World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:6, 2008

A Classification Scheme for Game Input and Output

P. Prema, and B. Ramadoss

***Abstract*—**Computer game industry has experienced exponential growth in recent years. A game is a recreational activity involving one or more players. Game input is information such as data, commands, etc., which is passed to the game system at run time from an external source. Conversely, game outputs are information which are generated by the game system and passed to an external target, but which is not used internally by the game. This paper identifies a new classification scheme for game input and output, which is based on player's input and output. Using this, relationship table for game input classifier and output classifier is developed.

***Keywords*—**Game Classification, Game Input, Game Output, Game Testing.

I. Introduction

COMPUTER game industry [1] has experienced exponential growth in recent years. A game [2] is a recreational activity involving one or more players. It can be defined by i) a goal that the players try to reach, ii) some set of rules that determine what the players can or cannot do. Games are played primarily for entertainment or enjoyment, but may also serve an educational or simulation role. A Game [3] presents a virtual reality in which players enjoy interacting with the virtual beings and taking challenges presented by the environment without facing any physical risk. A huge variety of games are currently available [4].

Game input [5, 6, 7] is an information such as data commands, etc., which is passed to the game system at run time from an external source (player, another system (such as PDA, mobile phone, etc.,)), which is used in some way to modify internal data. An example is a player's input. Conversely, game outputs are information which are generated by the game system and passed to an external target, but which is not used internally by the game. For example, game system supplies feedback to the player.

Testing [1] is the process of finding differences between the expected behavior specified by system models and the observed behavior of the implemented system. Graphical User Interface (GUI) [8] significantly changes and interaction concept between humans and computers.

P. Prema (Research Scholar) is with the Department of Computer Applications, National Institute of Technology, Tiruchirapalli, 620015, India (e-mail: mrgprem@ yahoo.com).

B. Ramadoss (Professor) is with the Department of Computer Applications, National Institute of Technology, Tiruchirapalli, 620015, India (e-mail: brama@nitt.edu).

Game testing raises some difficulties: 1) Testing game is required to help ensure the usability and safety of an entire software system. Testing is, in general, labor and resource intensive. 2) Game testing is based on GUI testing. GUI testing has characteristics different from those of traditional software, and thus, techniques typically applied to software testing are not adequate. Current GUI [9] testing techniques are incomplete, ad-hoc, and largely manual. Thus, there is a strong need for new techniques to automate the testing and maintaining the games. To maintain a game system we need to design for testability.

The following are the weaknesses for interactive systems [8]: i) uniformly handling input and output which require different methods to interpret and respond. ii) Lack of the potential to support multiple input devices and output devices, but it is also necessary to distinguish different devices of the same kind. Input and output classification scheme for game can be used to address these problems.

A proposed game classification scheme is used to design and develop a new game according to player's specified criteria such as build up a game according to player's some input classifiers and output classifiers. Testers can use this classification scheme to test and automate the game systems.

In this paper the following section describes the related work on input-output classifiers and dependency. Section three presents a new classification scheme for game input and output. Section four describes the relationship tables for input and output classifiers. Finally, conclusion is provided in section five.

II. Previous Work

Shaw [6] has defined a new model for the deficiencies of batch-style input-output for modern interactive systems. Interactive input and output are fundamentally different from conventional implementations of input and output in two ways: 1) the output device serves as a continuous sensor or observer of the game and provides current information about the state of the game, whereas conventional input and output provides information to the player only when the application software chooses to report. 2) Input is an interactive process requiring feedback where as input is conventionally treated as a simple parsing task. Moreover, interactive input is often under control of the player rather than the program. Shaw has identified the following input and output categories. The input categories are: user-based input, state based input and the categories for output are: output based on state and output to

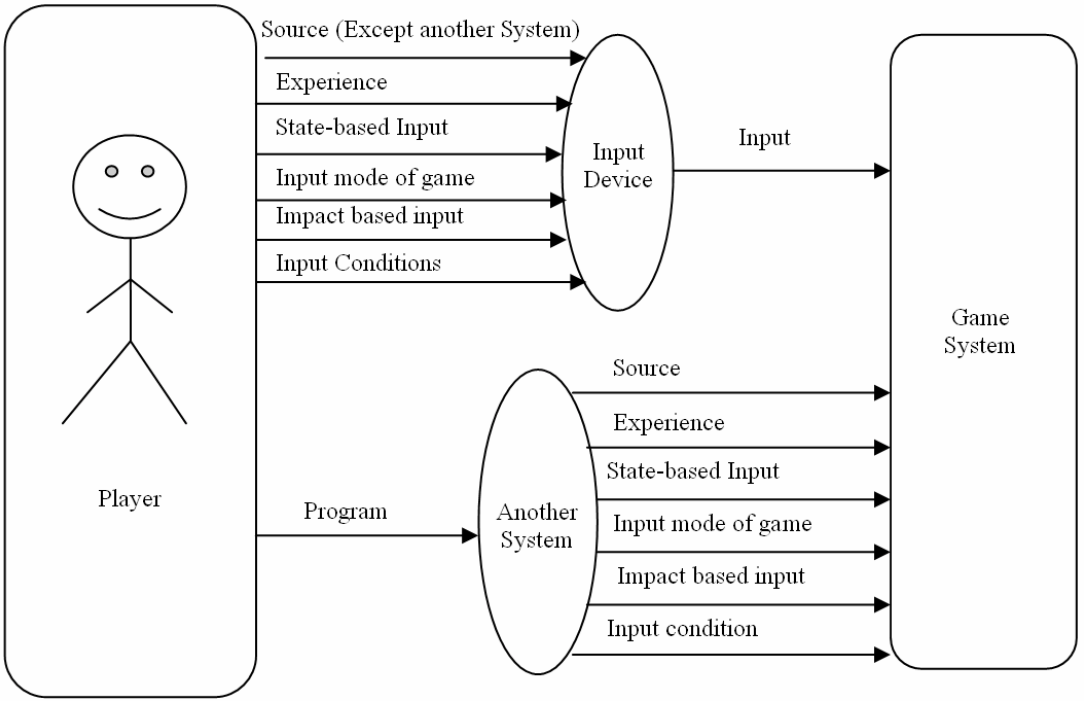

Fig. 1 input classifiers for game system

the player. Lakos [10] defined dependency as a component Y that depends on a component X iff X is needed in order to compile or link Y. Jungmayr [11] identified dependencies that are critical for testing, based on a set of testability metrics. They focus on the structural aspect of testability and presented a novel approach that allows identifying local dependencies which are critical for global testability.

There may not be not much current research work done based on our game classification scheme.

III. New Classification Scheme for Game Input and Output

This section describes the classifiers of game input and output with an example. The input to the game system would be a sequence of inputs that occur in a certain time order. A sequence of inputs is a series of inputs that is applied sequentially to the game system. For example, some inputs may be required (this means the inputs are required to play the game) and others may be optional (this means that the inputs may enter optionally).

A player may be allowed to enter inputs in any order (this means the inputs may be entered randomly), or a specific order (this means the inputs may be entered in a particular order) may be required.

If the inputs of the game system are concurrent (execution of two or more independent, interacting inputs over the same period of time; their execution can be simultaneous), the system response depends on global setting and state (condition) of the game. Some inputs may have been selected but for which the same input cannot be selected again throughout the game is called single-use inputs. This type of inputs is used only in a particular state. That is, it represents selection without replacement. For example, in the car race game select car color, it cannot be selected again during the game play. Selected Inputs must be used continuously in throughout the game is called continue-use inputs.

Similarly, the output to the game system would be a sequence of outputs that occur in certain time order. A sequence of outputs is a series of inputs that is applied sequentially to the game system.

Some outputs may have been displayed but for which the same output cannot be displayed again throughout the game. This type of outputs can be used only in a certain state. For example, in the car race game, the system displays 'wrong way' message and it can be displayed only when the car is going on the wrong way. Some outputs have been displayed and must continue to be used in throughout the game.

Based on these above observations we have identified a number of input classifiers and output classifiers. The following sections describe the details about it.

A. Input Classifiers

Game inputs can be identified and classified into seven classifiers (see Fig. 1) and each classifier can be subdivided into a number of sub classes (see Table I). The input classifiers of the game system are IC = $\{ic_1, ic_2 \ldots ic_n\}$. For each input classifier ic_a, the subclasses are $SC_a = \{sc_{a1}, sc_{a2} \ldots sc_{am}\}$.

Symbolically, *the input classifiers of a game system are represented as* $(ic_a, \{sc_{a1}, sc_{a2} \ldots sc_{am}\})$, *where* $ic_a \in IC$ *is an input classifier and each* sc_{ab} *is a subclass of input classifier* ic_a.

In Fig. 1, player can produce all types of information to a game system through the input device and also the player can write or setup some program, which can send input to game system. Table I represents the number of input classifiers and

TABLE I
INPUT CLASSIFIERS AND THE CORRESPONDING SUBCLASSES WITH AN EXAMPLE

Classifier	Subclass and its description	Example
source (S) – is a place where input data originated. For example, a player or external system supplies some information to the game system.	S_1*) player*: of a game is a person who participates in or is skilled at some game. with the help of input devices, the player produces some input to the game system.	in car race game, a player pressing the up arrow key which is used to move the car forward.
	S_2*) another system*: is also a system but it is not a game system. it provides the input data to support the data requirements to the game system.	if game entered into the minimum point requirement, the system automatically produces some input to the game system to control the game.
experience (E) – as a concept comprises knowledge of or skill in or observation of game gained through involvement in or exposure to that game.	E_1*) new player*: is a new comer to a particular game. that is a player has no idea about the game and continually fails to produce some input to game system and does not learn from their mistakes is called new player.	new player may not go to the next higher level within the time limit.
	E_2*) normal player:* has some knowledge to play the game but they have no idea about game special features.	player can complete the game level and go to next higher level with fewer points.
	E_3*) experienced player:* has more experience through practice and learning in a particular game.	player completes the level as early as possible, based on earning extra points.
state-based Input (SI) – A state captures all the current information in a game system and it depends on player input and difficulty of game setting. game inputs may vary depends on the state. based on availability of inputs we classified this classifier into two subclasses.	SI_1*) game-wide input:* is also called normal game control input. Some type of inputs available throughout the game. that is any state can access this type of inputs.	in some games player can press the 'learn to play' button any time throughout the game.
	SI_2*) available in some states:* of input is available in one or more than one states but not available in game-wide. that is this type of inputs may have been available only in one or some specified states but not available in throughout the game.	in car race game, the 'down arrow' key is used to reduce the car speed when the car is in high speed state, and also used to move the car backward when the car is near the tree.
input device (ID): is designed for gaming that can receive input from the player.	ID_1*) keyboard:* is a main input device for computer games. it typically controls movement of the character and also used for aiming the character.	a player can press the keyboard 'up arrow key', 'down arrow key'.
	ID_2*) mouse:* is a handheld pointing device. it may control the movement of the character and game camera / used for aiming. a mouse click is the action of pressing a button on a mouse in order to trigger an action.	The player can click the mouse left button, right button.
	ID_3*) virtual reality:* is a computer created sensory experience that allows a player to believe and barely distinguish a 'virtual' experience from a real one. directional sound, tactile and force feedback devices, voice recognition and other technologies are being employed to enrich the immersive experience and to create more 'sensualized' interfaces.	speaker, touch screen monitor.
	ID_4*) game pad:* is also called joy pad. it is a hand held device which is used to send position information and button presses to the computer. It is obviously designed for games, allowing the player to control objects like planes and cars on the screen.	a player can press the game pad action button.
	ID_5*) paddle:* is a type of game controller held with a round wheel and one or more than one fire buttons.	a player can press the fire button in the paddle.
	ID_6*) light gun:* is used to shoot targets in a game. It usually resembles roughly like firearms or ray guns. It is used in rail shooter or shooting gallery games.	pressing the button to shoot the enemy.
input Mode of game IIM): mode represents "modes of difficulty". based on experience the player can select the game mode. based on game difficulty we have classified this by n-ways. some games may use three levels (easy, normal and hard).	IM_1*) level-1: t*he input to the system is based on level-1. normally, the new player can select this level.	in panda-craze game, to complete level-1 the player can use 'left arrow, right arrow, down arrow, up arrow' keys and mouse left button.
	IM_2*) level-2:* the input to the system is based on level-2. usually new player or normal player can select this level.	in panda-craze game, to complete the level-2 the player can use the keyboard keys, 'left arrow, right arrow, down arrow, up arrow, S, D', and mouse left button.
	IM_n*) level-n:* the input to the system is based on level-n. this level contains some challenges to complete this level. experienced player can select this level.	in panda-craze game, to complete the level-1 the player can use 'left arrow, right arrow, down arrow, up arrow, S, D, space bar' keys and mouse left button.
impact based Input (II): has different degree of impact on the game. a player or another system can produce some input to the game system, which affects certain state, some states or global states of the game are called *impact-based input. b*ased on effect of the game	II_1*) affect certain state:* inputs can affect only a particular state of the game.	in tiny car game, press the 'right arrow key' is used to steer right. this input can affect certain state only.
	II_2*) affect some states:* inputs can affect some states of the game.	in Zuma game, press mouse left button is used to shoot the ball and it can affect next few states.
	II_3*) affect all states:* inputs can affect all the states of the game.	in tiny car game, the player select the car color can affect all the states.

input we have classified this classifier in to four ways.	*II_4) not affect any state:* input cannot affect any state of the game.	in some games, player press 'read instructions' button it display a new window with game information.
input Condition (IC): there may be some rules to select inputs for games. for example, some inputs may be required and others may be optional. A player may be allowed to enter inputs in any order or a specific order. based on these we have classified this classifier.	*IC_1) required:* inputs are need to entered compulsory by the player to play the game. the required input may be entered in sequence or any order.	In tiny car game, a player must enter the player name, which is required to play the game.
	IC_2) optional: a player may randomly select input from the set of choices and it is not compulsory to play the game is called *optional input.*	in car race game, a player can select the mode from the set of choices (window mode or full screen) is optional.
	IC_3) selection: a player should randomly select input from the set of choices with or without replacement based on certain game condition is called *selection based input.*	in panda-craze game, a player can select the level, which the player wants to play.

the corresponding subclasses with description and an example. In this table, the classifier 'state-based input' and the subclasses 'player' and 'experienced player' are based on Shaw [6].

B. Output Classifiers

Game outputs can be identified and classified into six classifiers (see Fig. 2) and each classifier can be subdivided into a number of sub classes (see Table II). The output classifiers of the game system are OC = $\{oc_1, oc_2 \ldots oc_x\}$. For each oc_p, the subclasses are $SC_p = \{sc_{p1}, sc_{p2} \ldots sc_{py}\}$. Symbolically, *the output classifiers of a game system are represented by* $(oc_p, \{sc_{p1}, sc_{p2} \ldots sc_{py}\})$, *where* $oc_p \in OC$ *is an output classifier and each* sc_{pq} *is a subclass of output classifier* oc_p.

In Fig. 2, game system can send all type of information to the player through the 'output device'. Similarly, the game system sends all type of information to another system which is connected to the corresponding game system. This system can then send all the information to the player later. Table II represents the number of output classifiers and the corresponding subclasses with description and an example. In this table, the classifier 'state-based output' and the subclass 'player' are based on Shaw [6].

IV. Relationship Tables for Input and Output Classifiers

This section proposes two new tables called *'Input Relationship Table' (IRT)* for input classifiers and *'Output Relationship Table' (ORT)* for output classifiers which are used to construct test cases. Relationship tables are developed based on the following definitions:

Definition 1: Let X and Y be subclasses. Two subclasses are said to be *exclusive* (⊗) iff these two subclasses cannot be selected or displayed at the same time. That is any one of those select or present at a time.

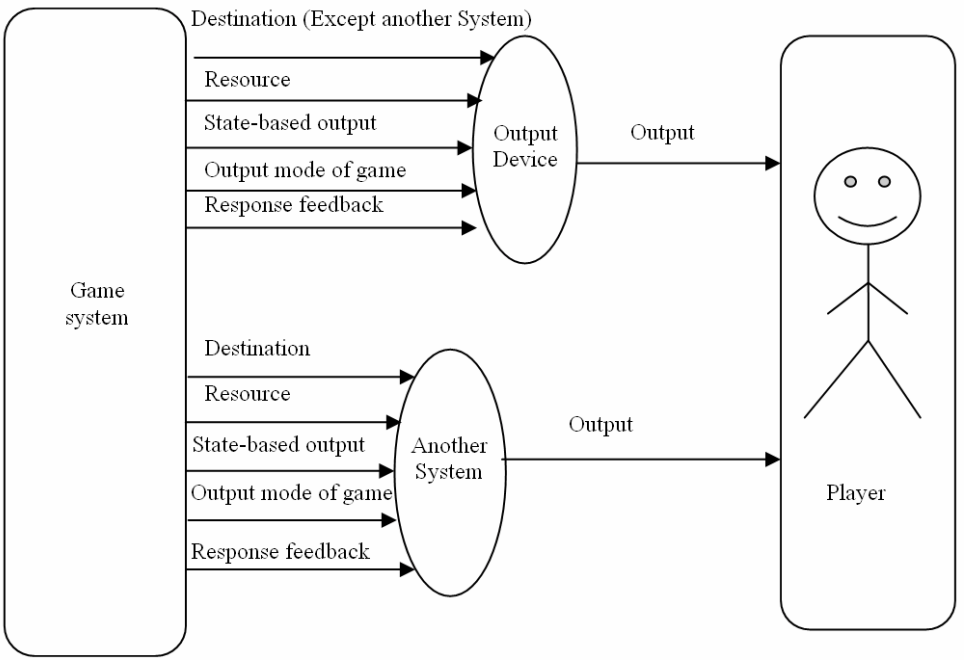

Fig. 2 Output classifiers for the game system

TABLE II
OUTPUT CLASSIFIERS AND THE CORRESPONDING SUBCLASSES WITH AN EXAMPLE

Classifier	Subclass and its Description	Example
destination (D) – is a place where a thing is sent. for example, a game system produces some information to the player or another system.	*D_1) player*: game system displays some information to the player through the game output device (such as virtual reality, etc).	game system displays 'game over' message to the player.
	D_2) another system: game system displays some information to another system.	game goes to next higher level, output to mobile phone, and output to PDA.
resource (RE) - is an item required to perform a task(s). it can be any persons, equipment, material or anything else needed to play game and / or complete the game level. the resource can be added by cumulative basis for each level or depends on game and the final resource can be displayed to the player or sent to another system.	*RE_1) game score:* output by a game system is based on point and point is stored in the resource.	in zuma game, the point can be added and displayed after each level end and total can be displayed at the end of game.
	RE_2) tools: output by a game system is based on equipment (weapons, etc.,), money etc., and it can be stored in resource.	in air-strike game, the weapons will be added after each weapon can be collected from the enemy and it will be stored and displayed in the game screen.
	RE_3) time: Output by a game system is based on time. this is stored in resource.	in zuma game, the time will be added after each level end and total time can be displayed at the end of game.
state-based output (SO) –is based on the situation of a particular state. outputs may vary depends on the state. some outputs may display at one state but for which the same output cannot be displayed again throughout the game, and some outputs may display at any state of the game.	*SO_1) game-wide output:* are displayed throughout the game. that means the output may display at any state of the game.	the system produces some feedback and display to the player.
	SO_2) available in some states: outputs are available only in one state or more than one state but not available in game-wide. that is this type of outputs may display only in some specified states but not available in throughout the game.	in zuma game, 'game stats' window can be displayed at end of each level state and also displayed at end of game state.
output device (OD): serves as a continuous sensor or observer of the game and provides current information about the state of the game to the player.	*OD_1) monitor:* is a device similar to a television screen that receives video signals from the game system and displays information to the player.	in panda craze game, game system displays the player name in the monitor.
	*OD_2) virtual reality: i*s a device that allows the player to perceive the virtual environment in a natural manner by any of audio, visual, haptic etc. that is game system produce some output to player through virtual reality.	head phone.
output mode of game (OM): in game, mode represent mode of difficulty. Based on the selection of player input the output may vary. game mode can be classified into n subclasses. Some games may use three levels (easy, normal and hard).	*OM_1) level-1:* the output by the game system is based on level-1 is called *level-1 based output.*	in panda craze game, only one zoo keeper can be displayed in this level-1.
	OM_2) level-2: the output by the system is based on level-2 is called *level-2 based output*.	in panda craze game, two zoo keepers can be displayed in this level-2.
	OM_n) level-n: the output by the system is based on level-3 is called *level-3 output.*	in panda craze game, more than two zoo keepers with extra features are displayed in this level.
response feedback (RF): is an output to the player that affects its behavior at some point in the future. the conditions under which outputs cause certain responses need to be studied. game system can produce some output to the player. For this player response is required.	*RF_1) required player response: g*ame system produces some output to the player and player need to produce some input to the game system is called *require player response* based output.	in some games, if the player wants to restart the game, the system asks the confirmation. for this player response is needed.
	RF_2) not require player response: game system produces some output to player. but player response is not necessary to play the game.	in Zuma game, if the balls go to skull mouth, system displays the number of lives left message and if it is greater than zero, it automatically go to the game play state.

TABLE III
IRT FOR INPUT CLASSIFIERS

Classifiers and subclasses		S		E			SI		ID						IM			II				IC		
		S_1	S_2	E_1	E_2	E_3	SI_1	SI_2	ID_1	ID_2	ID_3	ID_4	ID_5	ID_6	IM_1	IM_2	IM_3	II_1	II_2	II_3	II_4	IC_1	IC_2	IC_3
S	S_1	–	⊗	⊗	⊗	⊗	×	×	⇒	⇒	⇒	⇒	⇒	⇒	×	×	×	×	×	×	×	⇒	⇒	⇒
	S_2		–	⊗	⊗	⊗	×	×	⊗	⊗	⊗	⊗	⊗	⊗	×	×	×	×	×	×	×	⊗	⊗	⊗
E	E_1			–	⊗	⊗	×	×	⇒	⊗	⊗	⊗	⊗	⊗	⇒	⊗	⊗	×	×	×	×	⇒	⇒	⇒
	E_2				–	⊗	×	×	⇒	⇒	⊗	⊗	⊗	⊗	⇒	⇒	⊗	×	×	×	×	⇒	⇒	⇒
	E_3					–	×	×	⇒	⇒	⇒	⇒	⇒	⇒	⇒	⇒	⇒	×	×	×	×	⇒	⇒	⇒
SI	SI_1						–	⊗	×	×	×	×	×	×	×	×	×	⇒	⇒	⇒	⇒	×	×	×
	SI_2							–	×	×	×	×	×	×	×	×	×	⇒	⇒	⇒	⇒	×	×	×
ID	ID_1								–	⊗	⊗	⊗	⊗	⊗	×	×	×	×	×	×	×	⇒	⇒	⇒
	ID_2									–	⊗	⊗	⊗	⊗	×	×	×	×	×	×	×	⇒	⇒	⇒
	ID_3										–	⊗	⊗	⊗	×	×	×	×	×	×	×	⇒	⇒	⇒
	ID_4											–	⊗	⊗	×	×	×	×	×	×	×	⇒	⇒	⇒
	ID_5												–	⊗	×	×	×	×	×	×	×	⇒	⇒	⇒
	ID_6													–	×	×	×	×	×	×	×	⇒	⇒	⇒
IM	IM_1														–	⊗	⊗	×	×	×	×	×	×	×
	IM_2															–	⊗	×	×	×	×	×	×	×
	IM_3																–	×	×	×	×	×	×	×
II	II_1																	–	⊗	⊗	⊗	×	×	×
	II_2																		–	⊗	⊗	×	×	×
	II_3																			–	⊗	×	×	×
	II_4																				–	×	×	×
IC	IC_1																					–	⊗	⊗
	IC_2																						–	⊗
	IC_3																							–

Note: '–' represents two subclasses are same
'⊗' represents two subclasses are exclusive
'×' represents two subclasses are independent
'⇒' represents two subclasses are dependent

TABLE IV
ORT FOR OUTPUT CLASSIFIERS

Classifiers and sub classes		D		RE			SO		OD		OM			RF	
		D_1	D_2	RE_1	RE_2	RE_3	SO_1	SO_2	OD_1	OD_2	OM_1	OM_2	OM_3	RF_1	RF_2
D	D_1	–	⊗	⇒	⇒	⇒	⇒	⇒	⇒	⇒	×	×	×	⇒	⇒
	D_2		–	⇒	⇒	⇒	⇒	⇒	⊗	⊗	×	×	×	⊗	⊗
RE	RE_1			–	⊗	⊗	×	×	×	×	⇒	⇒	⇒	×	×
	RE_2				–	⊗	×	×	×	×	⇒	⇒	⇒	×	×
	RE_3					–	×	×	×	×	⇒	⇒	⇒	×	×
SO	SO_1						–	⊗	×	×	⇒	⇒	⇒	×	×
	SO_2							–	×	×	⇒	⇒	⇒	×	×
OD	OD_1								–	⊗	×	×	×	⇒	⇒
	OD_2									–	×	×	×	⇒	⇒
OM	OM_1										–	⊗	⊗	⇒	⇒
	OM_2											–	⊗	⇒	⇒
	OM_3												–	⇒	⇒
RF	RF_1													–	⊗
	RF_2														–

Note: '–' represents two subclasses are same
'⊗' represents two subclasses are exclusive
'×' represents two subclasses are independent
'⇒' represents two subclasses are dependent

Definition 2: Let X and Y be subclasses. Two subclasses are said to be *independent* (×) iff there is no relationship between X and Y. That is these two are select or present at the same time.

Definition 3: Let X, Y be subclasses. Two subclasses are said to be *dependent* (⇒) on each other iff Y depends on X or X depends on Y. That is X is needed to select Y or Y is needed to select X.

Table III shows IRT. In this table S, E, SI, ID, IM, II, IC are input classifiers and S_1, S_2, E_1, E_2….IC_3 are subclasses. Let $ic_{i,j}$ denote the element of the ith row and the jth column in IRT. In this table, for $ic_{1,8}$ the value is '⇒'. It means the subclass ID_1 is dependent on S_1. Similarly, for $ic_{1,9}$, $ic_{1,10}$, $ic_{3,8}$, etc are also dependent. For $ic_{1,6}$ the value is '×'. It means the subclasses S_1, SI_1 are independent.

Similarly, for $ic_{1,7}$, $ic_{3,6}$, etc are also independent. For $ic_{1,2}$ the value is '⊗'. It means the subclasses S_1, S_2 are exclusive. Similarly, for $ic_{2,8}$, $ic_{2,9}$, $ic_{3,4}$, etc., are exclusive. For $ic_{1,1}$ the value is '–'. It means the subclass S_1 with itself represents two subclasses are same. Similarly, for $ic_{2,2}$, $ic_{3,3}$, $ic_{4,4}$, $ic_{5,5}$, $ic_{6,6}$, etc., are also same. In Table III it is not required to consider same combinations. It is further clear that $ic_{i,j} = ic_{j,i}$, $\forall\ i \neq j$. For simplicity, the table consists of only above the diagonal entries.

Table IV shows the ORT. In this table, D, R, SO, OD, OM, IO, RF are output classifiers and D_1, D_2, R_1, R_2….RF_2 are subclasses. Let $oc_{p,q}$ denote the element of the pth row and the qth column of the ORT. In this table, for $oc_{1,4}$ the value is '⇒'. It means the subclass RE_1 is dependent on the subclass D_1. Similarly, for $oc_{1,5}$, $oc_{1,6}$, $oc_{2,7}$, etc., are also dependent. For $oc_{4,8}$ the value is '×'. It means the subclasses RE_2 and OD_1 are independent. Similarly for $oc_{4,9}$, $oc_{6,9}$, etc., are also independent.

Similarly for $oc_{4,9}$, $oc_{6,9}$, etc., are also independent. For $oc_{1,2}$ the value is '⊗'. It means the subclasses D_1 and the subclass D_2 are exclusive. Similarly, for $oc_{3,4}$, $oc_{4,5,}$ etc., are exclusive. For $oc_{1,1}$ the value is '–'. It means the subclass D_1 with itself represents two subclasses are same. Similarly for $oc_{2,2}$, $oc_{3,3}$, $oc_{4,4}$, $oc_{5,5}$, $oc_{6,6}$, etc., are also same. In Table IV it is not required to consider same combinations. It is further clear that $oc_{p,q} = oc_{q,p}$, $\forall\ p \neq q$. For simplicity, the table consists of only above the diagonal entries.

The IRT and ORT tables can be used to identify test cases by selecting one subclass from each classifier. From these tables the coverage criteria for selecting test cases is identified:

Exclusive Coverage Criteria - Let EC be a set of all exclusive subclasses, EC = {ec_1, ec_2…ec_n}. Let test case TC :{ t_1, t_2…t_m }, where t_1, t_2…t_m are subclasses iff, any of EC ∈ t_x, ($1 \leq x \leq m$). That is each ec_i ($1 \leq i \leq n$) appear at least one time. Formally, a set of test data TS satisfies the exclusive coverage criteria, where TS = {TC_1, TC_2 … TC_p} such that each element of EC appear at least one time.

V. Conclusion

A new classification scheme for game input and output has been developed based on which the relationship tables (which is used to identify the test cases) namely IRT for game input classifiers and ORT for game output classifiers is developed. This method is used to "eliminate" some invalid test cases, since IRT and ORT the exclusive relationship clearly indicates which subclasses can be combined together and which subclasses cannot be combined together. Future work includes measuring the testability based on the amount of effort needed to test the game.

References

[1] K. Claypool, M. Claypool, "Teaching Software Engineering Through Game Design", Proceedings of the 10th annual SIGCSE conference on innovation and technology in computer science education, 2005, Vol.37(3), pp. 123-127.
[2] http://en.wikipedia.org/wiki/Portal:Sports_and_games
[3] C. Crawford, The Art of Computer Game Design, Columbus: McGraw-Hill, 1984.
[4] D. Johnson, and J. Wiles, "Computer Games with Intelligence", The 10th IEEE International Conference on Fuzzy Systems, 2001, Volume 3, pp. 1355-1358.
[5] http://en.wikipedia.org/wiki/Game
[6] M. Shaw, "An Input-Output Model for Interactive Systems", Proceedings of the SIGCHI conference on Human factors in computing systems, (CHI '86), ACM Press, New York, 1986, pp. 261-273.
[7] P. Dickinson, Instant Replay: Building a Game Engine with Reproducible Behavior. http://www.gamasutra.com/features/20010713/dickinson_01.htm
[8] H. Jiang, G. Drew Kessler, J. Nonnemaker, "DEMIS: A Dynamic Event Model for Interactive Systems", VRST'02, ACM, 2002.
[9] A. M. Memon, "GUI Testing: Pitfalls and Process", Computer, 2002, Volume 35, Issue 8, pp. 87-88.
[10] Lakos, J., Large-Scale C++ Software Design, Addison-Wesley, 1996.
[11] S. Jungmayr, "Identifying Test-Critical Dependencies", In Proceedings of the International Conference on Software Maintenance, IEEE Computer Society, October 2002, pp- 404-413.

Paramasivam Prema received her M.C.A degree from The University of Madras in the year 1999. She has more than seven years of experience in academic / research and industry. She is currently doing Ph.D in department of Computer Applications, National Institute of Technology, Tiruchirapalli. Her current research area includes Software Testing and Software Quality.

Balakrishnan Ramadoss received the M.Tech degree in Computer science and Engineering in 1995 from the Indian Institute of Technology, Delhi. The Ph.D degree in Applied Mathematics in 1983 from Indian Institute of Technology, Powai. Currently he is working as a Professor of Computer Applications at National Institute of Technology, Tiruchirapalli. His current research area includes: Software Testing Methodologies, Software Metrics, Data Warehouse – EAI, Data Mining, WBL, and XML. He is a recipient of Best Teacher Award at National Institute of Technology, Tiruchirapalli, during 2006-2007. He is a Life Member (LM) of **ISTE**, New Delhi, Life Member (LM), Computer Society of India.