SEMGD: SCORING ENGINE FOR MULTIPLAYER GAME DESIGN

B.M. Monjurul Alom, Nafisa Awwal

Assessment Research Centre, Melbourne Graduate School of Education, The University of Melbourne, Australia

ABSTRACT

Online multiplayer educational games can be designed to support collaboration and assess different cognitive and social abilities among students. The educational games can capture student responses or actions, both shared and unshared, within the game environment and from extrinsic resources. This paper describes the design and integration method of scoring engine to process the log stream data generated by multiplayer games, which have been developed as part of the ATC21S™ research study by the University of Melbourne. Scoring engine, which sits at the heart of the multiplayer gaming architecture, consists of a significant number of scoring algorithms to process the log data generated by the students' responses when they participated in the games. Scoring algorithms are coded to search the log files for particular data event sequences. Each of these algorithms take process stream data (produced by the events of the participants in different tasks) as input and produces relevant output defined by the rule and methods for the corresponding algorithm. The scoring process ensures the reporting engine feeds out individual student results for teachers to use in their classroom teaching. The multiplayer component of the games a remaintained with the use of the AJAX and Web Socket application which allowed the communication protocol between the client and server to be established. HTML5 has been used in preference to other available technologies in creating the games to provide a consistent experience for students across all browsers, platforms, and devices. In addition, canvas has been used to create all animations and game objects.

KEYWORDS

Log data, scoring, HTML5, web socket, game design, collaborative, game, 21st century.

1. INTRODUCTION

The online educational games presented in this paper, were developed to measure student's collaborative problem solving (CPS) skills. The use of online games in education in recent years are quite revolutionary, specifically with the incorporation of log file technology and learning analytics. These additions have provided a method from which richer data can be collected about student behaviours and used to measure their learning. Students can be assessed in real-time in an automated technique which reduces the time and effort that would be crucial for teachers to observe student performance in the classroom. In this collaborative setting, minimally a pair of students is essential to work together in order to effectively advance through each game. Similarly, various data events (e.g., local and global) are generated from all the games(aka tasks in ATC21S study) that are stored in log files for scoring student responses captured as stream data. During game play every action (e.g., selection, movement and clicks) and communication completed by each student is captured in log file in a time linear structure. It is important that every event is captured in the database as each action and communication is used to interpret the students' performance and experience in the game. Each game is presented similarly, with the content and context varying.

The log stream captures data of student progress in a game, from which skills or behaviours can be predicted within and across these games. Therefore, in order to capture behaviours of the students playing the games, scoring engine containing defined algorithms extract information from log files for inference of such behaviours. The algorithms were established to capture indicative behaviours regardless of the role and viewpoint that students were offered with each game. The games developed during ATC21S study focus on collaborative problem solving; the ideas for some of which are an extension of the games formerly developed by (Zoanetti 2010) for single student use in computer based problem solving. The growth of online games for learning has advanced in recent years. Researchers (Marchetti and Valente 2014)address the necessity for digital games to be altered to permit students to play and express themselves during their learning. The development of games has been an emphasis of study for a number of educational researchers. They have inspected how numerous features of game design might be appropriated, borrowed and repurposed for the design of educational materials and learning (Chiu, Wu et al. 2000, Kurt, Henry et al. 2003, Prensky 2007).Multiplayer game design with integrated technology are described more in details by (Alom, Awwal et al. 2015).

Researchers have emphasised on the challenge for educational game design which is to effectively balance the contribution and engagement from participants required with in a collaborative environment(Manninen 2002).Generally, there is a consensus on commonly agreed 21st century skills. These include problem solving, critical thinking, collaboration, information technology literacy and creativity(Griffin, McGaw et al. 2012, Care and Griffin 2014, Griffin and Care 2015). The tasks in ATC21S are characterised with multiple social and cognitive skills under the Collaborative Problem Solving concept(Care and Griffin 2014, Hesse, Care et al. 2015). This paper discusses the design of the scoring engine that is used to process the collected data and convert them into meaningful codes that can be used for analysis in reference to the CPS framework.

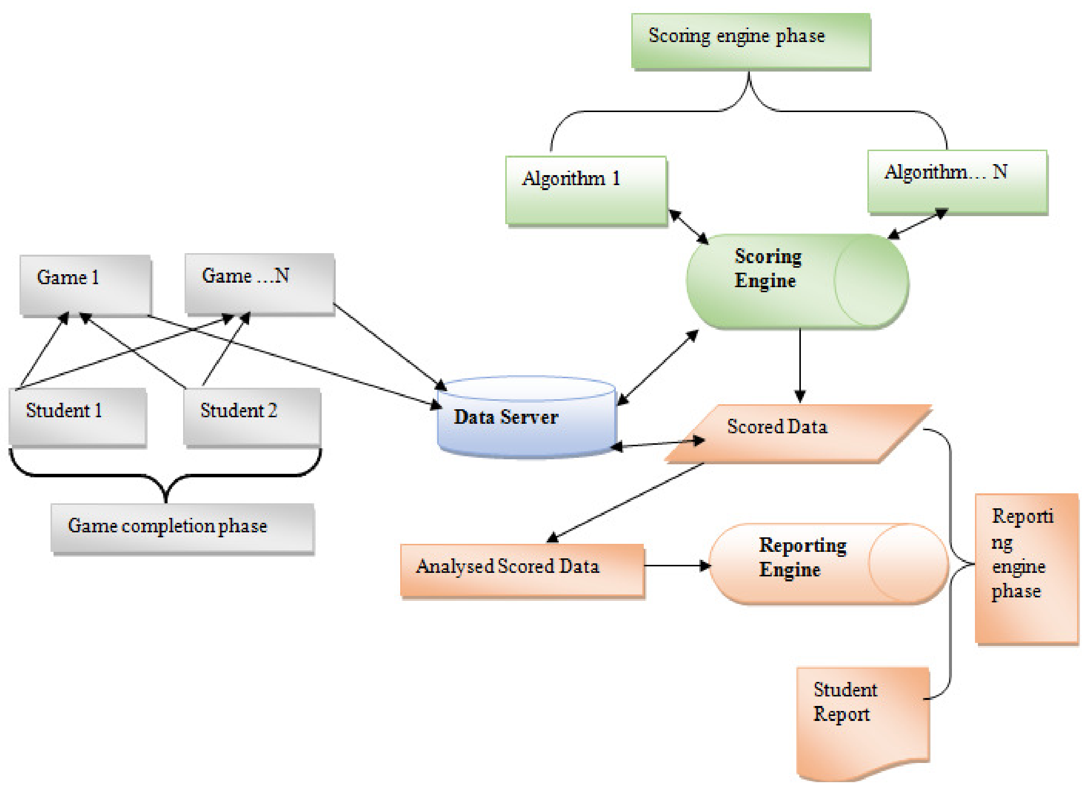

Figure 1: The overall system structure of MGA (adapted from Awwal, Griffin et al. 2015)

2. MULTI PLAYER GAME ARCHITECTURE

The overall system structure of the multiplayer game architecture (MGA) is presented in Figure 1.MGA consists of several phases, as seen in Figure 1,includinggame interface, game engine, scoring engine, and the reporting engine. At the completion of each game, log stream data gets stored into data server. The log files are then processed through the scoring engine to code the log stream data and to produce scored data from log files. These scored data then become the basis for the reporting engine to be considered as student responses to generate reports on students' performance.

In the ATC21Sstudy, the games are used in educational settings, most commonly in schools, where internet connections can be unsteady and present problems while working in an online or computer-based settings. To illustrate the structure, the collaborative views of one of the task *Balance Beam* is presented in Figure 2 and Figure 3 for each role in A and B. Students start off on a task by selecting a particular task presented in their respective dashboard and then by selecting on a role that they will play in the task. On completion of one of the task, students are forwarded to the task dashboard with the option to choose from other tasks that have been made available to them by their teachers. During a collaborative session, if one student from the pair comes across a technical problem or closes their window (or hits refresh/back button on their browser) by mistake, the gaming engine underlying the tasks are structured to permit both users to re-enter into the page on which they were last collaborating. Both students can move together, through the pages of the tasks in a linear way. The game engine supports students' engagement with the overall task by avoiding frustration on repeatedly doing the same thing. This decreases the chance of contamination in student responses from being frustrated due to technical issues.

The multiplayer architecture of the game is hosted on a remote server and includes several components including Linux, Apache HTTP Server, MySQL and PHP (Awwal, Griffin et al. 2015). The database is presented as a relational structure and the application packages are configured to support the various target languages and outputs the resultant game view at the client's end (Awwal *et al.* 2015). The scoring and reporting engine are based on the MySQL database, with PHP supporton a server that allows the PHP to be called externally.

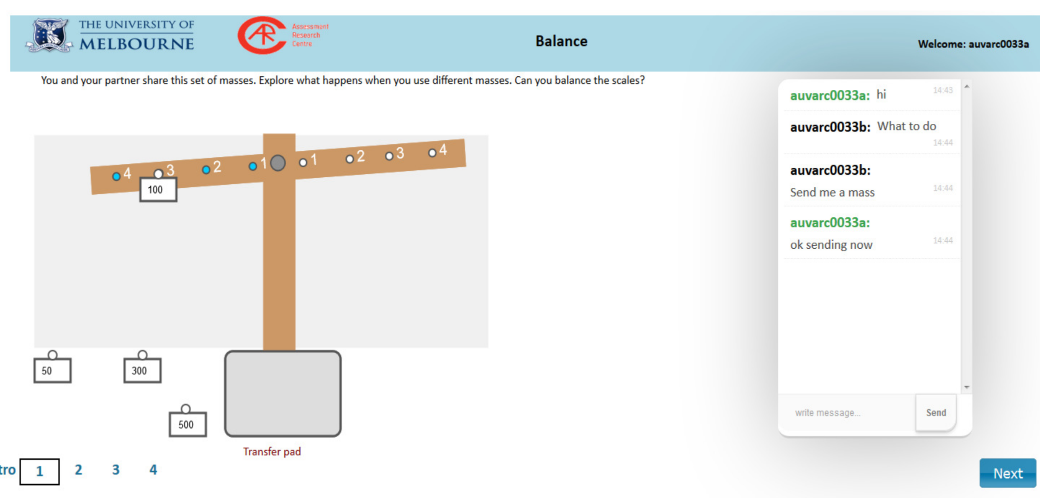

Figure 2: The Balance Beam with player A view(adapted from (Awwal, Griffin et al. 2015))

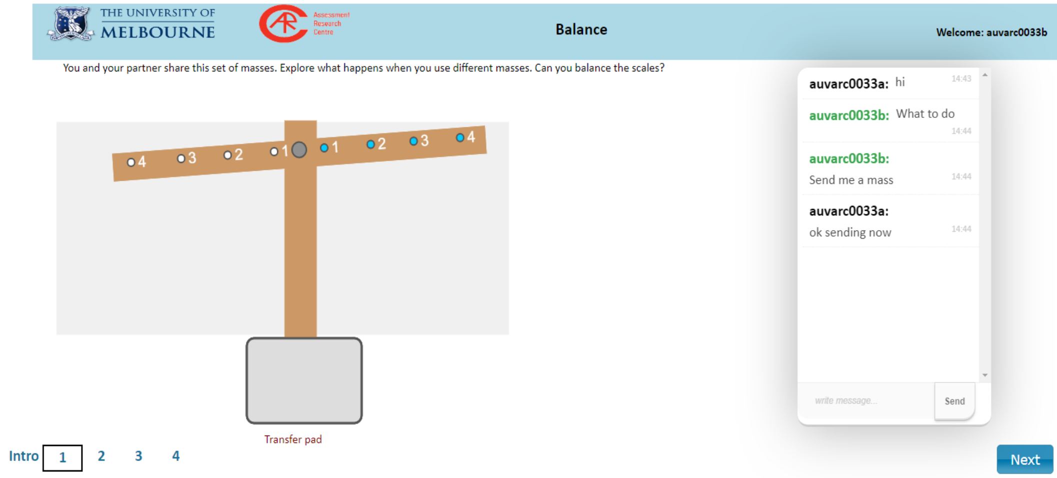

Figure 3: The Balance Beam with player B view(adapted from (Awwal, Griffin et al. 2015))

2.1 Log File Structure

Log files provide innovative ways to capture, store and analyse what students do and say. It can be utilised to interpret student learning through understanding of student behaviour. Log files are usually generated from student engagement with games to identify actions, interactions and communications between students while solving problems, including their lack of action. The log files also keep track of time and when actions were taken. A feature of the log files designed in ATC21S study was that they contained local and global events across games and students(Adams, Vista et al. 2015). For example, chat event is common for all the games and thus represents an example of global event. This allows for easier design, interpretation and analysis of the log files. The log files, capture data steadily so that behaviours can be readily recognized within and across games. Log file analysis can take the form of measurable or qualitative analysis and can be conducted either manually or automatically, this study adopting multiple approaches. The log files are used by the scoring engine to search for patterns of activity or behaviours for further analysis and interpretation. A log file of the multiplayer game is presented on Table 1 to illustrate as an example of its structure.

2.2 Database Structure

Data is collected from the games through the capture of keystrokes and mouse events such as typing, clicking, dragging, cursor movements, hovering time, and action sequences and so forth. This data is then recorded in the database in a relational structure. Each row of the database represents information about the student activity. Each student's response is recorded separately as an instance with corresponding user identification, present state, timestamp, record index and other data as considered necessary by researchers for the task analysis. The database consists of several columns labelled as record, actor_pid, team_id, task_id, page, player_id, event, data, bundle_id, timestamp(Awwal, Griffin et al. 2015). Record represents the number of the lines sequentially in the log file. Team_id presents the id that links both paired students together, this is essentially the same information attached to both students. Actor_pid allows for identification of a particular student associated with a record. For ease of identification, both students in a pair will have the same number sequence, with a different letter after the common number to differentiate between the two students. Bundle_id represents the assessment bundle being taken. Task_id allows for identification of which task the students played and in which events took place. Player_id helps differentiate between the two students and is represented by a letter which often is the same letter at the end of their student ID (A or B). Page_id refers to the page in which the event took place within. Event refers to the data events occurring. These usually represent the resources and information within eachtask that are being used. These may vary across tasks for different tasks, as different resources and information are utilised across tasks. Data refers to the variables or arguments that are captured within each of the events. This column represents more detailed information regarding a particular event. Timestamp represents the time and date of the event when it occurred and is captured in milliseconds.

2.3 Event Design

The data events capture all actions between student and the game environment, and also between student pairs. In addition, it includes all text entered into the chat box by each student. During game play every action (selection, movement and clicks) and communication completed by each student is captured in the log file in a time linear structure. It is important that every event is captured in the database as each action and communication. It is also good to capture and store actions that are may be assumed to be ineffective for collaborative problem solving, can later be used to interpret not only the students' performance but other experiences in the game. Within log files, multiple actions may occur concurrently, so this needs to be clearly distinguished from other activities in terms of the order of their occurrence as well as a type of the activities. In the example of *Balance Beam* task, the events designed for the tasks are labelled as Session start, Move weight, Place weight, Remove weight, Pass weight, chat, Build rule, confirm Rule, Select choice, Solution ready, Enter text, type message, Send message, Review, finished, Task end(Care, Griffin et al. 2015).

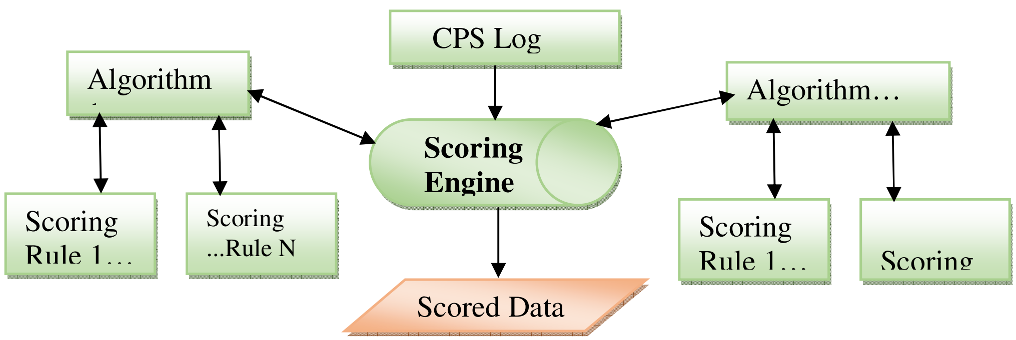

Figure 4: Scoring engine to process log stream data

3. Scoring Engine Design

3.1 Scoring Engine Structure

The scoring engine sits at the core of the multiplayer gaming architecture as it connects game engine, data servers and reporting engine. Scoring engine receives the log stream data as input and produces scored data as the final output. It searches through the data server, gets relevant log stream data, processes them into codes and then produces scored data based on defined algorithms within the scoring engine. Scoring engine is based on scoring algorithms. These algorithms are classified mainly into two types: *specific* and *global*. Each scoring algorithm is defined with multiple rules to find patterns in the collected data to understand student behaviour. In ATC21S, the specific algorithms are distinct algorithms for each of the tasks, whereas global algorithms are defined using a have common structure for all the tasks. Therefore, any global algorithm is applicable across all the tasks and the task specific ones are dissimilar for different tasks. each of the scoring algorithms are defined to identify the behaviour patterns by using different actions and chats (for example, action-chat-action, chat-action-chat and so on). Table 1 presents a sample input of log file for scoring engine, and a sample scored output from the scoring engine is presented in Table 2.The system structure of the scoring engine is presented in Figure 4.

Table 1: Sample log data for multiple CPS tasks

record	actor_pid	team_id	task_id	page	player_id	event	data	bundle_id	timestamp
165036	arcdemo1016	arc1008	100	0		start	Task started is 100	2	6/08/2014 10:22
165037	arcdemo1015	arc1008	100	0		start	Task started is 100	2	6/08/2014 10:23
165038	arcdemo1016	arc1008	100	0		ViewPage	0	2	6/08/2014 10:23
165039	arcdemo1015	arc1008	100	0		ViewPage	0	2	6/08/2014 10:23
165040	arcdemo1015	arc1008	100	0	A	Page request	User <A> requests page <1>	2	6/08/2014 10:23
165041	arcdemo1016	arc1008	100	0	B	Page request	User <B> requests page <1>	2	6/08/2014 10:23
165046	arcdemo1015	arc1008	100	1	A	action	3L_volume_at_drop 0:3L=0:5L=0	2	6/08/2014 10:23
165047	arcdemo1015	arc1008	100	1	A	action	3L_fill:3L=3:5L=0	2	6/08/2014 10:23
165048	arcdemo1015	arc1008	100	1	A	action	3L_volume_at_drop 3:3L=3:5L=0	2	6/08/2014 10:23
165049	arcdemo1016	arc1008	100	1	B	action	completedTransfer:3L=0:5L=3	2	6/08/2014 10:23
68294	arcdemo1015	arc1008	ICP0900166	4	A	Move weight	50,0->4,[200,0]	1	6/08/2014 10:51
68295	arcdemo1015	arc1008	ICP0900166	4	A	Place weight	50,4,[200,0]	1	6/08/2014 10:51
68296	arcdemo1015	arc1008	ICP0900166	4	A	Move weight	50,0->3,[350,0]	1	6/08/2014 10:51
68297	arcdemo1015	arc1008	ICP0900166	4	A	Place weight	50,3,[350,0]	1	6/08/2014 10:51
69695	arcdemo1002	arc1001	ICP0900166	1	B	Type message	i put 300g on 1	1	6/08/2014 10:51
69696	arcdemo1001	arc1001	ICP0900166	1	A	Type message	yes, I send a 300g miss to you	1	6/08/2014 10:51
68863	arcdemo1050	arc1025	ICP0900166	4	A	Task end	Player A pressed the Finished button	1	6/08/2014 10:51
68864	arcdemo1049	arc1025	ICP0900166	4	B	Task end	Player B pressed the Finished button	1	6/08/2014 10:51

Table 2: Scored data for Balance Beam task

teamID	studentID	player	W8L101A	W8L110A	W8L314A	W8L106B	W8L110B	W8L318B	W8L320B	W8L319B	W8L401B	W8L426B
coalir0033	coalir0033b	B	-1	1	-1	-1	1	1	-1	1	-1	1
coalir0033	coalir0033a	A	0	1	1	-1	-1	1	-1	1	-1	1
coalir0032	coalir0032b	B	-1	0	-1	0	1	1	-1	1	-1	0
coalir0032	coalir0032a	A	1	1	-1	-1	-1	1	-1	1	-1	1
coalir0031	coalir0031b	B	-1	0	-1	0	1	1	-1	-1	-1	1
coalir0031	coalir0031a	A	1	1	1	-1	-1	-1	-1	-1	-1	1
coalir0028	coalir0028b	B	-1	0	-1	-1	1	-1	-1	-1	0	0
coalir0028	coalir0028a	A	0	1	0	-1	-1	-1	-1	0	0	0
coalir0027	coalir0027b	B	-1	0	0	0	1	-1	-1	0	0	0
coalir0027	coalir0027a	A	1	1	0	-1	-1	-1	-1	0	1	0
coalir0026	coalir0026b	B	-1	0	0	-1	0	-1	-1	1	1	0
coalir0026	coalir0026a	A	0	0	1	-1	-1	-1	-1	1	0	1
coalir0025	coalir0025b	B	-1	-1	-1	-1	1	-1	-1	1	0	1
coalir0025	coalir0025a	A	1	1	0	-1	-1	0	-1	1	0	1
coalir0024	coalir0024b	B	-1	-1	-1	-1	1	0	-1	-1	0	1
coalir0024	coalir0024a	A	1	1	1	-1	-1	0	-1	0	1	1
coalir0023	coalir0023b	B	-1	-1	-1	1	1	1	-1	1	1	-1
coalir0023	coalir0023a	A	1	1	1	-1	-1	-1	-1	1	1	-1

3.2 Scoring Algorithms Structure

Sequences of data events as mentioned earlier (e.g. chat-action-chat, action-chat-action etc.), were identified and interpreted in relation to student learning of collaborative problem-solving skills. Scoring algorithms were used to search the log files for the data event sequences. The scoring engine work as an input-output flow. The patterns of events identified in the log stream data are considered as the input and a scored record file based upon the algorithms is considered the output. Each algorithm searches for the data event sequences for every student in each task they have played. The algorithms then code those event sequences and finally gives students score. If the algorithm finds the defined pattern then a student receives a '1',if it does not find that pattern then students get a '0' and there are some other defined rules for other occurrences. Each of the scoring algorithms takes process stream data (produced by the events of the participants in different games) as input and produces relevant output defined by the rule for the corresponding algorithm. Each algorithm in ATC21S is coded with a unique ID code to differentiate between each of the rules. For example, in the *Balance Beam* task one of the algorithm is named 'W8L101A', where 'W8' represents the *Balance Beam* task, 'L' indicates that it is a 'local' algorithm specific to that task(this would be replaced by 'G' to represent that it was a global algorithm that could be applied to all tasks), '004' is a numerical code specific to this algorithm which is provided for ease of referencing and is sequential within each task(in this case 004 was the fourth algorithm created for this task) and 'A' indicates that this algorithm is applicable to student A. In addition, if an algorithm is used to search and capture the quantity of interaction within a task, then the rule in the algorithm would be set to count the occurrences of the event 'chat' in the process stream. The coded output for this algorithm would be the numerical value representing the frequency of the chat. For more details on scoring and the rules of algorithm that are used to defined indicators within and across tasks, see details in Adams et al. 2015. Table 3 outlines some indicators from *Balance Beam* task that are derived using exemplar algorithms through the scoring engine. Each indicator represents a skill to indicate an observable behaviour identifiable during the play of the tasks. The algorithmic rules are used to recognise those patterns from the log stream dataset and used to extract those data patterns to define those indicative behaviours for each indicator. Some examples of those are presented in Table 3 below.

Table 3: Example of the algorithms (adapted from (Care, Griffin et al. 2015))

Skill	Behavior	An example of algorithmic rule for pattern recognition
Action	Active in scaffolded environments	Student A passes B a mass
Task Completion	Undertaking part of a task individually	Follow instructions, moves 100g to position 4
Responsiveness	Responding to contributions of others	Realises that some mases cannot balance. If student A resends 50 or 500, B returns it immediately
Sets Goals	Sets goals for a task	Requests mass amounts
Systematicity	Implements possible solutions to a problem	Trial of different combinations of masses on different beam positions
Solution	Correct answer	Number of successful balances achieved (3 optimum)

Table 4: Example of an algorithms in pseudo code

```
Algorithm 001
Step 1.    Begin
Step 2.    Retrieve logs for Page 1 for the game
Step 3.    Find the event Pass weight from a player
Step 4.    If Step 3 is found that record value as 1
Step 5.    Else record value as 0
Step 6.    If Step 4 and 5 is not found, then assign -1
Step 7.    End
```

The rules presented in Table 3 are then used to be translated into algorithm that uses the idea of data analytics to explore patterns in the data as per the rules and the defined algorithms converts them from data stream to codes for later analysis. This idea is illustrated with a pseudo algorithm example provided in Table 4.The algorithm reflects the skill, behaviour and rule defined and have been presented as the first row in Table 3. Algorithm 001 is defined such that it can extract information to represent indicators that represent appropriate exploration and understanding of the problem environment. The rule for this algorithm is to search and recognise patterns from the log stream files when player in role A passes on any mass to player in role B. This algorithm can be applied to log stream from any of the collaborative tasks, but in here the *Balance Beam* task is used as an example. In the machine language, this rule is translated as: Find an event that represents 'Pass weight' from a player. If an action associated with this event is found, then represent this pattern as a coded value of 1 to indicate pattern is present; else code as0 to indicate pattern is not found. If there is no such activities found in the logs from the player, then code it as-1; which would indicate missing value for this assessment. Possible output from using this algorithm are 0, 1, and -1.

4. FINDINGS

In ATC21S study, there has been a considerable number of scoring algorithms written in PHP language to define the various behavioural indicators. For the purposes of the study, there were around 400 algorithms developed to define the various indicators that were designed to define behavioural indicators. Amongst these 52 were global algorithms and the remaining totalled the task specific (around 340) algorithms. The database with log stream data files consisted of 2.35 million records. Different number of records captured for each of the task in the database are described in Table 5. Execution time for each task specific and global algorithm is also presented in column 5 and column 7 respectively.

Table 5: Scoring engine algorithms details

Total records in the DB	Task Name	Records in DB per task	Number of Specific alg.	Execution Time for task specific alg.	Number of Global alg.	Execution Time forglobal alg.
2.35 million	Balance	375725	40	58 seconds	52	54 minutes
	Hot Chocolate	336588	36	49 seconds		43 minutes
	Laughing Clown	346841	22	1 minute 37 seconds		38 minutes
	Plant Growth	302925	22	14 minutes 30 seconds		39 minutes
	Olive Oil	272312	27	40 seconds		40 minutes
	Hexagon	41680	39	11 seconds		7 minutes
	Small Pyramid	210381	67	36 seconds		48 minutes
	Game of 20	344160	59	51 seconds		56 minutes
	Shared Garden	33390	22	9 seconds		3 minutes

Execution time of the algorithm depends on mainly three things. Firstly, the number of records (i.e. the size of the log file) for each task in the database, as the algorithms needs to search through that amount of records to process the various defined rules within an algorithm structure. Secondly, the number of total algorithms and the difficulty or complexity of those algorithms, as more algorithm have more difficult rules which takes longer to process and the number of rules with each algorithm or algorithms within each task mean more processing time. And lastly, the type of algorithm (i.e. global or specific algorithm) that is being processed. The game specific algorithms usually are faster in processing data stream files than that of global algorithms. Most of the global algorithms are based on identifying patterns to determine the blocks of action-chat-action or chat-action-chat, which means that finding such pattern blocks takes more time to execute global algorithms than the task specific algorithms.

Table 6: Comparison of the execution time of the algorithm.

Total of records in DB	Number of cases	Difficulty level of alg.	Lines of code in alg.	Conditions and rules present	Execution time for alg.
2.35 million	4562	Low	7	2	25.5 sec
		Medium	54	3	41.3 sec
		High	230	6	2.24 minutes

To illustrate the above, Table 6 shows examples of three categories of algorithms that are contained in the scoring engine. They are categorised in terms of their difficulty levels based on the numbers of syntaxes and lines in each algorithm, the quantity of rules and conditions necessary to implement each algorithm, and the amount of time it takes to execute each of them. From Table 6 and Figure 5, it is apparent that more conditions and rules means more lines of code to define them, which in turn means longer period to run such algorithms.

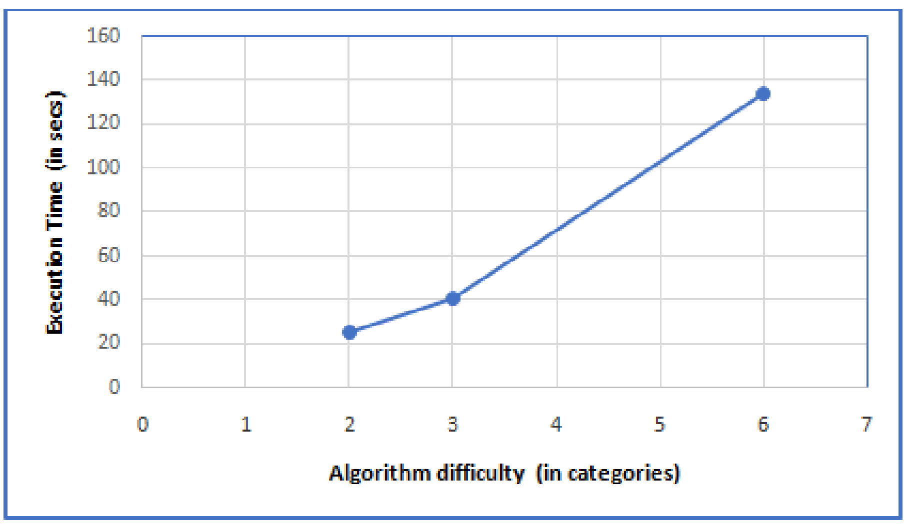

Figure 5: Relationship between execution time and difficulty levels of algorithms.

5. CONCLUSION

Designing multiplayer games for a collaborative virtual environment is a difficult effort. Adding the collaboration component on multiplayer is even more challenging to design when compared with games that are designed with single player functionality. However, new technologies such as HTML5 with Web Socket and other technologies that are used for synchronous communication, collaboration and gaming architecture have shown such development is possible while retaining consistent game flow and a positive user experience while users are engaged in such games, specifically if they are educational games(Awwal, Scoular et al. 2017).In this paper, we described the Scoring Engine (SEMGD) that were used to capture the observable behaviours from log files generated from multiplayer educational games during the research study in ATC21S and other research endeavours by the same team. The Scoring Engine is linked to multiple components of the game-like assessment architecture with the aim of reporting student results from the collaboration work to be presented to their teachers for use in the classroom. The algorithms are used to reflect the various definition of the behavioural indicators that are in turn used to make inferences about the performance of the students demonstrated during the Collaborative Problem Solving skills. The Scoring Engine comprises of the scoring algorithms which processes the captured information in the log files. The tasks were developed based on the principles of different features and event structure. These features of the game-like tasks allowed the search algorithm to be applied to each task in order to identify, code and extract information from log files. The Scoring Engine development during the ATC21S work has shown that such a mechanism is essential for evaluating students' complex skills sets. Further work is in progress by the research team to make such a scoring engine compatible across any gaming architecture and for evaluating any types games used for assessment of 21st century skills.

ACKNOWLEDGMENTS

The work presented here is from the research Assessment and Teaching of 21st Century Skills (ATC21S, www.atc21s.org) study (Griffin, McGaw et al. 2012, Griffin and Care 2015). The research was a joint funded effort between 2009 to 2012 from the industry partners Cisco, Intel, Microsoft; and from participating founder countries Australia, Singapore, USA, Finland; participating associate countries Netherlands, Costa Rica; and managed by the University of Melbourne. Since July 2012 the project has also been funded internally using Assessment Research Centre, The University of Melbourne funds. Acknowledgment for the work presented here is contributed to all the researchers of this project and the research team working in 21st century skills projects of the Assessment Research Centre at The University of Melbourne.

REFERENCES

[1] Adams, R., A. Vista, C. Scoular, N. Awwal, P. Griffin and E. Care (2015). Automatic Coding Procedures for Collaborative Problem Solving. Assessment and teaching of 21st century skills: Methods & approach. D. Springer, P. Griffin, E. Care. 2: 115-132.

[2] Alom, B. M., N. Awwal and C. Scoular (2015). Technology Integration in Multiplayer Game Design. European Conference on Games Based Learning Norway: 10-14.

[3] Awwal, N., P. Griffin and S. Scalise (2015). Platforms for delivery of collaborative tasks. Assessment and teaching of 21st century skills: Methods & approach. D. Springer, P. Griffin, E. Care. 2: 105-113.

[4] Awwal, N., C. Scoular and B. M. M. Alom (2017). An Automated System For Evaluating 21st Century Skills Using Game-Based Assessments. Education and New Learning Technologies. Barcelona: 1593-1598.

[5] Care, E. and P. Griffin (2014). "An approach to assessment of collaborative problem solving." Research & Practice in Technology Enhanced Learning, 9(3): 367-388.

[6] Care, E., P. Griffin, C. Scoular, N. Awwal and N. Zoanetti (2015). Collaborative Problem Solving Tasks. Assessment and Teaching of 21st Century Skills : Methods & Approach. D. Springer, P. Griffin, E. Care. 2: 85-104.

[7] Chiu, C.-H., W.-S. Wu and C.-C. Huang (2000). Collaborative Concept Mapping Processes Mediated by Computer. WebNet World Conference on the WWW and Internet 2000. San Antonio, Texas, Association for the Advancement of Computing in Education (AACE): 95-100.

[8] Griffin, P. and E. Care (2015). Assessment and teaching of 21st century skills : methods and approach, Dordrecht : Springer.

[9] Griffin, P., B. McGaw and E. Care (2012). Assessment and Teaching of 21st Century Skills, Springer Netherlands.

[10] Hesse, F., E. Care, J. Buder, K. Sassenberg and P. Griffin (2015). A Framework for Teachable Collaborative Problem Solving Skills. Assessment and Teaching of 21st Century Skills: Methods & Approach. D. Springer, P. Griffin, E. Care. 2: 37-56.

[11] Kurt, S., J. Henry, H. Walte, M. Heathe, O. D. Alice, T. K. Philip and T. Katie (2003). "Design Principles of Next-Generation Digital Gaming for Education." Educational Technology 43(5): 17-23.

[12] Manninen, T. (2002). Towards Communicative, Collaborative and Constructive Multi-player Games. Computer Games and Digital Cultures Conference. Tampere, Finland: 155-169.

[13] Marchetti, E. and A. Valente (2014). Design Games to Learn: A New Approach to Playful Learning Through Digital Games. European Conference on Games Based Learning. Berlin,Germany: 356-363.

[14] Prensky, M. (2007). Digital game-based learning, Paragon House; Paragon House Ed edition.

[15] Zoanetti, N. (2010). "Interactive computer based assessment tasks: How problem-solving process data can inform instruction." Australasian Journal of Educational Technology 26(5): 585-606.