An Automated Impact Analysis Approach for Test Cases based on Changes of Use Case based Requirement Specifications

Adisak Intana[1], Kanjana Laosen[2], Thiwatip Sriraksa[3]
College of Computing, Prince of Songkla University, Phuket, Thailand

***Abstract*—Change Impact Analysis (CIA) is essential to the software development process that identifies the potential effects of changes during the development process. The changing of requirements always impacts on the software testing because some parts of the existing test cases may not be used to test the software. This affects new test cases to be entirely generated from the changed version of software requirements specification that causes a considerable amount of time and effort to generate new test cases to re-test the modified system. Therefore, this paper proposes a novel automatic impact analysis approach of test cases based on changes of use case based requirement specification. This approach enables a framework and CIA algorithm where the impact of test cases is analysed when the requirement specification is changed. To detect the change, two versions as before-change and after-change of the use case model are compared. Consequently, the patterns representing the cause of variable changes are classified and analysed. This results in the existing test cases to be analysed whether they are completely reused, partly updated as well as additionally generated. The new test cases are generated automatically by using the Combination of Equivalence and Classification Tree Method (CCTM). This benefits the level of testing coverage with a minimised number of test cases to be enabled and the redundant test cases to be eliminated. The automation of this approach is demonstrated with the developed prototype tool. The validation and evaluation result with two real case studies from Hospital Information System (HIS) together with perspective views of practical specialists confirms the contribution of this tool that we seek.**

Keywords*—Change impact analysis approach; test case; black-box testing; use case based requirement specification; combination of equivalence and classification tree method

I. Introduction

Software testing is one of the most necessary and integral stages in the software development life cycle [1]. It consists of a set of activities that are performed by using a systematic approach. This enables a System Under Test (SUT) to be executed on a set of test cases to ensure that the software system is free from error. Black-box testing is one of the software testing methods that examines the functionality of software without knowing the internal structures or mechanisms. It is sometimes called specification-based testing as this type of software testing is being performed to validate the complete or integrated software based on the Software Requirements Specification (SRS) document. Test cases are usually derived from software artifacts such as specifications and design. Software testers initially go through the requirement document to understand requirements and specifications before designing and generating test cases.

Recently, black-box testing techniques have been widely applied in many practical software development domains including Point of Sales (POSs) [2], Internet of Things (IoTs) [3], School Payment Information System [4] and e-Learning system [5]. The result of this application can confirm that black-box testing can guarantee the correctness of the system functionality whether it meets the expected users' needs. Furthermore, in the software testing community, several black-box testing techniques have been introduced. The widely known black-box testing techniques include Equivalence Class Partition (ECP) [1], Boundary Value Analysis (BVA) [1], Classification Tree Method (CTM) [6], [7], Decision Table–Based Testing [1], State Transition Testing [1]. These influence the benefit which maximises test case coverage with a minimised number of test cases. However, one of the research challenges in software testing is that since the generated test case is normally derived from the SRS document, changing requirements always impacts on the existing previous test cases which are unable to be retested in the modified system [8], [9]. As the advanced technology demands businesses grow and evolve their needs, the requirement of systems becomes more complex and continuously changes. Over time, some domain features may not be needed, therefore either removing or replacing them with a new feature, whilst may be refined [8], [9]. Whenever the software requirement is changed, the testing process is affected. This leads to new test cases being created from the changed version of software requirements specification. Consequently, this causes considerable time and resource effort to generate test cases to test all new systems to ensure that all components in the system are connected and functioning correctly. Thus, Change Impact Analysis (CIA) for testing is needed to determine how much software testing and test automation should be performed by software testers after the change.

Therefore, in our previous work [10], we proposed the conceptual vision of impact analysis framework of test cases based on changes of use case based requirements. This framework analyses the impact on changes of test cases when variables in use case specification are changed. Five patterns of atomic changes were designed specifically according to the change of use case specification affecting the test case as 1) change of variable name, 2) change of variable type, 3) change of variable range value, 4) change of number of variables, and 5) change of variable order. The result of this impact analysis enables the existing test cases to be completely reused, partly updated as well as additionally generated. The modified version of test cases is generated with the Combination of Equivalence

and Classification Tree Method (CCTM), the hybrid testing generation approach of ECP with CTM testing technique. The combination of testing technique increases the level of testing coverage and reduces the redundant test cases. The effectiveness of this framework was demonstrated manually with the real case study, *Kidney Failure Diagnosis (KFD) system*.

The work proposed in this paper is extended from our previous work [10]. This paper presents an automated impact analysis approach of test cases based on changes of use case based requirement specification. The automation of this approach is demonstrated by an automatic prototype tool in which the CIA algorithm of the approach is implemented in this tool. To detect changes, the tool provides consistency checking by comparing two versions as before-change and after-change of use case XML files. Five patterns of atomic changes designed specifically are encoded into this tool to analyse and classify the cause of the change. The tool also performs the analysis of level of change impact on the before-change version of test cases that are classified into four groups as *no-change*, *delete*, *update* and *create* categories. To create the new test cases, CCTM was implemented in the tool. Furthermore, we show the applicability and efficiency of the proposed automatic approach and tool with two case studies formulated from real-world Hospital Information System (HIS) with perspective views of practical specialists. The result of this test case generation from these case studies by the tool was compared with the expected test cases that were calculated manually to validate the accuracy and effectiveness of both approach and implementation. The reusability level of test cases was also evaluated by the case studies. Moreover, the satisfactory level of the proposed approach and tool was evaluated from practical specialists for potential application in the future.

The rest of the paper is structured as follows. Section II gives the necessary background and related work. Then, Section III describes and demonstrates the proposed impact analysis approach and the detailed algorithm with our experienced case study. In Section IV, we demonstrate the proof of concept of our proposed approach through the evaluation of implemented prototype tool for impact analysis of test cases with two real-world case studies. Section V presents lesson learned and discussions of this study. Finally, conclusions and areas of future work are summarised in Section VI.

II. Related Work

There are some research studies dealing with the application of the CIA approach to analyse the effect level of test case changes [11], [12], [9], [8], [13]. Some of these focus on the impact analysis of test cases from changes in database schema such as the work proposed by [12], [13]. The author in [12] presented a tool for analysing the changes of database schema encoding in terms of embedded SQL in the source code. The database schema change is detected from the recorded log file before the impact of the corresponding embedded SQL in the source code is analysed and new test cases are finally generated. In [13], the impact of changes in database schema requirements such as addition, deletion, modification or change of some schema values is analysed before a set of test cases corresponding to the change is generated.

Some research studies proposed a CIA framework for test cases from the change of specification [11], [9], [8], [14]. The author in [11], [9] proposed the impact analysis framework of test cases from the change of use case specification. The author in [11] proposed a test case selection framework based on the change of two versions of use case description. These two versions of the use case description are encoded in XML file format and used as input for this framework. Requirement validation matrix mapping the use case with its corresponding generated test cases is used to check and track the change of use cases. When there are changes in the use case, the existing test case affected from the change is considered whether is can be reusable. The author in [9] extended the capability of [11] in which the requirement validation matrix is generated automatically from the old version of use case description when it does not exist. However, both former research studies only analyse the impact results in either the existing test case from the previous version can be reused or the new test case must be generated. They do not classify the cause of change into a pattern which may help testers to analyse the change of test cases faster and more efficiently. Furthermore, their framework does not support the automatic test case generation in the case that new test cases are required to be generated due to the added feature or requirement. Recently, [14] proposed a CIA approach for test cases affected from the change of inputs or outputs of functional requirements. The approach offers the change process enabling the change control over functional requirements, test cases and database schema. The rollback mechanism is also provided to support in the case of cancelling of changes. Similar to other earlier mentioned studies, the cause of change is not classified and the level of reusability is not specified in this work.

The closely related work is proposed by [8]. They proposed the impact analysis framework of test cases from the change of use case specification. They introduced seven patterns indicating the cause of changes into their impact analysis tool. This includes the change of 1) variable name, 2) data type, 3) variable value, 4) variable tag, 5) order, 6) link and 7) variable number. They are used to analyse the two versions of HTML document and XML schema file representing the previous and changed version of input and output specified in the Graphic User Interface (GUI) of web application. ECP and BVA testing techniques are used to generate the new test case. From our literature reviews, we can conclude that there are few attempts at the specification level. Therefore, our challenge is to contribute an impact analysis approach for test case based on changes of requirement specification.

III. Materials and Approach

A. Framework of Impact Analysis of Test Cases

A conceptual overview of an impact analysis framework of test cases based on changes of use case proposed in our preliminary work [10] as shown in Fig. 1. This framework is divided into four steps: 1) variables in two versions as before-change and after-change of the use case specification model are extracted. Then, 2) the extracted variables from these two versions are compared. This results in the variables affected from the requirement change to be identified. The cause of changes is also identified. There are five patterns of atomic changes classified for the cause of changes that are a) change of

variable name, b) change of variable type, c) change of variable range value, d) change of number of variables and e) changes of variable order. After that 3) the impact analysis of test cases is performed. This enables the corresponding test case from the before-change version recorded on the database to be retrieved in order to analyse the level of change impact. Consequently, the analysed test cases are classified into four groups that are *no-change*, *delete*, *update* and *create* categories. In the case of no change effect, the test case is totally reused in the new version. However, if change impacts of test cases are detected, either the affected test case is updated giving it improvement or 4) the new test case is fully generated according to the additional requirement by using CCTM technique.

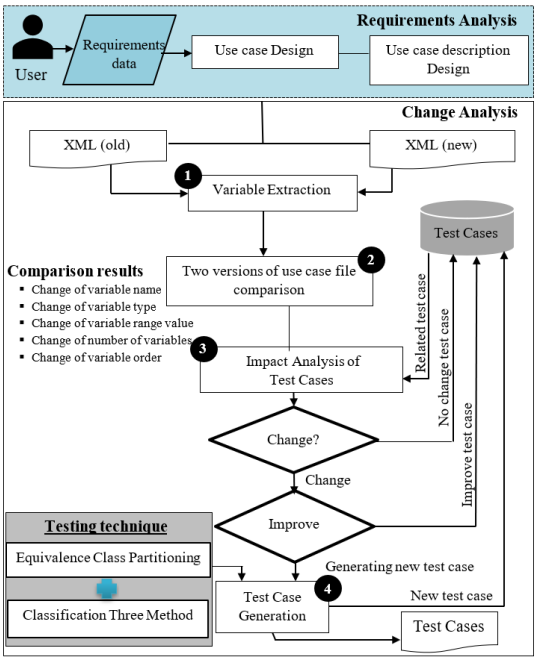


Fig. 1. Framework of impact analysis of test cases [10]

B. Case Study

To demonstrate the effectiveness of our approach, we formulate our case study, Medical Record System (MRS) from a real-world system, Hospital Information System (HIS)[1], an open-source software based on Java platform. The system was deployed over 90 hospitals in Thailand. This system has been chosen as it is a safety-critical system related to human diagnosis that requires a high degree of correctness. Furthermore, we select two real case studies from the subsystem that has the requirements changing issues, these types of modules require a high degree of accuracy calculation and measurement. To explore the effectiveness of our approach, we chose Kidney Failure Diagnosis (KFD) and Online Nursing Assessment (ONA) subsystems from HIS. Table I shows the criteria of the selected case studies from the different level of complexity. Number of implemented use cases is the first criteria to demonstrate the complexity of system. As the Cartesian product method is used in testing technique in which all variable range value are multiplied together to generate test cases, number of variable is the second criteria. Number of change requests influencing five patterns of atomic changes is the other criteria used for case study selection. As can be seen in this table, KFD is a simple case study that is used to demonstrate and evaluate how the proposed approach and its implemented prototype tool work. ONA is a more complex case study compared to KFD in terms of number of use cases, variables and change requests. Moreover, it contains the change requests covering all patterns of atomic changes. Therefore, ONA is more focused on the evaluation of the efficiency of the approach and implementation. Specifications of these systems are described in the following sections.

[1]http://www.opensource-technology.com

TABLE I. CASE STUDIES CHARACTERISTICS

System	# use cases	# variables	# change requests	# change of				
				variable name	*variable type*	*variable range value*	*number of variables*	*variable order*
KFD	3	6	3	×	×	-	×	×
ONA	5	19	4	×	×	×	×	×

1) Kidney Failure Diagnosis (KFD) Subsystem: KFD is a subsystem in MRS that records the history of patient treatment. KFD subsystem provides the result of kidney failure analysis to physicians helping them to diagnose whether their patients kidney disorders are abnormal. The Chronic Kidney Disease (CKD) of a patient is calculated and used to interpret the stage of kidney disease by this system. This calculation is based on two factors: *Glomerular Filtration Rate (GFR)* and *Uric Acid Creatinine (UO)*. *GFR* is a calculation that is used to determine how well the kidneys function by estimating the blood that is filtered by the kidneys. GFR is normally calculated by using mathematical formula including the *Age*, *Gender* and *Serum Creatinine (SCr)* of patients. Failure of the patient kidney is interpreted by determining *GFR* values together with the effects of *UO*. This results in the stage of kidney failure to be interpreted and suggests what treatment is applied to the patient. After this diagnosis and applying the treatment to the patient, the nurse records the information regarding the measured factors, interpretation and diagnosis results into the system, so that the physician can view the history of treatment in order to monitor the patient disease later.

Requirements. The requirements of this system are summarised as follows:

- ***REQ1:*** The system shall automatically calculate *GFR* value for the patient from *Gender*, *Age* and *Scr*.
- ***REQ2:*** The system shall accurately perform the kidney failure state from *GFR* and *UO* value mapping in five stages.
  - *Stage 1 (Risk Stage):* GFR >=90 AND UO > 300 mg/g
  - *Stage 2 (Injury):* GFR = 60-89 AND UO > 300 mg/g
  - *Stage 3 (Failure):* GFR = 30−59 AND (UO = 30−300 mg/g OR UO > 300 mg/g)
  - *Stage 4 (Loss):* GFR = 15−29 AND (UO <

30 mg/g OR UO = 30-300 mg/g OR UO > 300 mg/g)
- *Stage 5 (ESRD):* GFR < 15 AND (UO < 30 mg/g OR UO = 30-300 mg/g OR UO > 300 mg/g)

- ***REQ3:*** The interpreter of kidney failure states per a patient must be displayed at only one stage.

Use Case based Specification Model. The use case diagram of KFD subsystem is shown in Fig. 2. It consists of three modules as demonstrated as use cases *UC001-UC003* that covers all KFD functionality including GFR calculation, interpretation and display that correspond to requirements *REQ1-REQ3*, respectively. The detailed functionality of each use case is represented in the form of use case description as shown in Fig. 3. This figure demonstrates a brief description of use case *UC001* that describes the behaviour of function GFR calculation. It explains the sequence of the step for use case *UC001* including *main flow of events*, *alternative of events* and *exception flow* of each use case. The flow of events demonstrates the main step-by-step of GFR calculation. Furthermore, the input for GFR calculation including *Gender*, *Age* and *SCr* used to calculate *GFR* is also indicated in the corresponding step. After the GFR calculation, the GFR stage is interpreted as an output of this functionality. Based on the input and output, the relevant variables affected the change of test cases can be analysed and stored in the XML dictionary document for generating test data.

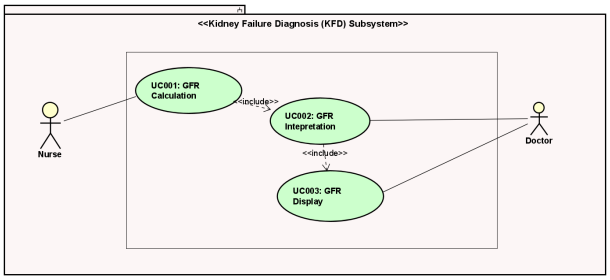


Fig. 2. Use case model for KFD subsystem

Use Case#	UC001: GFR Calculation	
Pre-conditions:	System shows the menu to record GFR	
Primary Actor:	Nurse	
Flow of Events:	1	The system shows the GUI to input data.
	2	The nurse inputs medical record (*Gender*, *Age* and *SCr*).
	3	The nurse saves the medical record.
	4	The system calculates GFR [E1] [E2] [E3].
	5	The system interprets GFR stage via the use case *UC002, GFR Interpretation*.
	6	The system displays GFR stage via the use case *UC003, GFR Display*.
Alternative of Events:		…
Exception Flows:	1	[E1] If the nurse inputs the variables (*Gender*, *Age* and *SCr*) that are not in the correct range, the system will display "Error GFR calculation".
	2	[E2] If the system maps wrong input of variables, the system will display "Wrong Interpretation".
	3	[E3] If the system maps input that is not in the correct range, the system will display "GFR stage does not match".
Version	01	

Fig. 3. Example of use case description of UC001 (Version 1)

Considering each use case diagram of the KFD subsystem, variables from the data dictionary of the system are introduced and used as input data, as shown in Table II. This table contains *variable names*, *variable types* and *data range* values. The type of variable and the range value of data for each variable are determined for the purpose of creating the test data such as valid variables of *Gender* are *F* and *M*, variable *Age* contains the correct data range values: *0-120*, which is in the range of data interest.

TABLE II. DATA DICTIONARY OF UC001 (VERSION 1)

Variable Name	Description	Data Type	Data Range
Gender	Gender of patients	Varchar(1)	F, M
Age	Age of patients	Integer	0-120
SCr	Serum Creatinine	Float	0.0-10.0
UO	Urine Albumin	Integer	0-∞
GFR	Glomerular Filtration Rate	Integer	0-∞
Stage	Kidney Failure Stage	Varchar(255)	Stage 1-Stage 5

Requirement Change Requests. There were the changes of the detailed requirement in the first version of use case based specification model from the user that were recorded in the requirement change request form as shown in Table III. The user requested to change the formula for GFR calculation as identified in ***CH01***. This change request affected the functionality of requirement ***REQ1***. In the first version, only a single formula was used to calculate GFR for all ages. In the changed version, five formulas are used for different ages instead as demonstrated in Table IV. The GFR calculation is measured separately for an adult patient whose age is greater than and equal to *18* with different *Sex* and *SCr* and a child patient who has age less than *18*. Moreover, the factor *Height* is also included in the changed formula for the child patient. There was also change request ***CH02*** that results in the change in requirement ***REQ2*** in which variable *Gender* was renamed to be *Sex*. In the last change request ***CH03***, the data type of GFR were also changed from integer to float. The relevant variables in the changed version are analysed as shown in Table V.

TABLE III. THE CHANGE REQUEST OF KFD SUBSYSTEM

Change ID	Change Description	Affected Requirement ID	Affected Use Case ID
CH01	Change the formula for GFR Calculation	REQ1	UC001
CH02	The name of variable change from gender to sex based on the table of GFR formula	REQ2	UC003
CH03	Change data type of *GFR* to decimal	REQ3	UC002

TABLE IV. THE CHANGED GFR FORMULA

Age	Sex	SCr	Formula
≥18	F	≤ 0.7	GFR=144×$(SCr/0.7)^{-0.329}$×$(0.993)^{Age}$
		> 0.7	GFR=144×$(SCr/0.7)^{-1.209}$×$(0.993))^{Age}$
	M	≤ 0.9	GFR=141×$(SCr/0.9)^{-0.411}$×$(0.993)^{Age}$
		> 0.9	GFR=141×$(SCr/0.9)^{-1.209}$×$(0.993)^{Age}$
<18	F/M	GFR=0.413 × $Height((cm)/SCr(mg/dL)$	

Note: F=$Female$, M=$Male$

Fig. 4 and Table V describe the use case description and its corresponding details of data dictionary in the changed version respectively. Based on the requirement change, there is a change in variable *Age* in which the data range value of this variable is changed from *0-120* to two range values that are *0-17* (*<18*) and *18-120* (*≥ 18*). As the value of variable *SCr* is calculated based on the value of variables *Sex* and *Age*

as shown as the different equations in Table IV, This affects the value of this variable to be changed. The range value of factor *SCr* is divided into two main criteria for female and male which is ($\leq$ 0.7, $>$ *0.7*) and ($\leq$*0.9*, $>$ *0.9*) respectively. Furthermore, as the height of patients is included into the changed formula, variable *Height* with its range value *(0-300)* is added into the new version of data dictionary. Finally, the type of variables *GFR* is changed from integer to floating point as to increase more accurate and precise result interpretation.

Use Case#	UC001: GFR Calculation	
Pre-conditions:	System shows the menu to record GFR	
Primary Actor:	Nurse	
Flow of Events:	1	The system shows the GUI to input data.
	2	The nurse inputs medical record (*Gender*, *Age*, *Height* and *SCr*).
	3	The nurse saves the medical record.
	4	The system calculates GFR by using the formula depends on *Gender* and *Age* [E1] [E2] [E3].
	5	The system interprets GFR stage via the use case *UC002, GFR Interpretation*.
	6	The system displays GFR stage via the use case *UC003, GFR Display*.
Alternative of Events:		…
Exception Flows:	1	[E1] If the nurse inputs the variables (*Gender*, *Age*, *Height* and *SCr*) that are not in the correct range, the system will display "Error GFR calculation".
	2	[E2] If the system maps wrong input of variables, the system will display "Wrong Interpretation".
	3	[E3] If the system maps input that is not in the correct range, the system will display "GFR stage does not match".
Version	02	

Fig. 4. Use case description of UC001 (Version 2)

TABLE V. DATA DICTIONARY OF UC001 (VERSION 2)

Variable Name	Description	Data Type	Data Range
Sex	*Gender of patients*	*Varchar(1)*	*F, M*
Age	*Age of patients*	*Integer*	*$<$ 18, $\geq$ 18*
SCr	*Serum Creatinine*	*Float*	*$\leq$ 0.7, $>$ 0.7, $\leq$ 0.9, $>$ 0.9*
Height	*Height of patients*	*Integer*	*0-300*
UO	Urine Albumin	Integer	0-∞
GFR	*Glomerular Filtration Rate*	*Float*	*0.0-∞*
Stage	Kidney Failure Stage	Varchar(255)	Stage 1-Stage 5

2) Online nursing assessment (ONA) subsystem: ONA subsystem is used to record the clinical information of the patient in order to be used to evaluate the symptoms of the patient and consider the treatment for the doctor including medicine dispensing to patients and accurate treatment.

Requirement. The following are the requirements of this subsystem:

- ***REQ1:*** Nurses shall record nursing assessment data such as patient severity (*Triage*), service type (*VisitType*), service status (*VisitStatus*), *allergy*, *pregnancy*, and *lactation* status.
- ***REQ2:*** Nurses shall record the clinical information of the patient including vital signs (V/S), which includes pulse rate (*pulse*), respiratory rate (*RR*), body temperature (*Temp*) and blood pressure (*BP*), as well as weight (*weight*), height (*height*) and waistline (*waistline*) for patients. The normal range value of each clinical information is shown as follows.
  - *Pulse:* 60-100 time/min.
  - *PR:* 12-18 time/min.
  - *Temp:* 36.5-37.5 °C
  - *BP:* 90/60-120/80 mmHg
- ***REQ3:*** The system shall automatically calculate Body Mass Index (*BMI*) for patients. The formula is calculated as *BMI = Weight divided by square meter of height*.
- ***REQ4:*** The level of nutrition is divided into five levels as follows [15]:
  - *BMI $<$ 18.5:* Underweight
  - *BMI $\leq$ 18.5 and BMI $\geq$ 24.9:* Normal weight
  - *BMI $\leq$ 25.0 and BMI $\geq$ 29.9:* Pre-obesity
  - *BMI $\leq$ 30.0 and BMI $\geq$ 34.9:* Obesity class I
  - *BMI $\leq$ 35.0 and BMI $\geq$ 39.9:* Obesity class II
  - *BMI $\geq$ 40.0:* Obesity class III

Use Case based Specification Model. The use case diagram corresponding to the requirements described in the previous section is shown in Fig. 5. There are two actors specified in this use case, the nurse and doctor. Two main use cases are identified, *UC01: Assessment Record* and *UC02: Vital Sign Record*. Use case *UC01* implements requirement ***REQ1*** regarding nursing assessment record, whereas requirement ***REQ2*** regarding operation recording the clinical information of the patient is implemented by use case *UC02*. After recording such clinical information, the calculation and interpretation operation is performed based on requirements ***REQ3*** and ***REQ4***. Therefore, three use cases that are *UC021: BMI Calculation*, *UC022: Nutrition Interpretation* and *UC023: Nutrition Display* are included by use case *UC02*.

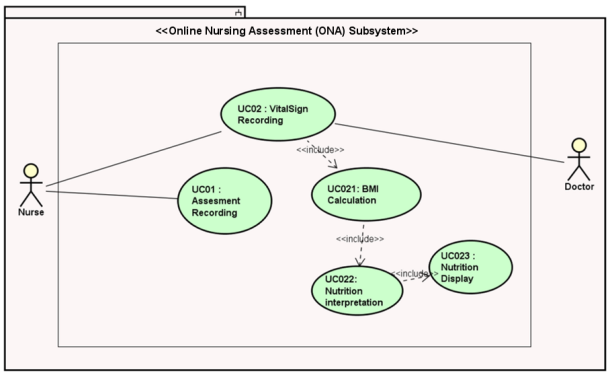


Fig. 5. Use case model for ONA subsystem

Fig. 6 shows the use case description of use case *UC01* (Version 1). This figure specifies the details of use case including the sequence of each step of use case *UC01* including *Main Flow of Events*, *Alternative of Events* and *Exception Flows*. As can be seen in this figure, the description of use case *UC01* is explained regarding the functionality of nursing assessment record. Considering the flow of events in this table, variables *Triage*, *VisitType*, *VisitStatus*, *Allergy*, *Pregnancy*, *Lactation* and *PainScore* are indicated as an input data. These variables and their data structure are defined in the data dictionary as the form of XML file format. This file is used later to generate test data. These variables including their data structure and possible data range are shown in Table VI. There are 19 variables included in the first version of data dictionary used by the functionality of ONA subsystem. For example, variable *VisitType* that has data type as a string for recording the reason of visit. This variable contains nominal data value to be *Self*, *Relative* and *Other*. Another example is variable *Triage* that

determines the priority of patients' treatments based on the severity of their condition. This variable has ordinal data range value of a character from *1* to *5*.

Use Case#	UC01: Assessment Recording	
Pre-conditions:	Select the patient from the system	
Primary Actor:	Nurse	
Flow of Events:	1	The system displays the assessment recording window.
	2	The nurse inputs assessment information including severity (*Triage*), type of service (*VisitType*), service situation (*VisitStatus*), *Allergy*, *Pregnancy*, *Lactation* and *PainScore* of patients.
	3	The nurse clicks "Save" button to save assessment information [A1].
Alternative of Events:	1	[A1] The nurse clicks "Cancel" button in the case of cancellation [E1].
Exception Flows:	1	[E1] The system will clear screen.
Version	01	

Fig. 6. Use case description of UC01 (Version 1)

TABLE VI. DATA DICTIONARY OF UC01 (VERSION 1)

Variable Name	Description	Data Type	Data Range
Sex	Gender of patients	Varchar(1)	F, M
Age	Age of patients	Integer	1-120
Pulse	Pulse Rate	Integer	0-201
RR	Respiratory Rate	Integer	0-120
Temp	Body Temperature	Integer	35-46
BP_upper	Systolic blood pressure	Integer	0-350
BP_lower	Diastolic blood pressure	Integer	0-350
Weight	Weight of patients	Integer	0-200
Height	Height of patients	Integer	0-300
WL	Waist of patients	Integer	5-80″
BMI	Body Mass Index	Float	18.50-30.00
VisitType	Service Type	Varchar(255)	Self, Relative, Other
VisitStatus	Service Status	Varchar(255)	Walk, Wheelchair, Other
Triage	Triage Level	Varchar(1)	1-5
Allergy	Allergy Status	Varchar(1)	Y, N
Preg	Pregnancy Status	Varchar(1)	Y, N
Lactation	Lactation Status	Varchar(1)	Y, N
Pain Score	Pain Level	Integer	1-10
NT Level	Nutrition Level	Varchar(2)	L1-L5

Requirement change requests. After we analysed the requirement change request, we discovered that this change of requirements affected the functionality of requirement as shown in Table VII. There are two types of changes, deleting and adding some requirements respectively. This impacted the use case description and data dictionary to be changed as shown in Fig. 7 and Table VIII, respectively. According to change request ***CH01***, *Pregnancy* and *Lactation* of patient are excluded from inputs of assessment information in the second version of the use case description. This affected the variables *Preg* and *Lactation* specified in the first version of data dictionary were deleted. Change request ***CH02*** introduces the new functionality into the subsystem to display the nursing record information after assessment recording functionality, the new use case (*UC011*) is added and extended from *UC01*. This added use case results in variable *DateTime* to be added in order to record the patient information based on the selected date. This variable contains nominal data range values *CD* and *UD* for the current date and updated date respectively. Furthermore, some variables changed their data range value. For example, from change request ***CH03***, variable *Pulse* were separated from one partition (*0-201*) into two partitions as *60-100* and *90-130* for different age groups respectively. Change request ***CH04*** affected the name of variable *NT Level* to be renamed to *BMI Level* according to the international standard name based on WHO.

TABLE VII. THE CHANGE REQUEST OF ONA SYSTEM

Change ID	Change Description	Affected Requirement ID	Affected Use Case ID
CH01	Delete pregnancy and lactation details	REQ1	UC01
CH02	The system can display the nursing record information based on the selected date (current date: *CD* and updated date: *UD*)	REQ2	UC011
CH03	Adjust the patient's normal range including *Pause*: 90-130 times per minute for children aged 18 years old and 60-100 times per minute for adult aged over 18 years old	REQ3	UC02
CH04	Rename the nutrition level display to BMI level	REQ4	UC021
	Change data type of *Weight* and *Height* to decimal	REQ4	UC023

Use Case#	UC01: Assessment Recording	
Pre-conditions:	Select the patient from the system	
Primary Actor:	Nurse	
Flow of Events:	1	The system displays the assessment recording window.
	2	The nurse inputs assessment information including severity (*Triage*), type of service (*VisitType*), service situation (*VisitStatus*), *Allergy* and *PainScore* of patients.
	3	The nurse clicks "Save" button to save assessment information [A1].
Alternative of Events:	1	[A1] The nurse clicks "Cancel" button in the case of cancellation [E1].
Exception Flows:	1	[E1] The system will clear screen.
Version	02	

Fig. 7. Use case description of UC01 (Version 2)

C. CIA Algorithm

To achieve a better understanding of our impact analysis framework as described in Section III-A, the detailed algorithm is explained in this section. This is demonstrated with using our experienced case study, KFD subsystem as an example.

Step 1: Variable Extraction. After the change request is identified by the change analysis, variables specified in the use case diagram are extracted. As variables and their data structure are normally defined in the data dictionary, the XML of data dictionary that is extended from the XML of conformed use case is used as a source for this variable extraction in our developed prototype tool. The variable information in the XLM file including *use case id*, *variable name*, *variable type*, *variable range value*, *variable number* and *variable order* are extracted before it is analysed for the level of change impact.

Fig. 8 demonstrates an example of the changed version of the XML of data dictionary that conforms to the changed version of use case description. The information of variables *Sex*, *Age*, *Height* and *SCr* encoded in this XML file is extracted. Especially, the data structure that are *variable type* and *variable range value* from the extracted information is normally used as information to generate test cases or test data. For instance, considering variable *Age* which has data type as an integer, there are four possible test cases to be considered, the age in the range of *0-17*, *18-120* and out of the range that are less than *0* and greater than *120*.

Step 2: Two Versions of Use Case File Comparison. This step performs consistency checking in order to identify the variables affected from the change of requirement. Two versions of the extracted variable information between before-change and after-change of the used case diagram from the previous step are compared and analysed. This results in the

TABLE VIII. DATA DICTIONARY OF UC01 (VERSION 2)

Variable Name	Description	Data Type	Data Range
Sex	Gender of patients	Varchar(1)	F, M
Age	Age of patients	Integer	1-120
Pulse	*Pulse Rate*	*Integer*	*60-100 for children aged 18 years old, 90-130 for adult aged above 18 years old*
RR	Respiratory Rate	Integer	0-120
Temp	Body Temperature	Integer	35-46
BP_upper	Systolic blood pressure	Integer	0-350
BP_lower	Diastolic blood pressure	Integer	0-350
Weight	*Weight of patients*	*Float*	*0.0-200.0*
Height	*Height of patients*	*Float*	*0.0-300.0*
WL	Waist of patients	Integer	5-80″
BMI	Body Mass Index	Float	18.50-30.00
VisitType	Service Type	Varchar(255)	Self, Relative, Other
VisitStatus	Service Status	Varchar(255)	Walk, Wheelchair, Other
Normal	*Normal Status*	*Boolean*	*1=true,0=false*
Datetime	*Recorded Date Time*	*Varchar(2)*	*CD, UD*
Triage	Triage Level	Varchar(1)	1-5
Pain Score	Pain Level	Integer	1-10
BMI Level	*BMI Level*	*Varchar(2)*	*L1-L5*

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:project>
    <xs:usecase id="UC001" name="GFR Calculation"/>
        <xs:ucelement name="input">
        <xs:complexType>
        <xs:sequence>
        ⋮  ⋮  ⋮  ⋮  ⋮
        <xs:element id='2' name="Age">
            <xs:simpleType>
            <xs:restriction base="xs:integer">
            <xs:minLength id='1' value="0"/>
            <xs:maxLength id='1' value="17"/>
            <xs:minLength id='2' value="18"/>
            <xs:maxLength id='2' value="120"/>
            </xs:restriction>
            </xs:simpleType>
        </xs:element>
        ⋮  ⋮  ⋮  ⋮  ⋮
        </xs:sequence>
        </xs:complexType>
        </xs:ucelement>
</xs:project>
</xs:schema>
```

Fig. 8. The XML file of UC001 (Version 2)

cause of changes to be identified and analysed. There are five patterns of atomic changes for determining the cause of changes that are 1) change of variable name, 2) change of variable type, 3) change of variable range value, 4) change of number of variables and 5) change of variable order. The possible cause of change patterns affects to the addition, deletion and modification of target variables.

Considering the changed GFR calculation formulas in KFD subsystem as shown in Table IV, for example, the comparison of two versions of use case enables us to discover the causes of variable changes. 1) the modification of variable value is detected as the range value of variable *Age* was changed from *0-120* to *0-17* and *18-120*. Furthermore, 2) the addition of variable is identified as variable *Height* for GFR calculation. The before-changed test cases directly affected from these variable changes are later discovered to be re-generated.

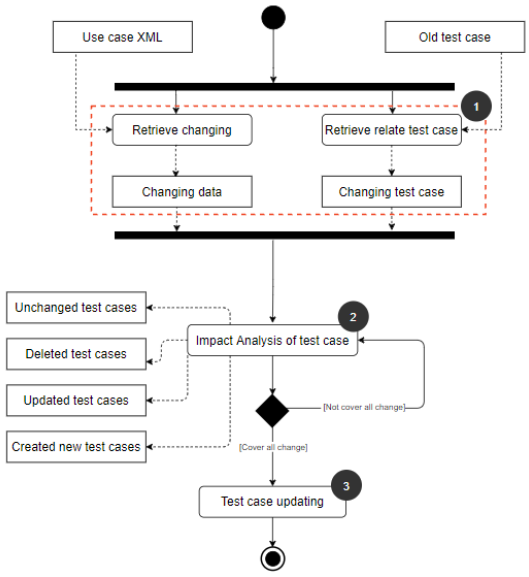


Fig. 9. Procedure of impact analysis of test cases

Step 3: Impact Analysis of Test Cases. After the variable affected by the change is identified and considered, the change impact of test cases is analysed. Fig. 9 shows the procedure of the change impact of test cases that is divided into three steps.

(1) Retrieve Inputs for Impact Analysis: this step retrieves and prepares the necessary inputs used for analysing the change impact of test cases. Two sets of data are prepared in this step: the information of changed variables and a set of before-changed test cases. To prepare this, the information of the changed variables together with the cause of changes is input from the previous step, whereas a set of before-changed test cases is retrieved from the test case database. This results in the information of changing data in affected variables and changing test cases to be generated and used in the next step.

(2) Impact Analysis of Test Cases: the information of changed variables prepared from the previous steps is used to analyse and validate the impact on the before-changed test cases. This results in these test cases being classified into four groups that are *no-change*, *delete*, *update* and *create* categories. Our developed tool also supports this step to analyse effectively the change impact from the multiple various change patterns occurring in parallel at the same time. The patterns for identifying the action of effect on test cases based on type of change impact are summarised in Table IX. The percentage of estimated reusability is also calculated for each pattern.

Considering in the case of 1) the change of variable name, the impact of this case affects the corresponding test cases of this variable to be not changed. Thus, they can be totally reused (100%). For 2) the change of data type, the existing

TABLE IX. SUMMARY OF FIVE PATTERNS OF AUTOMATIC CHANGES WITH EFFECT ON TEST CASES

Type of Impact		Action				%Est.Reuse
		No Change	*Update*	*Delete*	*New*	
Change of variable name	-	×				100
Change of data type	-		×			100
Change of number of variables	Add			×	×	0
	Delete			×	×	0
Change of variable value	Add	×	×		×	50
	Delete	×	×	×		50
	Update		×			100
Change of variable order	-			×	×	0

test cases can be 100% reused with only updating their data type. 3) the change of number of variables, in both cases of adding and deleting the variables, results in all existing test cases not to be totally reused. This is because of the Cartesian product used for generating test cases needs to recalculate the multiplication of all variable range value to support the deletion or addition of variable. This affects the existing test cases corresponding to these variables to be the deleted and the new test cases to be generated. 4) The change of range value of variables is divided into three possible cases (4.1) adding a new range value in which the valid partition is added or extended from the other valid partition. This results in all test cases generated from this partition are totally reused as there is no change in the valid partition. However, this case causes the intersection of range values between the addition valid partition and the other invalid partition. This, therefore, results in the range value of affected invalid partition to be reduced by removing the overlapped value. As a result of this, the existing test cases affected from this reduced invalid partition to be updated and new test cases are generated from the additional valid partition. Similar to this, (4.2) deleting existing range value affects the existing test cases generated from unchanged partition to be totally reused and those generated from deleted partition to be deleted. Furthermore, this case affects the range value of invalid partition to be expanded to cover the deleted partition. Therefore, this results in the test case corresponding to this to be updated. In cases (4.1) and (4.2), the percentage of reusable test cases is estimated to be 50%. In the case of (4.3) updating existing range value that one partition is split into two partitions, this causes the existing test cases to be updated (100% of reusability). Finally, we have discovered that the change number of variables as adding new variables and deleting existing variable affects the order of variable. Thus, 5) the change of variable order affects some exisitng test cases to be deleted and new test cases to be generated.

(3) Test Case Modification: the before-changed test cases are considered to be modified based on the result of impact analysis that is classified into the impact group as mentioned earlier. For the test case that is classified into *no-change* group, it can be totally reused for the new version. However, the before-changed test cases have their value updated or are deleted when the result of impact analysis is classified into the *update* or *delete* groups respectively. If the impact analysis of test cases results in the *create* group, a new test case is generated. All modified and created test cases are generated by using CCTM algorithm.

Table X shows an example of the variable partitions for test case generation after analysing the impact of test cases. As can be seen in this table, based on the requirement change request of GFR Calculation module mentioned before, the range of variable *Age* is changed from one partition (partition *5* in variable before change) for the valid test case, *0-120* years old, to two partitions (partitions *5* and *6* in variable after change) for the valid test case, *0-17* and *18-120* years old. This causes the updated test case to be improved by creating a new version of test cases corresponding to the updated partition as shown in Table XI. The new test cases corresponding to the separated partitions after the change (0-17 and 18-120 years old) are re-generated as in test cases *2-3* (as shown in the highlighted row in the table). Furthermore, from the no-change group of impact analysis, there are two original (unchanged) test cases that are totally reused in the new set of test cases, test cases *1*, *4-5* respectively.

TABLE X. EXAMPLE OF UPDATING THE VALUE OF A VARIABLE AFTER CHANGING

Variables before change					Variables after change				
Class	*Var.*	*Min.*	*Max.*	*Type*	*Class*	*Var.*	*Min.*	*Max.*	*Type*
1	Sex	F	F	Valid	1	Sex	F	F	Valid
2		M	M	Valid	2		M	M	Valid
3		N/A	N/A	Invalid	3		N/A	N/A	Invalid
4	Age	-∞	-1	Invalid	4	Age	-∞	-1	Invalid
5		0	120	Valid	5		0	17	Valid
6		121	∞	Invalid	6		18	120	Valid
					7		121	∞	Invalid

TABLE XI. EXAMPLE OF TEST CASES AFTER CHANGING

#TC	Class	Variable name		Sequence		Type
		Sex	*Age*	*Sex*	*Age*	
1	1,4	F	-4	1	2	Invalid
2	*1,5*	*F*	*12*	*1*	*2*	*Valid*
3	*1,6*	*F*	*55*	*1*	*2*	*Valid*
4	1,7	F	176	1	2	Invalid
...	...	...	...	...	...	...

Step 4: Test Case Generation. The approach performs test case generation when test cases need to be recreated. CCTM, the combination technique of ECP and CTM, is applied to cover generated test cases with all functions as required. The main steps are as follows:

Step 4.1: Analyse Requirements using a Classification Tree:

The classification tree technique basically enables a set of requirements in the specification to be classified as a branch in the tree. The top of the tree as a root is the main use case in the system before the sub use case is considered to extend the root to create the sub-tree. A lower level of the tree, variables of the use case are identified as terminal classification. The terminal class is the range value of the variable and identified at the lowest level as a leaf of the classification tree. There are seven principle combination patterns proposed in [16]. One example of these combination patterns that was revealed by the tool is shown in Fig. 10. The tool detected two redundant classification trees that were built to support the calculation of two age groups as shown in Fig. 10a and 10b respectively. These redundant terminal classes were eliminated by CTM enabling in our tool. The terminal class values were combined into the same tree as shown in Fig. 10c. This influences the

benefit to test case generation step in which the potential redundancy of generated test cases is reduced.

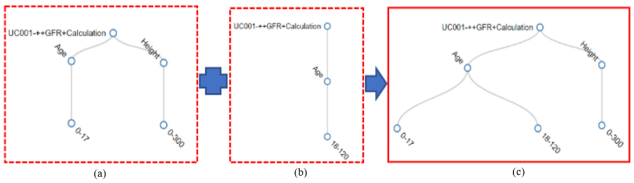


Fig. 10. Example of a combined use case

Fig. 11 shows an example of classification tree of GFR calculation module after the analysis of changes. It contains the variables and their corresponding possible range value visualising in terminal classification (parent node) and terminal class (leaf node) respectively. The range value of all relevant variables demonstrated in this tree can be explained as follows: *Sex = F, M*, *Age = 0-17, 18-120*, *Height = 0-300* and *SCr = 0-0.9, 1.0-10.0*. These are considered to create equivalence class partitioning with ECP technique for the next step.

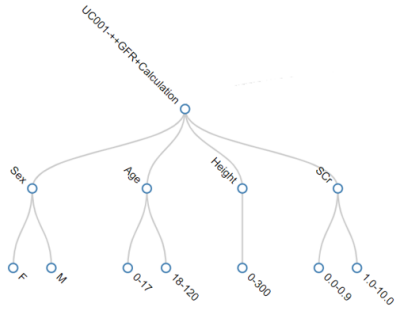


Fig. 11. The classification tree of GFR module

Step 4.2: Generate Test Case from the Equivalence Class Partitioning: This step generates the test case from the terminal class by using the ECP technique enabled by CCTM method. The value of variable indicated in this terminal class is considered to create the partition. In our developed prototype, the partition of changed variables is generated automatically resulting as in Fig. 12. This figure illustrates the partition created from the terminal class in the updated classification tree demonstrated in Fig. 11. Variable *Age* is divided into four partitions (two valid partitions and two invalid partitions) corresponding to the changed version of variables. To generate test cases, ECP performs the Cartesian product of multiplying a data value selected from all partitions.

IV. Proof of Concept

A. Tool Development

A prototype tool was developed to demonstrate the effectiveness of the CIA framework for test cases explained in Section III-A. It is a Java based web application implemented in the Node.js 8.9.4[2], JavaScript run-time environment which is widely used as a standard for large-scale application. MySQL

Fig. 12. Example of equivalence class partitioning

version 5.7.22[3] is used to keep the data of use case version, variables and test data for test suite export.

Fig. 13 demonstrates an example of our developed prototype tool. Fig. 13a is the screen showing two versions (before-change and after-change) of XML file of use case diagram after they are uploaded before the consistency checking performs. Then, the tool performs variable analysis of each version of the XML file. All variables defined in the use case are analysed. This includes variable name, variable type and variable range value as demonstrated in Fig. 13b. Fig. 13c shows the screen of impact analysis result. It shows the summary of impact analysis which is divided into two parts. The first part reports the number of changes classified and grouped into five patterns of atomic changes as we proposed in the framework as mentioned in Section III-A. The reuse percentage of test cases affected from the change is also calculated and reported on this screen. The latter part reports the number of test cases affected from the changes grouped by the causes of changes that are *No Change*, *Delete*, *Update*, *New (Create)*. Finally, test case generation is performed when test cases need to be recreated. CCTM technique is used for test case generation. This results in the classification tree and its corresponding equivalence class partitions is created and analysed as shown in Fig. 11 and 12, respectively before generating the test cases and test data.

B. Tool Validation

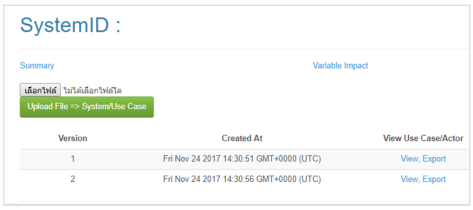
Before evaluating the the effectiveness and efficiency of our proposed approach, we validated the correctness of the tool to confirm whether all functionalities of the tool demonstrated in Section IV-A perform correctly according to the framework presented in Section III-A. Therefore, we conducted experimental validation aiming to answer the following question.

EQ1: *Are all functionalities implemented in the tool implemented and do they perform correctly according to the proposed approach?*

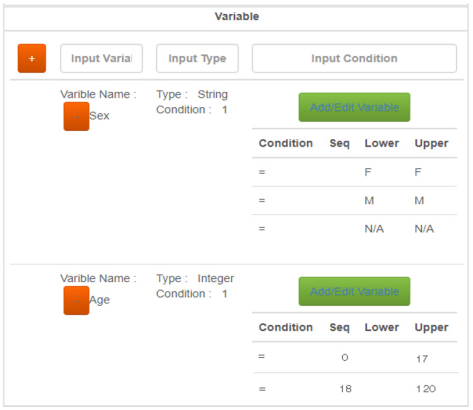
To accomplish this, three test scenarios corresponding to four steps of the CIA algorithm described in were designed as shown in Table XII including 1) *TS-01 : Validate Comparing Two Versions of Variable Function*, 2) *TS-02: Validate Test Case Analysis Function* and 3) *TS-03: Validate CCTM function*. Test scenario *TS-01* aims to validate the comparison

[2]https://nodejs.org/en/about/

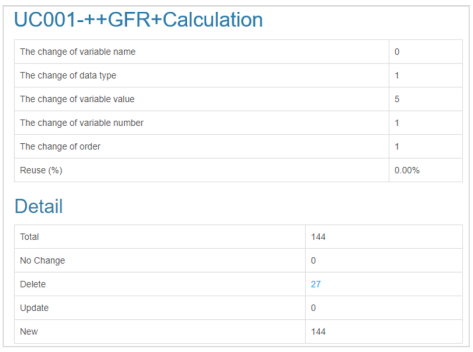
[3]https://www.mysql.com/



(a) screen of XML version



(b) variable management



(c) impact analysis

Fig. 13. Example of the tool screens

operation of versions of variable XML function according to Steps 1 and 2 in CIA algorithm, after two versions of the XML data dictionary containing the data structure of variables are uploaded. The change of variables is compared based on the proposed five patterns. For test scenario *TS-02* aiming at the validation from Step 3 of the algorithm, impact analysis of test case function is validated after the comparison of change of variable based on five patterns. Test scenario *TS-03* is used to validate the operation of classification tree generation by CCTM technique. As the main objective of CCTM technique is to reduce the duplicated test cases by merging the duplicated classification tree caused from the redundant requirements, we also created the scenario for validating this tree merging operation.

Table XIII shows the result of testing. Test cases in test scenario *TS-03* enabled us to reveal the fault in our developed tool. As can be seen in this table, seven test cases were designed from merging two classification approach proposed in [16] and used to validate our merging classification tree

TABLE XII. THE RESULT OF TOOL TESTING

Test Scenario	Corresponding Steps	Result	Revision
TS-01 Validate Comparing Two Versions of Variable Function	Steps 1 and 2	***Fail***	Pass
TS-02 Validate Test Case Analysis Function	Step 3	Pass	-
TS-03 Validate CCTM Function	Step 4	***Fail***	Pass

function. We discovered the fault in test case *TC-037* regarding the case that two use cases have the same variable name but some range value of variable in one use case is overlapped with that of the other. The expected result of this test case should result in three sub-trees to be created after merging tree as the intersection of variable range between two use cases and two exteriors of variable range in both use cases. The test result revealed that our tool created only two sub-trees instead of three sub-trees after merging tree operation. This led us to fix the code to perform the correct operation of this merging tree case. Furthermore, the fault was also found in test scenario *TS-01* caused by the problem of XML data dictionary that could not trace the change of variable. Thus, we redesigned the XML data dictionary and retested this function. All revision tests were passed as we expected.

TABLE XIII. THE TESTING RESULT OF TEST CASE IN TEST SCENARIO *TS-03*

Test Cases	Objective	Pass/Fail
TC-031	The integration of two use cases in which the other use case has no information	Pass
TC-032	The integration of two use cases that have the same terminal sub-tree	Pass
TC-033	The integration of two use cases in which these two use cases have the same variable name and value range	Pass
TC-034	The integration of two sub-use cases in which share the same use case (parent) but have the different variable name and value range	Pass
TC-035	The integration of two use cases in which these two use cases have the same variable name but different value range	Pass
TC-036	The integration of two use cases in which these two use cases have the same variable name but all value range of variable in one use case is in that of the other	Pass
TC-037	The integration of two use cases in which these two use cases have the same variable name but some value range of variable in one use case is overlap with that of the other	***Fail***

Based on experimental validation, the testing result of test scenarios and cases confirms the accuracy and coverage of functionality in which all functionalities corresponding to the steps in CIA algorithm were implemented in the tool and they performed correctly according to the proposed approach.

C. Tool Evaluation

The effectiveness and efficiency of our proposed approach were evaluated by using two real-world case studies, 1) KFD and 2) ONA subsystems, explained in Section III-B. The result of change impact analysis and test case regeneration from these two case studies generated automatically from the tool was compared with that of manual operation. Furthermore, we evaluated the satisfactory level of the developed tool from the target user for potential application in the future. The aim of tool evaluation is to answer the following evaluation questions.

***EQ2**: How error-prone can be identified from the comparison between manual CIA and automated tool?*

***EQ3**: How the impact analysis result can influence the level of reusability of existing use cases?*

***EQ4**: How do potential users assess a satisfactory result of CIA from the automated tool compared with the manual CIA?*

1) Effectiveness evaluation: To answer ***EQ2***, the comparison between the actual result of impact analysis created by the automatic tool and the expected result calculated by experts was conducted. The precision, recall and F-measure computation were calculated by comparing the result produced by the manual CIA and automated tool. The computation metrics were adapted from [17] as follows.

$$Precision = \frac{|\{Expert\,Identified\} \cap \{Tool\,Identified\}|}{|\{Tool\,Identified\}|} \times 100 \quad (1)$$

$$Recall = \frac{|\{Expert\,Identified\} \cap \{Tool\,Identified\}|}{|\{Expert\,Identified\}|} \times 100 \quad (2)$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

Table XIV demonstrates the result of impact analysis. We compared the number of actual test cases after performing impact analysis generated automatically by the tool with the number of expected test cases calculated manually by experts. Considering the calculated F-measure with precision and recall of KFD case study, the accuracy of the automatic tool performing with this case study is very high. This is because the KFD case study is not a complex case study containing basic changes such as renaming a variable name and change the variable type which did not affect much to the test cases. However, as ONA is a more complex case study containing change of variable range value that was not specified in KFD case study, the accuracy of the automatic tool performing with ONA case study in that pattern is low. This also affected the accuracy of analysing pattern change of variable order to be low. After we revealed the tool, we discovered that that tool excluded the additional range value affected from the adjusted value of variables. This led us to correct the tool to include this case when the tool performs test case generation. After fixing the tool, the number of regenerated test cases by the tool is the same as that of expected test cases hand-operated by experts. The result produced by the corrected tool is shown in Table XV.

2) The level of reusability: To answer question ***EQ3*** regarding the level of reusabiliy, we considered the impact analysis result of test case evaluated from two case studies as shown in Table XV. This table shows the total number of test cases generated after change impact analysis of all use cases of two case studies obtained from the developed tool. The amount of reuse equation adapted from [18] is used to assess the level of reusability of test cases as shown in Eq. (4).

$$\%Reuse = \frac{\#\,of\,Reused\,Test\,Cases}{\#\,of\,Total\,Test\,Cases} \times 100 \quad (4)$$

In KFD subsystem, 185 test cases were generated. Based on this number, three types of changes are discovered by the tool. (1) Changing variable name in use case *UC003* resulted in 6 test cases of the first version to be totally reused. Furthermore, 35 test cases to be updated due to changing data type in use case *UC002*. Considering the case of (3) adding a new variable (as *Height*) in use case *UC001*, 27 test cases of version 1 were deleted and 144 of new test cases were generated for version 2. Furthermore, the reusability level of test cases was also measured from the no-change and update group of test cases resulted from the impact analysis. Around 22% of existing test cases for KFD subsystem testing can be reused in the new version of test cases. The reusability rate is low. This is because of the effect from the addition of new variable. As the Cartesian product of multiplying all variable range value is used to generate test cases, all existing test cases affecting to this change were not totally reused. All existing test cases regarding to these variable to be deleted and new test cases to be created by recalculating the Cartesian product instead.

In ONA subsystem, five types that cover all possible changes were identified by the tool. (1) Changing variable name in use case *UC023* affected to 21 test cases of the first version to be totally reused. (2) Changing data type in use case *UC023* resulted in 9 test cases to be updated. Furthermore, (3) adding a new variable as in use case *UC011* caused 6 of new test cases to be generated for after-changed version. (4) Deleting an existing variable in use case *UC01* resulted in 2,592 of test cases to be deleted and 288 of new test cases to be created for version 2. Lastly, there was (5) the addition of variable range value occurred in use case *UC02* of this case study, in which the first case study did not have, was revealed by the tool. This resulted in 4,374 test cases in version 1 to be reused 100 percent (no change), another 4,374 test cases in this before-changed version to be updated and 17,496 of new test cases to be generated. The resuability level of test cases for ONA subsystem is approximately 33% of existing test cases being reused in the new version. This is because there was change of variable range value. This affects half of existing test cases related to this change effect can be totally reused.

3) Satisfaction evaluation: To answer ***EQ4***, we evaluated the satisfaction of our proposed approach and tool from the target user. Practical specialists who have experience in HIS system development for more than five years were selected as the target user. This was also chosen from the wide range of the role in the development team that are 1 development manager, 2 software testers and 1 developer. The evaluation process started with the tool being demonstrated and trained to specialists before they used the tool with the prepared case study. The satisfactory level with the tool was evaluated by using the evaluation form. Likert scale ranking from strongly agree (5) to strongly disagree (1) indicated a satisfactory level in each question of the evaluation form.

Table XVI demonstrates the question used in a satisfactory evaluation form and the evaluation analysis result is shown in Fig. 14. Most specialists strongly agreed that the tool enables the accurate and appropriate impact analysis of changes in test cases and recommends the accurate and appropriate results based on this impact analysis (Q1 and Q2 respectively). Furthermore, all specialists strongly agreed (Q3) that they are confident in the high accuracy level of test case generation provided by this tool. For future application (Q4), 75% of specialists agreed that the tool will be applied to other systems

TABLE XIV. THE RESULT OF IMPACT ANALYSIS

System	Type of atomic changes	# Test cases		Precision	Recall	F-measure
		identified by an expert	*identified by the tool*			
KFD	Change of variable name	6	6	100%	100%	100%
	Change of variable type	35	35	100%	100%	100%
	Change of variable range value	-	-	-	-	-
	Change of number of variables	144	144	100%	100%	100%
	Change of variable order	185	185	100%	100%	100%
ONA	Change of variable name	21	21	100%	100%	100%
	Change of variable type	9	9	100%	100%	100%
	Change of variable range value	26,244	8,748	100%	33.33%	50%
	Change of number of variables	294	294	100%	100%	100%
	Change of variable order	26,568	9,072	100%	34.15%	52%

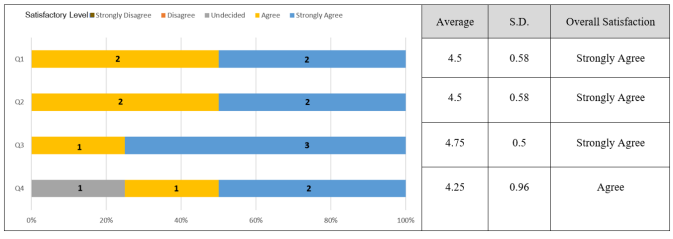
TABLE XV. SUMMARY OF THE IMPACT ANALYSIS OF TEST CASE BASE ON REQUIREMENT CHANGING

System	#Test Cases	Impact	Type of Changes (Number of Test Cases)							
			Variable Name	*Data Type*	*Number of Variable*		*Variable Value*			*Variable Order*
					Add	Delete	Add	Delete	Update	
KFD	185	No Change	6	-	-	-	-	-	-	-
		Update	-	35	-	-	-	-	-	185
		Delete	-	-	27	-	-	-	-	-
		New	-	-	144	-	-	-	-	-
ONA	26,568	No Change	21	-	-	-	4,374	-	-	-
		Update	-	9	-	-	4,374	-	-	26,568
		Delete	-	-	-	2,592	-	-	-	-
		New	-	-	6	288	17,496	-	-	-

in the future. Moreover, specialists also gave feedback in the questionnaire (Q5). Most specialists said that the tool is accurate, reliable and easy to use. The tool enables the benefit in which the test case from the previous can be reused. However, some of them gave very useful feedback for the future improvement. They suggested that the tool should support other input file formats rather than just XML file format. We will use all feedback and suggestions from the specialists to improve the tool in the future. Overall, the specialists were mostly satisfied with our impact analysis tool.

TABLE XVI. SATISFACTION QUESTIONS

Questions	Average
Q1. The tool provides accurate and appropriate analysis of the impact of changes in test cases.	Likert scale (Mandatory)
Q2. The tool can offer accurate and appropriate results and recommendations based on the impact analysis.	Likert scale (Mandatory)
Q3. The tool can generate accurate and appropriate test cases.	Likert scale (Mandatory)
Q4. The tool can be applied to other system case studies in the future.	Likert scale (Mandatory)
Q5. Comments and suggestions for the tool application and improvement	Open-ended question



Average	S.D.	Overall Satisfaction
4.5	0.58	Strongly Agree
4.5	0.58	Strongly Agree
4.75	0.5	Strongly Agree
4.25	0.96	Agree

Fig. 14. Results of the four Likert scale questions

V. LESSON LEARNED AND DISCUSSION

A. Discussion

The key findings discovered from our developed tool is the ability to analyse the impact of test cases based on changes of use case based requirement specification that conforms to the proposed approach and framework. Based on the analysis and evaluation results, the developed tool can perform the impact analysis from the change of the requirements in two versions as before-change and after-change of the use case specification model according to the proposed five patterns of the cause of changes. These patterns are designed specifically for use case based specification including 1) change of variable name, 2) change of variable type, 3) change of variable range value, 4) change of number of variables and 5) changes of variable order. Compared with the study work of [8], they classified the cause of changes into seven patterns in which the first five patterns are the same as our proposed patterns. The other two patterns were not included as they are related to the change on web tag a link that is specific to web application. The decision after the impact analysis is also classified by this tool into four groups as *no-change*, *update*, *delete* and *create* groups as we expected.

Furthermore, CCTM is integrated into this tool to support automatic generation when new test cases require to be created. The validation and evaluation results confirm the key findings that our proposed approach supports all cases for merging two classification trees proposed by [19] as explained in Section IV-B and IV-C. From the validation and evaluation results, we can confirm that CCTM influences the benefit that reduces the time and increases the testing coverage for new test case generation. However, it has the limitation of testing a large and complex system that may not complete with a single classification tree as found in [6], [7]. We suggest that it needs to separate the testing into several testing units. From our experiences, for example, we separated the testing of our SUT,

MRS, into two testing units, one is for KFD subsystem and the other for ONA subsystem.

B. Comparison of the Proposed Approach and Other Approaches

This section describes the comparison of our proposed approach with other approaches by [11], [9], [8]. Five criteria are considered for this comparison including *Impact Analysis Methods*, *System Domain*, *Source Type*, *Test Case Generation Techniques* and *Reusablity Supports* as shown in Table XVII.

Our approach performs the impact analysis from the change of the requirements and classifies the cause of the change into five patterns as mentioned earlier. Comparing this with the approach proposed by [8], the cause of changes of a web application is classified into seven patterns in which the first five patterns are the same as our proposed patterns. Unlike the work of [11] and [9], it does not classify the cause of change into patterns during impact analysis. The existing test cases affected from the change is only analysed whether they can be reused.

Furthermore, as new test cases are required to be generated in the case that new features or requirements are added, CCTM, the hybrid testing approach that integrates ECP with CTM test case generation technique, is integrated into this tool to support automatic generation when new test cases require to be created. With the CTM technique, classification trees merging approach enables the test case redundant from two classification trees to be eliminated. Moreover, ECP technique in the CCTM provides the test case coverage of all possible scenarios as well as failure scenario. Comparing our implemented test case generation technique with the work of [8], [11] and [9]. In the work proposed by [8], their impact analysis framework provides ECP and BVA. These testing techniques only provide the test case coverage of all possible scenarios from the various range of separated partition. Unlike the work proposed by [11] and [9], they only focus on the impact analysis methodology that results in only the consideration of the affected test cases in the old version to be usable or unusable. Their framework, therefore, does not provide test case generation technique for the new test case generation.

Lastly, considering the comparison in terms of the reusability supports as demonstrated in Table XVII, our proposed approach provides the reusability level of test cases as a percentage from the impacted test cases that are implemented in a similar way with the work proposed by [8]. In the work proposed by [11] and [9], they classified only two levels of reusability of the existing test cases that are resuable or unusable.

VI. Conclusion and Future Work

This paper has demonstrated an automatic impact analysis approach of test cases based on changes of use case based requirement specification. Our experiment results with two case studies, KFD and ONA, have shown that the developed tool enables the benefits in which the impact on changes of test cases is analysed from the change of variables in use case specification. Two versions as before-change and after-change of the use case model are compared for consistency checking to detect the change. This results in the cause of variable changes to be classified into five patterns of atomic changes are encoded in this tool. These classified patterns enable the impact on changes of the existing test cases to be analysed whether the existing test cases to be completely reused, partly updated as well as additionally generated. Consequently, the level of reusability of existing test cases is measured and the time to create the whole new cases for the after-changed version is reduced. Furthermore, CCTM, the hybrid test case generation technique encoded in the tool for generating new test cases influences the benefit to increase the level of testing coverage with a minimised number of test cases and reduces the redundant test cases as demonstrated. The beneficial contribution delivered by our proposed approach and tool is also confirmed by the validation and evaluation results from the practical specialists which are consistency with the results discovered from the researchers' perspective.

As our proposed approach and developed CIA tool supports the impact analysis of change of variable types that are only primitive programming data types including integer, floating-point number, boolean, character and string, the adjustment of the tool to support the ready-made "real world" data type e.g. date and time have been considered for the future work. To increase the capability and reliability of the tool, another research issue is the further evaluation of the tool with different system domains as suggested by the practical specialists.

Deployment and Availability

Our developed tool is available at https://sites.google.com/phuket.psu.ac.th/testciatool/. User guide manual document and source of example case studies (KFD and ONA subsystems) are also available on the website.

References

[1] P. Jorgensen, *Software Testing: A Craftsman's Approach*, 3rd ed. Boca Raton, NY: Auerbach Publications, 5 2013.

[2] I. Dewi, Y. Miftahuddin, M. Fattah, C. Palenda, and S. Erawan, "Point of sales system in inhome café website using agile methodology," *Journal of Innovation and Community Engagement*, vol. 1, no. 1, pp. 01–19, Mar 2021.

[3] S. Purwanti, A. Febriani, M. Mardeni, and Y. Irawan, "Temperature monitoring system for egg incubators using raspberry pi3 based on internet of things (iot)," *Journal of Robotics and Control (JRC)*, vol. 2, no. 5, pp. 349–352, 2021.

[4] I. R. Munthe, B. H. Rambe, R. Pane, D. Irmayani, and M. Nasution, "Uml modeling and black box testing methods in the school payment information system," *Jurnal Mantik*, vol. 4, no. 3, pp. 1634–1640, 2020.

[5] S. Sutiah and S. Supriyono, "Software testing on e-learning madrasahs using blackbox testing," *IOP Conference Series: Materials Science and Engineering*, vol. 1073, no. 1, p. 012065, Feb 2021.

[6] M. Grochtmann and K. Grimm, "Classification trees for partition testing," *Software Testing, Verification and Reliability*, vol. 3, no. 2, pp. 63–82, 1993.

[7] M. Grochtmann, K. Grimm, J. Wegener, and D.-b. Ag, "Tool-supported test case design for black-box testing by means of the classification-tree editor," in *Proceedings of the EuroSTAR (EuroSTAR, 1993)*, 1993.

[8] S. Phetmanee and T. Suwannasart, "A tool for impact analysis of test cases based on changes of a web application," in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2015*, 2015, pp. 497–500.

[9] T. Sakkarinkul and T. Suwannasart, "Test case impact analysis from use case description changes," in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2015*, 2015, pp. 523–527.

TABLE XVII. COMPARISON OF FEATURES AND CAPABILITIES OF OUR PROPOSED APPROACH WITH COMPARABLE WORK

Features	**Proposed Approach**	**[8]**	**[11]**	**[9]**
Impact Analysis Methods	Indicate and classify the impact analysis result into five patterns	Use seven patterns of the change of a web application to classify the impact analysis result	Analyse the impact from the change list of use case specification but do not classify the pattern of changes	Analyse the impact from the change list of use case specification but do not classify the pattern of changes
System Domain	Hospital Information System	Web Application	Web Application	Web Application
Source Type	Use Case Spec.	GUI spec.	Use Case Spec.	Use Case Spec.
Test Case Generation Techniques	Use the CCTM (ECP and CTM) technique	Use ECP and BVA techniques	No test case generation	No test case generation
Reusablity Supports	Calculate the percentage of the reusability level of test cases	Calculate the percentage of the reusability level of test cases	Only indicate that the impacted test cases can be reused or updated	Only indicate that the impacted test cases can be usable or unusable

[10] A. Intana and T. Sriraksa, "Impact analysis framework of test cases based on changes of use case based requirements," in *Proceedings of the 23rd International Computer Science and Engineering Conference (ICSEC, 2019)*. IEEE, 2019, pp. 230–235.

[11] M. Raengkla and T. Suwannasart, "A test case selection from using use case description changes," *Lecture Notes in Engineering and Computer Science*, vol. 2202, pp. 507–510, Mar 2013.

[12] C. Sriarpanon and T. Suwannasart, "A source code and test cases impact analysis tool for database schema changes," *Lecture Notes in Engineering and Computer Science*, vol. 1, pp. 466–469, Mar 2015.

[13] A. Kampeera and T. Suwannasart, "Impact analysis to database schema and test cases from inputs of functional requirements changes," in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2016*, Mar 2016, pp. 449–453.

[14] N. Cherdsakulwong and T. Suwannasart, "Impact analysis of test cases for changing inputs or outputs of functional requirements," in *2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2019, pp. 179–183.

[15] WHO, "Mean body mass index," https://www.euro.who.int/en/health-topics/disease-prevention/nutrition/a-healthy-lifestyle/body-mass-index-bmi, Jun 2021.

[16] B. Ramadoss and P. Prema, "An approach for merging two classification-trees," in *Proceedings of the IEEE International Advance Computing Conference 2009*, 2009, pp. 1602–1607.

[17] K. M. Ting, *Precision and Recall*. Boston, MA: Springer US, 2010, pp. 781–781. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_652

[18] A. L. Imoize, D. Idowu, and T. Bolaji, "A brief overview of software reuse and metrics in software engineering," *World Scientific News*, no. 122122, p. 56–70, 2019.

[19] B. Ramadoss, P. Prema, and S. R. Balasundaram, "Combined classification tree method for test suite reduction," in *Proceedings on International Conference and workshop on Emerging Trends in Technology (ICWET, 2011)*, no. 11, 2011, pp. 27–33.