are then evaluated quantitatively with respect to the following parameters:

1. **Number of RBAC permissions identified from the domain -** This is the total number of RBAC permissions identified through domain analysis.
2. **Number of features used to model the identified RBAC permissions -**This is the number of features used to abstract the identified RBAC permissions.
   - ➢ The existing Feature definition characterizes an access control permission as a whole feature. This feature is composed of role and resources set as its constituent features (Parts). This composition is governed by the aggregation relationship defined in [11]. As identified earlier in chapter 3, the Feature meta-model based approach does not provide Generalization / Specialization relationship (inheritance) and Feature groups.
   - ➢ With FORBID Feature definition, appropriate FORBID Features are grouped together with Aggregation, Generalization/Specialization, XOR and OR Feature group relations.

Table 9
Number of features used to model RBAC Permissions

Parameters	*Existing Feature definition*	*FORBID Definition [As per the case study]*
Total number of identified RBAC permissions	**13**	**13***[Table 6]*
Number of Features used to model RBAC permissions based on AND Aggregation (Whole-part) relationship with Root.	**13**	**4***[Table 7]*

contd. table 9

Parameters	*Existing Feature definition*	*FORBID Definition [As per the case study]*
Number of Features used to model RBAC permissions based on Generalization / Specialization relationship	**0** [Non availability of Generalization / Specialization Relationship]	**3 abstract Features that have 7 children in total** *[Table 7]* **1. Create with** 2 Specializations **2. Modify** with 2 Specializations **3. Approve** with 3 Specializations
Number of Features used to model RBACP permissions with XOR aggregation	**0** **[Non-availability of Feature groups]**	**2 Features in Debit / Credit Group** *[Table 7]*
Number of Features used to model RBACP permissions with OR aggregation	**0**	**0***[Table 7]*
Total number of Features required to model RBACP permissions. ***Sum of Features taken from 2nd, 3rd, 4th and 5th rows***	**13** **[13 +0+0+0]**	**9** **[4 + 3 + 2]** **[Highlighted above]**

3. Number of Features that can undergo changes

- This is the number of features that can undergo different kinds of changes with respect to the changes in RBAC permissions.
- In existing Feature definition, the mandatory features cannot undergo changes, thus leaving the optional features alone to undergo changes. However, the changes in optional features will not be able to represent any changes with respect to the level of granularity and abstraction in RBAC permissions.
- The FORID Features are flexible such that they can undergo any kind of changes that demand the changes in the feature definition with respect to the level of granularity and abstraction in RBAC permissions.

Based upon the above given three parameters, the results are tabulated in table 10.

Table 10
Quantitative evaluations

S.No	*Parameters*	*Existing Feature Definition*	*FORBID Feature*
1	Number of RBAC permissions identified (using domain analysis) **(as per table 6)**	**13**	**13**
2	Number of Access control features used to model the identified RBAC permissions **(as per table 9)**	**13**	**9**
3	Number of Features that can undergo changes	**0**	Minimum = **8** Maximum = *fn* (Possible changes defined for the feature & its children)

From the above table, it is evident that FORBID approach used lesser number of FORBID features to abstract the voluminous RBACP permissions available in the Case study. This is possible by the ameliorated FORBID feature definition, which includes new attributes, relationships and constraints.

IV. CONCLUSION

A RBAC permission can be defined based upon its level of granularity, level of abstraction and type of enforcement. It is important to interpret and model all types of RBAC permissions unambiguously. ‘Feature’ – a design level abstraction was found to be more suitable to model all types of RBAC permissions without any ambiguity subjected to certain ameliorations. Based upon the RBAC permission modelling capability requirements, the ameliorations that are required to render Feature amenable to model all RBAC permissions are identified and applied on the existing Feature definition [Feature attributes, relationships and constraints]. The ameliorated Feature is named as FORBID Feature. The newly introduced FORBID Feature attributes, relationships and constraints are elaborately discussed.

A case study based evaluation is used to enumerate that FORBID Feature can interpret and model all types of RBAC permissions unambiguously and completely. Further research includes defining a FORBID oriented process and a system to handle the evolutions systematically as the RBAC permissions evolve with respect to user’s preferences or market requirements.

REFERENCES

[1] S. Northcutt, R. Hammer and P. Leight, “Role Based Access Control to Achieve Defense in Depth,” 2006.

[2] D. Ferraiolo, D. Kuhn and R. Chandramouli, Role Based Access Control, Artech House, 2007.

[3] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn and R. Chandramouli, “Proposed NIST standard for role-based access control,” *ACM Transactions on Information and System Security,*, vol. 4, pp. 224-274, 2001.

[4] L. A. Zaid, “The Feature Assembly Approach for Modelling & Knowledge Management of Software Variability,” Brussel, 2013.

[5] S. Apel and C. Kastner, “An Overview of Feature-Oriented Software Development,” *Journal of Object Technology,* vol. 8, no. 5, pp. 49-84, 2009.

[6] L. Sun and G. Huang, “Modeling Access Control Requirements in Feature Model,” in *Software Engineering Conference, APSEC ’09,Asia-Pacific*, 2009.

[7] S. Kim, D. K. Kim, L. Lu, S. Kim and S. Park, “A Feature-Based Approach for Modeling Role-Based Access Control Systems,” *Journal of Systems and SoftwareVol. 84, No. 12,* pp. 2035-2052., 2011.

[8] D. K. Kim, L. Lu and S. Kim, “A verifiable modeling approach to configurable role-based access control,” in *In the proceedings of FASE 2010. LNCS, vol. 6013,*, 2010.

[9] M. Svahnberg, “Supporting Software Architecture Evolution-Architecture Selection and Variability,” Department of Software Engineering and Computer Science,Blekinge Institute of Technology, 2003.

[10] K. Berg, J. Müller, J. v. Zyl and J. Bishop, “The Use of Feature Modelling in Component Evolution,” University of Pretoria, 2004.

[11] H. Mei, W. Zhang and H. Zhao, “A metamodel for modeling system features and their refinement, constraint and interaction relationships,” *Software and System Modeling 5(2),* pp. 172-186, 2006.

[12] K. ShanthaKumari and T. Chithralekha, “A Comparative Analysis of Access Control Policy Modeling Approaches,” *International Journa lof Secure Software Engineering,* pp. 65-83, 2012.

[13] K. Kumari and T.Chithralekha, “Challenges in Modeling Evolving Access Control Policies using Feature Modeling,” *JOURNAL OF SOFTWARE,* vol. 9, no. 5, pp. 1089-1094, 2014.

[14] L. Abo Zaid, F. Kleinermann and O. De Troyer, “Feature Assembly Framework: Towards Scalable and Reusable Feature Models,” in *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems,* Namur, Belgium, 2011.

[15] P. Ebraert, A. Classen, P. Heymans and T. D’Hondt, “ Feature Diagrams for Change-Oriented Programming,” in *International Conference on Feature Interactions in Software and Communication Systems, ICFI 2009,* Lisbon, Portugal, 2009.

[16] A. Pleuss, G. Botterweck, D. Dhungana, A. Polzer and S. Kowalewsk, “Model-driven support for product line evolution on feature level,” *Journal of system and software,* vol. 85, no. 10, pp. 2261-2274, 2012.

[17] C. Salinesi, A. Etien and I. Zoukar, “A Systematic Approach to Express IS Evolution Requirements Using Gap Modelling and Similarity Modelling Techniques,” in *16th International Conference, CAiSE 2004,* , 2004.

[18] L. K. C. Passos, S. Apel, A. Wasowski, C. Kästner, J. Guo and C. Hunsen, “Feature-Oriented Software Evolution,” in *The Seventh International Workshop on Variability Modelling of Software-intensive Systems*, 2013.

[19] C. Rolland, C. Salinesi and A. Etien, “Eliciting Gaps in Requirements Change,” *Requirements Engineering Journa*, vol. 9, no. 1, pp. 1-15, 2004.

[20] H. Pospesel, Introduction to Logic: Propositional Logic, Revised Edition (3rd Edition), 1999.

[21] I. Ray, N. Li, R. B. France and D. K. Kim, “Using UML to visualize role-based access control constraints,” in *Proceedings of the Symposium on Access Control Models and Technologies (SACMAT)*, 2004.

[22] D. Kim, I. Ray, R. France and N. Li, “ Modeling Role-Based Access Control Using Parameterized UML Models,” in *FASE 2004. LNCS, vol. 2984*, 2004.

[23] J. Jürjens, “UMLsec: Extending UML for Secure Systems Development.,” in *UML ’02 Proceedings of the 5th International Conference on The Unified Modeling Language*, Germany, 2002.

[24] T. Lodderstedt, D. Basin and J. Dose, “ Secureuml: A uml-based modeling language for model-driven security,” in *Proceedings of the International Conference on the Unified Modeling Language, UML’2002*, 2002.

[25] T. Priebe, E. B. Fernandez, J. I. Mehlau and G. Pernul, “A PATTERN SYSTEM FOR ACCESS CONTROL,” in *Proceedings of Eighteenth Annual Conference on Data and Applications Security*, Catalonia, Spain, 2004.

[26] D.-K. Kim and P. Gokhale, “A Pattern-Based Technique for Developing UML Models of Access Control Systems,” in *COMPSAC ’06 Proceedings of the 30th Annual International Computer Software and Applications Conference - Volume 01*, 2006.

[27] E. B. Fernandez, G. Pernul and M. M. Larrondo-Petrie, “Patterns and Pattern Diagrams for Access Control.,” in *Trust, Privacy and Security in Digital Business.*, 2008.

[28] G. Georg, R. France and I. Ray, “An Aspect-Based Approach to Modeling Security Concerns,” in *Proceedings of the Workshop on Critical Systems Development with UML*, Dresden, Germany, 2002.

[29] R. France, I. Ray, G. Georg and S. Ghosh, “An Aspect-Oriented Approach to Design Modeling,” *IEE Proceedings - Software, Special Issue on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, Vol. 151, No. 4*, pp. 173-185, August 2004 .

[30] G. Georg, I. Ray and R. France, “Using aspects to design a secure system,” *In Proceedings of the International Conference on Engineering Complex Computing Systems (ICECCS 2002), Greenbelt, MD, ACM Press.*, 2002.

[31] G. Georg, I. Ray, K. Anastasakis, B. Bordbar, M. Toahchoodee and S. H. Houmb., “An aspect-oriented methodology for designing secure applications,” *Information and software technology*, pp. 846-864, 2009.

[32] E. Song, R. Reddy, R. France, I. Ray, G. Georg and R. Alexander, “Verifiable composition of access control features and applications,” in *Proceedings of 10th ACM Symposium on Access Control Models and Technologies (SACMAT 2005)*, 2005.

[33] S. Kim, D.-K. Kim, L. Lu and E. S. , “Building hybrid access control by configuring RBAC and MAC features,” *Information and Software Technology*, vol. 56, p. 763–792, 2014.