Risk Assessment for Relational Database Schema-based Constraint Using State Machine Diagram

Kanjana Eiamsaard[1], Nakornthip Prompoon[2]

Abstract— **Information is a critical asset of an organization. This is why reducing the chance of data loss must be a primary concern. This research focuses on the risk assessment issue in physical level of data implementation, which includes identifying risk factors and assigning a hazard level. This research will investigate how to predict the chance of incorrect data based on breaking risk assessment constraints using a state machine diagram. The diagram represents the status of the data when that data must change as a result of a program execution. The advantages of this research include helping software development teams identify what risks may occur in different scenarios, and helping reduce development cost by allowing them to minimize those risks.**

Index Terms— **Risk Assessment, Relational database, Schema-base constraint, State Machine Diagram**

I. Introduction

Ideally all applications usually use relational database databases that include correct constraints processing. But today some applications still generate inconsistent data in immature relational databases by breaking schema constraints. Changing the RDBMS is not always possible in practice for a variety of reasons, such as during software migration when only one application is changed but others still require the old RDBMS.

This is why researchers are trying to solve this problem by using better code patterns in applications [9]. The code pattern solution is useful, but once data is already corrupted and the program has been already written. It is too late to fix the problem with only a code pattern solution. These problems will usually show up when the application is almost done. Applications using MySQL's popular MYISAM storage engine are one example where the logic to enforce constraints must be put into the application code because the engine does not enforce constraints.

Ramez Elmasri classifies database constraints into three groups: Inherent model-based constraints, schema-based constraints, and semantic constraints. The research focuses on schema-based constraints which can be represented in the Data Definition Language (DDL) created in the software design phase. The schema-based constraints contain 4 constraints: domain constraints, key constraints, NULL constraints, and referential integrity constraints [3]. Applications created which obey all of these constraints will never allow inconsistent data to be stored in the database. However, in practical, there is always the case that these constraints may do not enforce.

The paper is structured as follows. Section 2 presents underlying concepts. Section 3 mentions some related works. Section 4 proposes our approaches. Section 5 describes the application of our proposed approach. Finally, the paper ends with conclusions and future works in section 6.

Manuscript received Jan 08, 2013.

Kanjana Eiamsaard is with the Software Engineering Laboratory, Center of Excellence in Software Engineering Department of Computer Engineering, Faculty of Engineering Chulalongkorn University, Thailand ; e-mail: Kanjana.E@student.chula.ac.th

Nakornthip Prompoon is with the Software Engineering Laboratory, Center of Excellence in Software Engineering Department of Computer Engineering, Faculty of Engineering Chulalongkorn University, Thailand ;e-mail: Nakornthip.S@chula.ac.th

II. Underlying concepts

A. Risk

Boehm defined the terms risk factor as the probability of loss or injury and its severity of undesired event [1]. Meanwhile NASA also defined Risk factor as the combination of the probability that a program or project will experience an undesired event such as safety mishap, compromise of security, or system component failure; and the consequence, impact, or severity of the undesired event were it to occur [2]. Risk exposure is defined by the relationship shown here:

$$RF = P(UO) * L(UO) \quad (1)$$

RF : Risk factor
$P(UO)$: A probability of loss or injury
$L(UO)$: Impact or severity of the undesired event

When an injury happens on a project, the resulting hazard level of each project may be different. NAZA classified the hazard level into four levels:

1) Catastrophic means the loss of an entire system.
2) Critical means the system sustained major damage.
3) Moderate means the system sustained minor damage.
4) Negligible means the system was under stress, but no system damage occurred.

B. Schema-based constraints

Constraints are one of many attributes that relational databases must be concerned. The types of constraints this paper will consider are [3]:

1) A domain constraint specifies the type of each attribute.
2) An entity constraint specifies whether the value of each attribute can be NULL or not.

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

3) A key constraint specifies that all elements of a set are distinct. This means no two rows have the same combination of values for these attributes.
4) A referential integrity constraint specifies a relationship between two tables and is used to maintain the consistency of data among rows in the two relations.

C. State machine diagram

A state machine diagram is used for representing the changing status of a system, sub-system or an object. Each object will respond to an event and change to another state depending on the event. There are many kinds of state machine diagrams but this paragraph will discuss only "Change event state machine diagram". This diagram is the satisfaction of a Boolean expression that depends on a designated attribute value. It is important because it focuses the model on the true dependency—an effect that occurs when a given condition is satisfied [8].

III. RELATED WORKS

A. Architectural-Level Risk Analysis Using UML

This research considers a dynamic risk factor by not only considering risk in a component but also considering risk in the connections between components. The algorithm evaluates the hazard level with a hazard analysis. Then it applies a Markov model to analyze the risk of various scenarios in each use case. Finally, it shows a critical component and/or connection which the developer must verify will work correctly [5].

B. Generating test data from state-based specifications

This research generates test data by using full predicate coverage criterion [7]. It also presents a prescriptive approach that uses an expression parse tree. An expression parse tree is a binary tree that has binary and unary operators for internal nodes and variables and constants at leaf nodes. The tree's node consists of AND (∧) OR (∨) and NOT (~). An output from this method is the "Truth table" which it will be used as test data. The method to generate the truth table is described below:

First, a test clause is chosen. Next, the parse tree is walked from the test clause up to the root, then from the root down to each clause. While walking up a tree, if a given clause's parent is OR, its sibling must have the value of False. If its parent is AND, its sibling must have the value of True. If a node is the inverse operator NOT, the parent node is given the inverse value of the child node. This is repeated for each node between the test clause and the root. Once the root is reached, values are propagated down the unmarked subtrees using a simple tree walk. If an AND node has the value True, then both children must have the value True; if an AND node has the value of False, then at least one child must have the value False (which one is arbitrary). If an OR node has the value of False, then both children must have the value False; if an OR node has the value of True, then at least one child must have the value True (which one is arbitrary). If a node is the inverse operator NOT, the child node is given the inverse value of the parent node. For example, the truth table for $(A \vee B) \wedge C$ is shown in table 1:

Table I. The truth table for $(A \vee B) \wedge C$ [7]

	$(A$	$\vee$	$B)$	$\wedge$	C
1	T		F		T
2	F		F		T
3	F		T		T
4	F		F		T
5	T		T		T
6	T		T		F

After test data is generated, it will be considered with triggering events inside a state machine diagram. This paper suggests implementing this by assuming two versions of the triggering event variable, X and X', where X represents the before-value of X and X' represents its after-value. Finally, expression rules are used with the triggering event as shown here:

- $@T(X) = \sim X \wedge X'$
- $@T(X \wedge Y) = \sim(X \wedge Y) \wedge (X' \wedge Y') = (\sim X \wedge \sim Y) \wedge (X' \wedge Y')$
- $@T(X \vee Y) = \sim(X \vee Y) \wedge (X' \vee Y') = (\sim X \wedge \sim Y) \wedge (X' \vee Y')$
- $@F(X) = X \wedge \sim X'$
- $@F(X \wedge Y) = (X \wedge Y) \wedge \sim(X' \wedge Y') = (X \wedge Y) \wedge (\sim X' \vee \sim Y')$
- $@F(X \vee Y) = (X \vee Y) \wedge \sim(X' \vee Y') = (X \vee Y) \wedge (\sim X' \wedge \sim Y')$

$@T$ or $@F$: An event which causing $X's$ content change.
$@T(X)$: An event which changes from false state to new state
$@F(X)$: An event which causing $X's$ content change.
X *and* X' : These two variables represent the value before and after event.

C. Test cases generation from a state chart diagram

This research proposed the approach to generating test case which related to the full predicate coverage criteria, consists of 2 steps [8].
1) Considering the condition of the state transition from UML specification.

A condition of the state transition is represented in table, which consists of "Parent state", "Current state", "Condition" and "Destination state". For example, given the considered state machine diagram in Fig. 1, a corresponding table of the state transition is shown in Table 2.

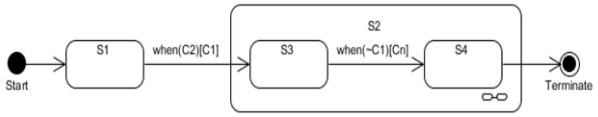

Fig 1. An example state machine diagram.

Table II. A condition of the state transition

Parent state	Current state	Event condition				Destination state
		C1	C2	…	Cn	
-	S1	t	***@T***	-	-	S3
S2	S3	***@F***	-	-	F	S4

After a table of the state transition is created, then the expressions are generated. From Table 2, two expressions are generated:

1) $@\boldsymbol{T}(\boldsymbol{C2}) \wedge \boldsymbol{C}1 = \sim C2 \wedge C2' \wedge C1$
2) $@\boldsymbol{T}(\boldsymbol{C1}) \wedge \boldsymbol{C}n = \sim C1 \wedge C1' \wedge Cn$

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Finally, a test specification is generated using a generated expression as shown in Table 3.

Table III. Test specification from a state machine diagram.

Predicate No.	Parent State	Current State	Event condition	Destination State
P1	-	S1	$\sim C2 \wedge C2' \wedge C1$	S3
P2	S2	S3	$\sim C1 \wedge C1' \wedge Cn$	S4

2) Considering each predicate to generate test case specification.

In this step the truth table is modified to be a test case specification by following an algorithm which was proposed by Offutt [7] as shown in Table 4.

Table IV. Test specification generated from a state machine diagram.

Predicate NO.	Parent State	Current State	C1	C2	…….	Cn	After value	Destination State
P1	-	S1	t	F			$C2'$=True	S3
		S1	t	T			$C2'$=True	S1
		S1	f	F			$C2'$=True	S1
		S1	t	F			$C2'$=False	S1

IV. Our Approach for Risk Assessment for Relational Database Schema-based Constraint Using State Machine Diagram

This proposed approach is to be used for evaluating risk in various scenarios which are related to changing data in a database in the case where there may be occurrences of breaking any kind of schema-based constraints. It considers risk factors in each scenario by not only considering them in a component but also considering them in the connections between components. In this paper, a "scenario" means an event in software which effects data stored in a database. A "component" means part of a scenario which represents a kind of schema-based constraint. The events may come from the insert, delete and update operation. After both scenarios and components are created, their risk factors are computed by using a risk factor model. Finally, hazard analysis is adapted to identify severity levels for each component that will be described later. All of the steps can be represented with an activity diagram as shown in Fig. 2

This paper also presents an algorithm that shows how to calculate the risk factor as shown in Fig. 3

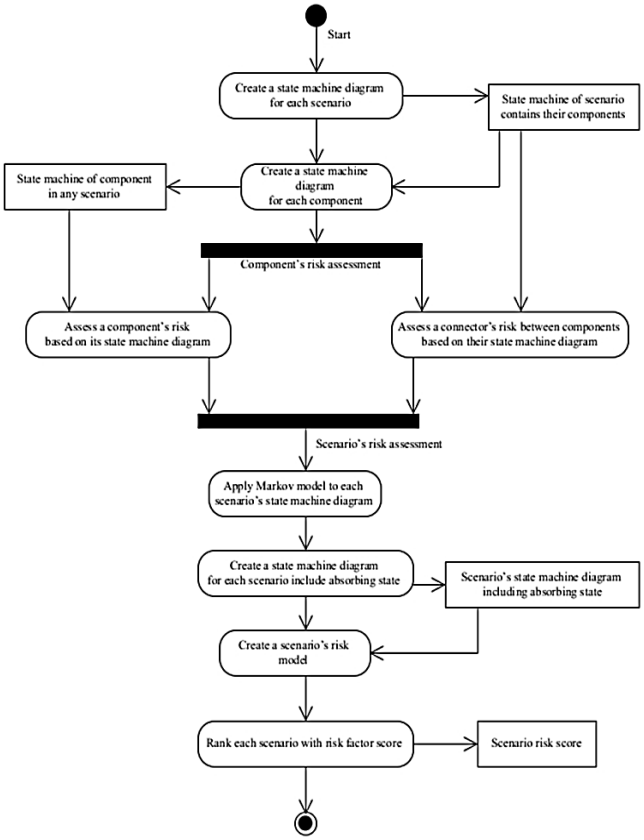

Fig 2. Activity diagram of the proposed risk assessment approach

Algorithm: Risk assessment for relational database schema-based constraint using behavioral state machine diagram.
Input: State machine diagram in XML format and Entity relationship diagram in DDL format.
Output: Rank of scenario's risk factors and the summary of all scenario risk factors.
Pre-condition: Syntax of state machine diagram is correct according to the UML2.0 specification.

1. For each scenario
1.1 For each component
1.1.1 Identify risk factor
1.1.2 Define the hazard score by using the number of attributes and relations from entity relationship diagram (ER)
1.1.3 Calculate risk factor
1.2 For each connector
1.2.1 Identify risk factor
1.2.2 Define the hazard score by using the number of relations which relate to connectors from entity relationship diagram (ER)
1.2.3 Calculate risk factor
1.3 Create state machine diagram of the normal execution software and state machine diagram of software with absorbing state
1.4 Apply Markov model to generate state machine diagram from step 1.3
1.5 Summarize risk factors
2. Rank the scenarios in order by risk factor score

Post-condition: The risk factor score from the algorithm will be captured in a form that can be used in the risk management process.

Fig 3. Risk assessment algorithm

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

The activity diagram in Fig. 2 is explained step by step as follows:

1. Create a state machine diagram for each component of each scenario. This step will generate a state machine diagram of components in scenarios effects the data in a relational database.

2. Risk Assessment consists of two steps:

2.1. Component risk assessment is calculated by using a model as shown here:

$$rf_i^x = PUOCS_i \cdot svt_i \quad (2)$$

rf_i^x : Risk factor of component i in scenario x

$PUOCS_i$: A probability of loss or injury of component i

svt_i : Impact or severity of the undesired event of component i

There are six steps for calculating the probability and severity for the model:

2.1.1. Calculate the probability of a satisfactory component (PSC) transition from state p to q by using full predicate coverage criterion which will generate a number of test case as shown here:

$$PSC_{pq} = \frac{|TrueTestCases_{pq}|}{|TotalTestCases_{pq}|} \quad (3)$$

$|TrueTestCases_{pq}|$: The number of test cases which make a correct transition from state p to q.

$|TotalTestCases_{pq}|$: The total number of test cases generated.

2.1.2. Calculate the probability of a satisfactory path through the state machine diagram using this formula:

$$PSC_{initial-final} = \prod_{i=1}^{n-1} PSC_{P_iP_{i+1}} \quad (4)$$

$PSC_{initial-final}$: A probability of a satisfactory path through the state machine diagram from the initial node to the destination node.

$PSC_{P_iP_{i+1}}$: A probability of a satisfactory transition between states. The result of this formula is the probability of state i multiplied by the probability of state (i+1).

2.1.3. Calculate the probability of an unsatisfactory path through the state machine diagram using this formula:

$$PUC_{initial-final} = 1 - PSC_{initial-final} \quad (5)$$

2.1.4. Calculate the probability of an unsatisfactory outcome (PUOC) using this formula:

$$PUOC_{\text{i}} = \sum_{k=1}^{m} \left[\left[\frac{PP_k}{\sum_{l=1}^{m} PP_l} \right] \cdot \left(PUC_{initial-final} \right)_k \right] \quad (6)$$

$PUOC_{\text{i}}$: A probability of unsatisfactory outcome of component i in each scenario.

m : The number of possible paths in a state machine diagram.

k: The number of the path in state machine diagram.

PP_k: A product of the number of test cases in the k[th] path.

$\sum_{l=1}^{m} PP_l$: A summary of the product of number of test cases in each possible path.

$\left(PUC_{initial-final}\right)_k$: The probability that the k[th] path will be unsatisfactory.

2.1.5. Normalize $PUOC_{\text{i}}$ to make it appropriate for the scenario x (S_{x}) by using this formula:

$$PUOCS_{\text{i}} = \frac{PPC_i}{\sum_{j=1}^{n} PPC_j} \cdot PUOC_i \quad (7)$$

PPC_i: The total number of test cases generated for component i.

$\sum_{j=1}^{n} PPC_j$: The total number of test cases generated for every component in the same scenario.

2.1.6. Define the hazard score by using an entity relationship diagram. This score is calculated as shown in Fig. 4.

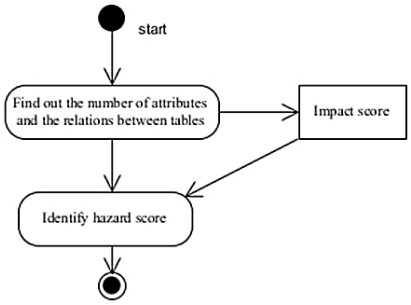

Fig 4. Activity diagram for identifying hazard score.

2.2. Connector's risk assessment.

Using the formula from Goseva-Popstojanova [5], each connector risk factor is calculated using the following formula.

$$EOC_{ij}^x = \frac{|MT_{ij}^x|_{i.j \in S_x, i \neq j}}{|MT^x|} \quad (8)$$

2.3. Scenario's risk assessment

This step creates a scenario model based on state-based modeling. After the state machine diagram of each scenario is created, then the Markov property is applied on the state machine diagram in order to identify the probability of the transition one state to another.

2.3.1. Scenario's risk assessment.

This model is represented by a control flow graph that contains a single start node and terminating node. The other nodes are used for representing an executable's software components. Those components are related to database schema-based constraints. Then a probability matrix is created by applying Markov property, so it means a probability of transition from component i to component j. This matrix is named as P^x and its formula is shown here:

$$p_{ij}^x = \frac{n_{ij}^x}{\sum_j n_{ij}^x} \quad (9)$$

2.3.2. Create a software's scenario risk model

Each transition of a scenario risk model is used to compute a probability in the transition probability matrix $\overline{P_{ij}}^x$. This matrix represents the case where a component i does not fail, and then the control is transferred to the component j, and finally the interaction between i and j does not fail. A formula for creating the transition probability matrix is shown here:

$$\overline{P_{ij}}^x = (1 - rf_i^x) \cdot p_{ij}^x \cdot (1 - rf_{ij}^x) \quad (10)$$

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

After a software execution state machine diagram is created, it will be combined with abnormal execution states which imply hazard levels when software execution fails. A state machine contains (n+1) transient nodes where n is the number of components and a starting node, and (m+1) absorbing nodes where m is the number of failure nodes and a terminating node. Then a probability matrix is created by applying the Markov property on the state machine diagram. This matrix is named as $\overline{P^x}$ and shown here:

$$\overline{P^x} = \begin{bmatrix} Q^x & C^x \\ 0 & I \end{bmatrix} \quad (11)$$

$\overline{P^x}$: The transition probability of all components in the scenario risk model.

Q^x : The (n+1) by (n+1) matrix of probability of transition from component to component.

C^x: The (m+1) by (m+1) matrix of probability of transition from a component to failure node.

0: The (m+1) by (n+1) zero matrix.

I: The (m+1) by (m+1) identity matrix.

Each variable of the above formula is provided by the previous formulas. Therefore, it is possible to compute all probabilities of the transition matrix. Next, matrix A^x is defined to represent a probability that starting from a transient node will transition to an absorbing node. The formula used for generating matrix A^x is shown here:

$$A^x = (I - Q^x)^{-1} C^x \quad (12)$$

Finally, the total probability of transition from a transient node to each kind of failure node is computed, and it is used to compute a risk factor score for the considering scenario.

2.4. Rank the scenario order by risk factor score

This last step of risk assessment occurs when various scenario's risk factors in the software are calculated. A scenario risk factor score is ordered, so it will be easy to use in the risk management process.

V. The Application of the Proposed Approach

This section describes the application of the proposed risk assessment approach using an employee management system as our case study. The system operation consists of inserts, updates and deletes of employee records. Therefore, an insert of an employee record which considers schema based constraints is used for demonstrating this approach. The entity relationship of an employee management system is shown in Fig. 5.

Apart from an ER diagram, which is used in the generation of a state machine diagram, a class diagram generated from data manipulation aspect as shown in Fig. 6 is also an essential diagram which must be considered.

When an object is created from "EmpEntity" class in Fig. 6, this object's attributes and methods are related to the changing data in a database. Therefore, a state machine diagram of an object is created as shown in Fig. 7 – 10.

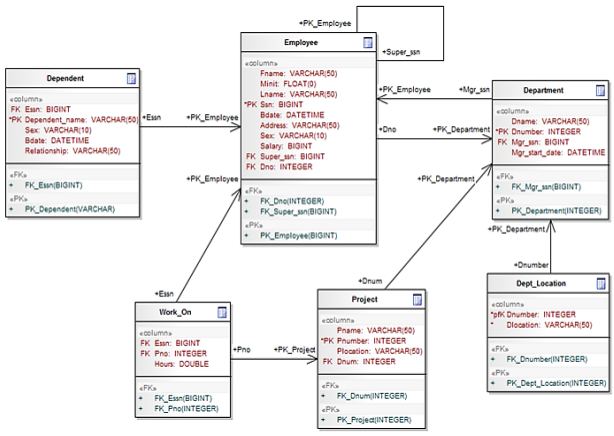

Fig 5. Employee management system's entity relationship diagram

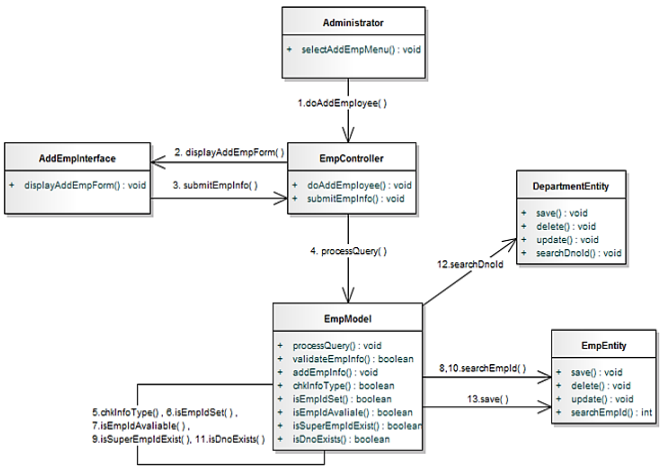

Fig 6. Class diagram of data manipulation layer

1) A component is created for checking NULL value of an employee's data which is related to an "Entity constraint" or "NULL constraint". This component is used to check if a NULL value was used. A state machine diagram of this component is shown in Fig. 7

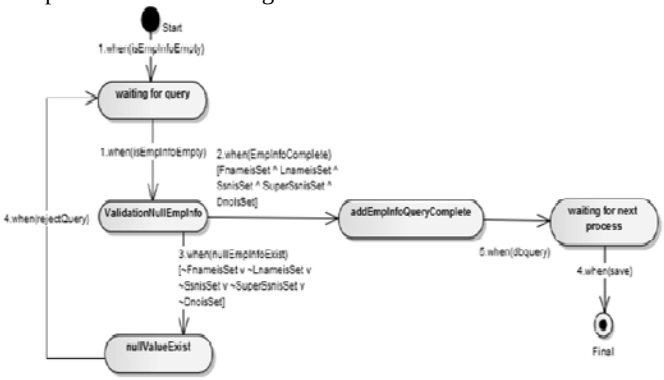

Fig 7. A state machine diagram of the checking for a NULL value in the employee record.

2) A component is created for checking domain of an employee's data which is related to a domain constraint. This means before inserting any data into the database, the variable contents must have the correct data type. A state machine diagram of this component is shown in Fig. 8.

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

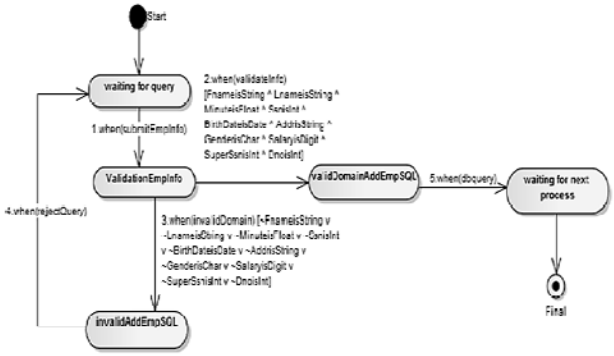

Fig 8. A state machine diagram for checking a domain constraint of an employee's data.

3) A component is created for checking an employee code which is related to "Key constraint". In this case, an employee code (SSN) is a primary key, so the application must ensure SSN is a unique value before inserting an employee's data into the database. A state machine diagram of this component is shown Fig. 9.

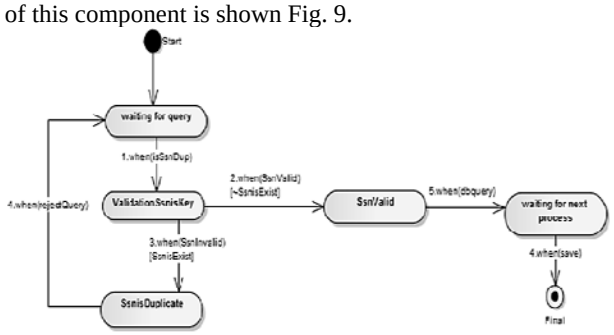

Fig 9. A state machine diagram for checking a key constraint in an employee's data.

4) A component is created for checking a leader employee code and department code which is related to a "Referential integrity constraint". This means that the application must ensure the exist of the Super_Ssn in the Employee table and Dno in the Department table before inserting an employee's data into the database. A state machine diagram of this component is shown in Fig 9.

Fig 10. A state machine diagram for checking a referential integrity constraint in an employee's data.

After applying our proposed approach, the result of risk assessment is presented in Table 5 – 8.

Table V. The probability of the satisfied outcomes after enforcing the NULL constraint.

No.	Current State	Destination State	PSC_{pq}
1	Waiting for query	ValidationNullEmpInfo	0.333333333
2	ValidationNullEmpInfo	addEmpInfoQueryComplete	0.777777778
3	ValidationNullEmpInfo	nullValueExist	0.333333333
4	nullValueExist	Waiting for query	0.333333333
5	addEmpInfoQueryComplete	Waiting for next process	0.333333333

Table VI. The probability of the satisfied outcomes after enforcing the domain constraint.

No.	Current State	Destination State	PSC_{pq}
1	Waiting for query	ValidationEmpInfo	0.333333333
2	ValidationEmpInfo	validDomainAddEmpSQL	0.0714286
3	ValidationEmpInfo	invalidAddEmpSQL	0.8461538
4	invalidAddEmpSQL	Waiting for query	0.333333333
5	validDomainAddEmpSQL	Waiting for next process	0.333333333

Table VII. The probability of the satisfied outcomes after enforcing the key constraint.

No.	Current State	Destination State	PSC_{pq}
1	Waiting for query	ValidationSsnisKey	0.333333333
2	ValidationSsnisKey	SsnValid	0.25
3	ValidationSsnisKey	SsnisDuplicate	0.25
4	SsnisDuplicate	Waiting for query	0.333333333
5	SsnValid	waiting for next process	0.333333333

Table VIII. The probability of the satisfied outcomes after enforcing the referential integrity constraint.

No.	Current State	Destination State	PSC_{pq}
1	Waiting for query	ValidationRefkey	0.333333333
2	ValidationRefkey	RefkeyValid	0.2
3	ValidationRefkey	Reference missing	0.6
4	Reference missing	Waiting for query	0.333333333
5	RefkeyValid	waiting for next process	0.333333333

Then, a connector's risk assessment is computed and the result of the EOC_{ij}^{x} is shown in Table 9.

The next step is to generate scenario risk factors composed of two steps.

1. Generate software behavioral model and calculate a probability of transition between nodes. An example of the insertion operation of an employee's data using state machine is shown in Fig. 11 and risk factor for connections between components is shown in Table 10.
2. Generate software's scenario risk model and calculate a probability that starting from a transient node to an absorbing node (A^x). The result is shown in Fig. 12.

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

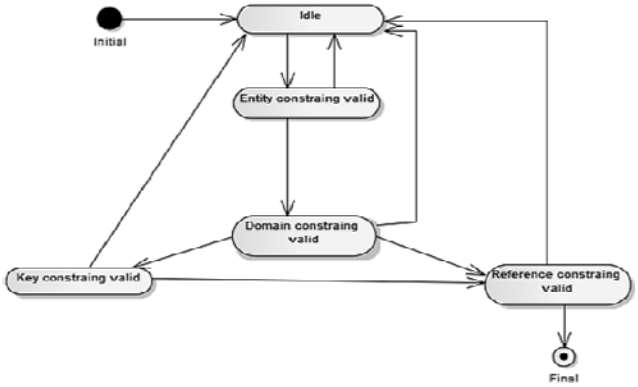

Fig 11. A software behavioral model for inserting employee's data

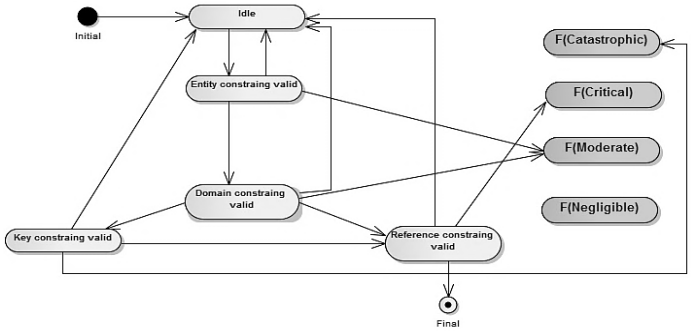

Fig 12. A software's scenario risk model for inserting employee's data

Table IX. Risk factor for the connection between components.

Sender \ Receiver	S	Idle	Entity valid	Domain valid	Key valid	Ref valid	T
S	0	0.090909091	0	0	0	0	0
Idle	0	0	0.090909091	0	0	0	0
Entity valid	0	0.090909091	0	0.090909091	0	0	0
Domain valid	0	0.090909091	0	0	0.090909091	0.090909091	0
Key valid	0	0.090909091	0	0	0	0.090909091	0
Ref valid	0	0.090909091	0	0	0	0	0
T	0	0	0	0	0	0	0

Table X. A probability of transition between two transient nodes of inserting employee's data scenario.

Sender \ Receiver	S	Idle	Entity valid	Domain valid	Key valid	Ref valid	T
S	0	1	0	0	0	0	0
Idle	0	0	1	0	0	0	0
Entity valid	0	0.5	0	0.5	0	0	0
Domain valid	0	0.33	0	0	0.33	0.33	0
Key valid	0	0.5	0	0	0	0.5	0
Ref valid	0	0.5	0	0	0	0	0.5
T	0	0	0	0	0	0	1

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Finally, A^x will be generated using formula (10) and those scores will be used for ranking various scenarios in the software.

VI. CONCLUSION AND FUTURE WORK

The proposed risk assessment approach is to calculate the probability that an application will be breaking the schema-based constraints in a relational database using state machine diagram. Ideally this approach will be used for evaluating a scenario's risk factor which covers the evaluation of component and connection's risk factor. For evaluating component and connection risk factors we apply a full predicate coverage to generate an unsatisfactory probability. In addition, the number of attributes and relations of the database's schema are used for identification of the hazard level. Finally, a scenario's risk factor is calculated using the component and connection's risk factor. The advantage of this approach is an acquisition of scenario risk factors which will be used in the risk management process.

To make this approach possible to use, the next step is developing a tool that implements the proposed approach. The result of this tool would be the ordered list of the scenario's risk factors which may cause inaccuracy of data in the relational database. This tool requires input of state machine diagrams in the XML format and the database's schema in DDL format.

REFERENCES

[1] B.W. Boehm, "Software risk management: principle and practices", *IEEE Software*, vol. 08, no.1, pp. 32-41, Jan 1991.

[2] NASA Technical Std. NASA-STD-8719.13A, Software Safety, March 2004,pp. 26-27.

[3] Ramez Elmasri, Shamkant B. Navathe, "Fundamentals of database systems sixth edition", pp 67-74.

[4] James Rumbaugh, Ivar Jacobson, Grady Booch "The Unified Modeling Language Reference Manual Second Edition" USA, Addison-Wesley, 2005, pp 79-89.

[5] Katerina Goseva-Popstojanova, Ahmed Hassan, Ajith Guedem "Architectural-Level Risk Analysis Using UML", *IEEE transactions on software*, vol. 29, no. 10, Oct 2003

[6] Akekachai Tangsuksant and Nakornthip Prompoon, "Risk Assessment Framework based on Goal-oriented Requirements Engineering and Object Behavioral Model"

[7] Jeff Offutt, Shaoying Liu, Aynur Abdurazik and Paul Ammann *Generating test data from state-based specifications*, Springer-Verlag Berlin Heidelberg, February 2003

[8] Karunee Bowornprasirtkul and Nakornthip Prompoon, "Test cases generation from a state chart diagram", 2004

[9] Hongyu Zhang, Hee Beng Kuan Tanc, Lu Zhangd, Xi Lina,b, Xiaoyin Wangd, Chun Zhangd and Hong Meid (2011,December). "Checking enforcement of integrity constraints in database applications based on code patterns", *Journal* [Systems and Software], vol. 84(12), pp 2253-2264

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)