Chapter 6
Automatic Coding Procedures for Collaborative Problem Solving

Raymond Adams, Alvin Vista, Claire Scoular, Nafisa Awwal, Patrick Griffin, and Esther Care

Abstract This chapter examines the procedure followed in defining a scoring process to enable the reporting of individual student results for teachers to use in the classroom. The procedure begins with the identification of task features that match elements of the skills frameworks, and is followed by the generation of simple rules to collect data points to represent these elements. The data points are extracted from log files generated by students engaged in the assessment tasks and consist of the documentation of each event, chat and action from each student. The chapter includes examples of the process for defining and generating global and local (task specific) indicators, and examples of how the indicators are coded, scored and interpreted.

The development of coding and scoring of data generated when students engage in collaborative problem solving tasks is described. The data generated are captured in a process stream data file. Patterns of these data are coded as indicators of elements defined in the conceptual framework outlined in Hesse et al. (2015; Chap. 2) and the relative complexity of indicators is used in a scoring process. The scored data are then used to calibrate the tasks. The calibrations form the basis of interpretation and these are used in forming reports for students and teachers. Figure 6.1 summarises the entire process from task development to the reporting of student ability based on a developmental framework.

R. Adams (✉) • A. Vista • C. Scoular • N. Awwal • P. Griffin • E. Care
Assessment Research Centre, Melbourne Graduate School of Education, University of Melbourne, Parkville, VIC, Australia
e-mail: r.adams@unimelb.edu.au

© Springer Science+Business Media Dordrecht 2015
P. Griffin, E. Care (eds.), *Assessment and Teaching of 21st Century Skills*, Educational Assessment in an Information Age,
DOI 10.1007/978-94-017-9395-7_6

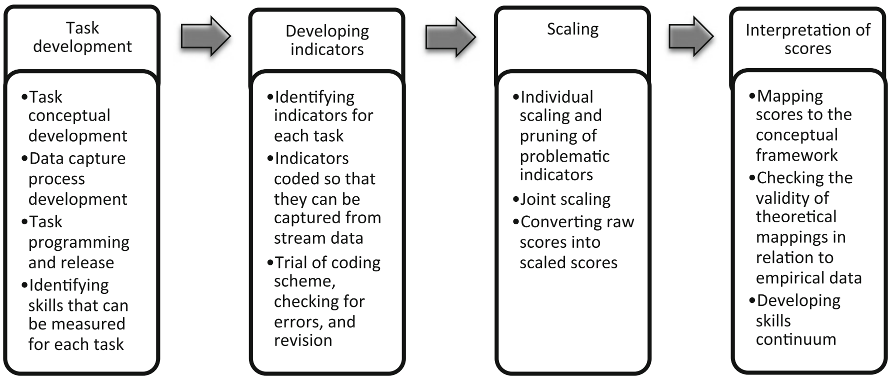

Task development
•Task conceptual development
•Data capture process development
•Task programming and release
•Identifying skills that can be measured for each task
Developing indicators
•Identifying indicators for each task
•Indicators coded so that they can be captured from stream data
•Trial of coding scheme, checking for errors, and revision
Scaling
•Individual scaling and pruning of problematic indicators
•Joint scaling
•Converting raw scores into scaled scores
Interpretation of scores
•Mapping scores to the conceptual framework
•Checking the validity of theoretical mappings in relation to empirical data
•Developing skills continuum

Fig. 6.1 Process overview from task development to interpretation of scores

Existing Approaches to Autoscoring

There is currently very little research regarding a scoring approach for collaborative problem solving. Our project has therefore focused on and adapted existing scoring processes for problem solving. The literature suggests that current processes mainly use a dichotomous success-failure scoring system which records whether the problem has been solved and ignores the cognitive procedures involved (Greiff et al. 2012). This type of system is simple to implement and works well for tasks which are designed to tap into specific problem solving skills. For example, a task where deductive reasoning is imperative for success can be scored dichotomously. An example of this style of dichotomous scoring can be observed in a project by Greiff et al. (2012) who have determined three measures which represent dynamic problem solving (DPS): Model Building, Forecasting and Information Retrieval. Each of these measures is scored across 11 DPS tasks and students are awarded a false (0) or true (1) score determined by their success or failure on the task. In contrast, the focus in the ATC21S™ project[1] is not only to determine whether students are succeeding at solving the tasks but to draw inferences about *how* students solve problems. While the assumption in traditional test design is that the attainment of the solution is the sole criterion, here the focus is on the process and quality of problem solving. A distinction needs to be made between what might be called simple problem solving tasks, using a dichotomous scoring process, dynamic problem solving, using a series of dichotomous scores, and complex problem solving, using rubrics and partial credit approaches.

The procedural aspects of problem solving (PS) have been considered important for some time (Polya 1945, 1957; Schoenfeld 1985). The framework proposed in the ATC21S project outlines five broad components that represent collaborative

[1] The acronym ATC21S™ has been globally trademarked. For purposes of simplicity the acronym is presented throughout this chapter as ATC21S.

problem solving (CPS) within social skills (participation, perspective taking, social regulation) and cognitive skills (task regulation and knowledge building). Within these five components, students are assessed on three tiered levels of ability across 19 specific elements. A set of assessment tasks, each tapping into different and overlapping skills, was developed in order to provide teachers with sufficient information to interpret students' capacity in CPS subskills, so that a profile of each student's performance can be developed for formative instructional purposes. In order to provide this interpretive feedback, there was a need to develop a robust automated scoring system which highlighted the procedural and developmental thinking processes that take place in CPS.

Design of Process Data Stream – Capturing and Identifying the Data

Many recent computer based PS tasks have been able to assess and record detailed interactions between the problem solver and the task environment, and thereby capture salient solution processes in an unobtrusive way (Zoanetti 2010; Bennett et al. 2003). Their recorded input can be linked to the cognitive skill level and development of students and used to evaluate the process and efficiency with which problem solvers complete tasks (Pelligrino et al. 2001; Williamson et al. 2006). Within the current CPS framework, actions and chat, and the placement of these, can be scored.

In order to record descriptive, purposeful actions, the CPS assessment tasks were designed to capture detailed interactions between problem solvers working as a dyad as well as between the individual problem solver and the task. In the context of computer based assessments, the files generated for the automatic records of these types of student–task interactions are referred to as a 'session log file.' They contain free-form data referred to as 'process stream data.' The log files were stored as free-form text files with delimited strings of text or in database architecture. In this instance, MySQL database architecture was used for recording the interactions with the task environment thereby describing relevant solution processes in an unobtrusive way (Bennett et al. 2003).

In the context of these assessment tasks, process stream data describe distinct key strokes and mouse events such as typing, clicking, dragging, cursor movements, hovering time, action sequences and so on. In the database, each discrete action is recorded with a corresponding timestamp. A timestamp refers to the time at which an event was recorded by the system into a log file and reflects, or is close to, the time of the event itself. To ensure that the data captured can be presented in a consistent format, allowing relatively easy comparison of two different records and tracking progress over time, a sequential numbering of events is used in a consistent manner. In this way, timestamps enable detailed analysis of action sequences and inactivity. This ensures further transparency for data storage and the sequential logging of events for data processing. These forms of time-stamped data have been

Table 6.1 Examples of common events defined from the process stream data

Event type	Process stream data format	Explanation of data captured
Session start	Student ***student_id*** has commenced task ***task_id***	Records the start of a task with student and task unique identification
Session finish	Student ***student_id*** has completed task ***task_id***	Records the end of a task with student and task unique identification
Chat text	Message: "*free form of message using the chat box*"	Captures the contents of the chat message the students used to communicate with their partner
Ready To progress	Requested to move to page: ***page_id***	Indicates whether the student is ready to progress or not, and records the navigation endpoint which they are ready to progress to for multipage tasks
Other click	Screen x coords: ***x_coordinate***; Screen y coords: ***y_coordinate***;	Captures the coordinates of the task screen if the student has clicked anywhere outside the domain of the problem

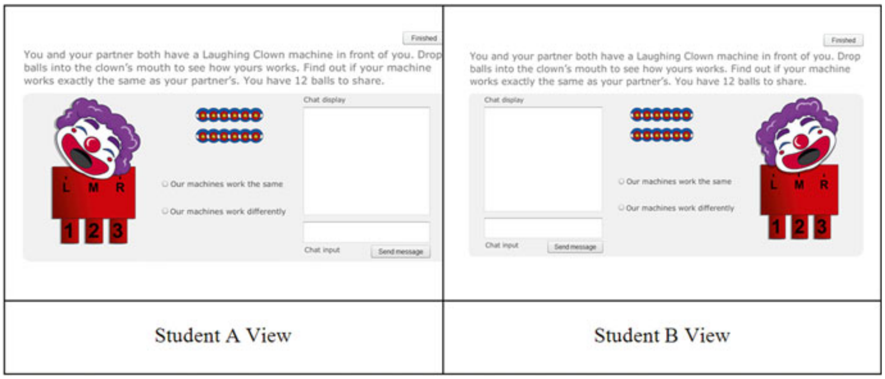
referred to variously as "log-file data" (Arroyo and Woolf 2005), "discrete action protocols" (Fu 2001), "click-stream data" (Chung et al. 2002) and "process data" (Zoanetti 2010). For our purpose, we use the term 'process stream.'

Each task has a variety of events that can occur, categorised into two types: common and unique events. As the name suggests, 'common' classifies the universal nature of the process stream events and applies to all collaborative assessment tasks. Examples of these events can been seen in Table 6.1. They include indications of the beginning and end of a task, system confirmation messages of user actions, navigational system messages for multiple page assessment tasks, free-form chat messages for communication with partners, or variations of these.

Unique events within the process stream data are not common across assessment tasks. They are unique to specific tasks due to the nature of the behaviours and interactions those tasks elicit. These data are defined using event types to match specific requirements that may arise only in a particular interactive problem space. Examples of such events for the Laughing Clowns task, illustrated in Fig. 6.2, are presented in Table 6.2 (for a detailed explanation of this task see Care et al. 2015; Chap. 4).

The accumulation of the different types of process and click stream data collectively forms the process data stream, accumulated and stored in files commonly referred to as *session logs*. An excerpt of a session log for the Laughing Clowns task can be seen in Fig. 6.3, which represents the events that occurred for one team (two students) while playing the task. Both common and unique event types of process stream data were captured in string format as shown in Tables 6.1 and 6.2. Process stream string data were recorded in the MySQL database as a single row and tagged with corresponding student identifier, task identifier, page identifier and role allocation of the acting student in the collaborative session with time-stamping and appropriate indexing.

To facilitate collaboration, a chat box tool was used (see Care et al. 2015) as the messaging interface for communication between respective partners during the collaborative sessions. This enabled the students to explore and learn about their respective resources and share or report information to each other. The chat box tool


You and your partner both have a Laughing Clown machine in front of you. Drop balls into the clown's mouth to see how yours works. Find out if your machine works exactly the same as your partner's. You have 12 balls to share.

You and your partner both have a Laughing Clown machine in front of you. Drop balls into the clown's mouth to see how yours works. Find out if your machine works exactly the same as your partner's. You have 12 balls to share.

Fig. 6.2 Screenshots from the Laughing Clowns task

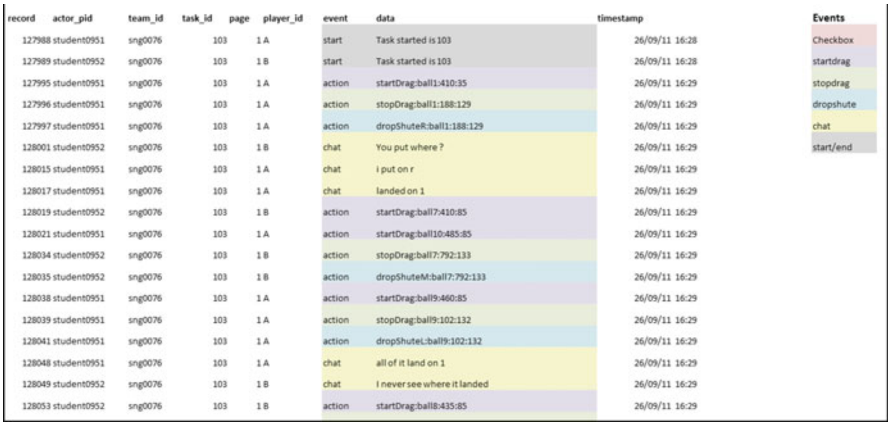
Table 6.2 Examples of unique events defined from the Laughing Clowns task within the process stream data

Event type	Process stream data format	Explanation of data captured
StartDrag	startDrag: *ball_id*; *x,y coordinates of the ball at the start of the drag*	Records the identifier of the ball which is being dragged by the student, and its coordinates
StopDrag	stopDrag: *ball_id*; *x,y coordinates of the ball at the end of the drag*	Records the identifier of the ball which is being dragged by the student and its coordinates at the end of the drag
DropShute	dropShute*PosofShuteId*: *ball_id*; *x,y coordinates of the ball when it was dropped*	Records the identifier of the ball, its coordinates and the value of the clown head shute when it was dropped by the student
Check box	Selection Value: *option_value*	Captures data if students agree or disagree on how their machines work

captures text exchanged between students and was captured in string data format. All chat messages generated by users and the system appeared in the tool and were recorded with a corresponding timestamp.

Defining the Indicators

Each task was examined for indicative behaviours of identifiable cognitive and social skills that can be captured algorithmically. These skills were identified through actions, chats or a combination within the process stream. The behaviours that were observed in the process stream data were used as indicators of



record	actor_pid	team_id	task_id	page	player_id	event	data	timestamp
127988	student0951	sng0076	103	1	A	start	Task started is 103	26/09/11 16:28
127989	student0952	sng0076	103	1	B	start	Task started is 103	26/09/11 16:28
127995	student0951	sng0076	103	1	A	action	startDrag:ball1:410:35	26/09/11 16:29
127996	student0951	sng0076	103	1	A	action	stopDrag:ball1:188:129	26/09/11 16:29
127997	student0951	sng0076	103	1	A	action	dropShuteR:ball1:188:129	26/09/11 16:29
128001	student0952	sng0076	103	1	B	chat	You put where ?	26/09/11 16:29
128015	student0951	sng0076	103	1	A	chat	i put on r	26/09/11 16:29
128017	student0951	sng0076	103	1	A	chat	landed on 1	26/09/11 16:29
128019	student0952	sng0076	103	1	B	action	startDrag:ball7:410:85	26/09/11 16:29
128021	student0951	sng0076	103	1	A	action	startDrag:ball10:485:85	26/09/11 16:29
128034	student0952	sng0076	103	1	B	action	stopDrag:ball7:792:133	26/09/11 16:29
128035	student0952	sng0076	103	1	B	action	dropShuteM:ball7:792:133	26/09/11 16:29
128038	student0951	sng0076	103	1	A	action	startDrag:ball9:460:85	26/09/11 16:29
128039	student0951	sng0076	103	1	A	action	stopDrag:ball9:102:132	26/09/11 16:29
128041	student0951	sng0076	103	1	A	action	dropShuteL:ball9:102:132	26/09/11 16:29
128048	student0951	sng0076	103	1	A	chat	all of it land on 1	26/09/11 16:29
128049	student0952	sng0076	103	1	B	chat	I never see where it landed	26/09/11 16:29
128053	student0952	sng0076	103	1	B	action	startDrag:ball8:435:85	26/09/11 16:29

Events
Checkbox
startdrag
stopdrag
dropshute
chat
start/end

Fig. 6.3 Excerpt from a process stream log file for the Laughing Clowns task

cognitive and social skills as defined in Hesse et al. (2015). These indicative behaviours were then coded into rule-based indicators that can be extracted from the task process streams through an automated algorithmic process similar to that described by Zoanetti (2010). Zoanetti showed how process data (e.g., counts of actions) could be interpreted as an indicator of a behavioural variable (e.g., error avoidance or learning from a mistake) (see Table 3 in Zoanetti 2010). For example, in the Laughing Clowns task, a count of the 'dropShute' actions (dropping the balls into the clown's mouth) can indicate how well the student managed their resources (the balls).

Coding

The coded indicators became the primary source of data for the scoring process. The indicators were classified into two main types: those that occur only in specific tasks and those that can be observed in all tasks. Indicators that can be captured in all tasks are labelled 'global'. They included total response time, response time to partner questions, action counts, and other behaviours that were observed regardless of the task. Indicators that were task-specific were labelled 'local'. There were two categories of local indicators: direct and inferred. Direct indicators represented those that can be identified clearly, such as a student performing a particular action. Inferred indicators related to such things as sequences of action/chat within the data. Patterns of indicators were used to infer the presence of behaviour which is indicative of elements in the conceptual framework (see Hesse et al. 2015; Chap. 2). Within these indicators there were differences in intensity or patterns that provided additional information about the relative complexity of the indicated behaviour.

Each indicator was coded with a unique ID code. Using the example of the unique ID code 'U2L004A', 'U2' represents the Laughing Clowns task, 'L' indicates that it is a 'local' indicator specific to that task ('G' would represent that it was a global indicator that could be applied to all tasks), '004' is a numerical code specific to this indicator which is provided for ease of referencing and is sequential within each task (in this case 004 it was the fourth indicator created for the task) and 'A' indicates that this indicator is applicable to student A.

To capture the required data, once the indicators are identified they need to be defined in the process stream through programming algorithms. Each of the scoring algorithms takes process stream data (produced by the events of the participants in different tasks) as input and produces relevant output defined by the rule for the corresponding indicator. For example, if capturing the quantity of interaction within a task, the algorithm would count the occurrences of the event 'chat' in the process stream. The output for this indicator would be the numerical value representing the frequency of the chat. Table 6.3 outlines some exemplar algorithms. The first column in the table represents the indicator name. Details of the scoring rule for each indicator are described in column two. The third and fourth columns elaborate the algorithm and its output respectively.

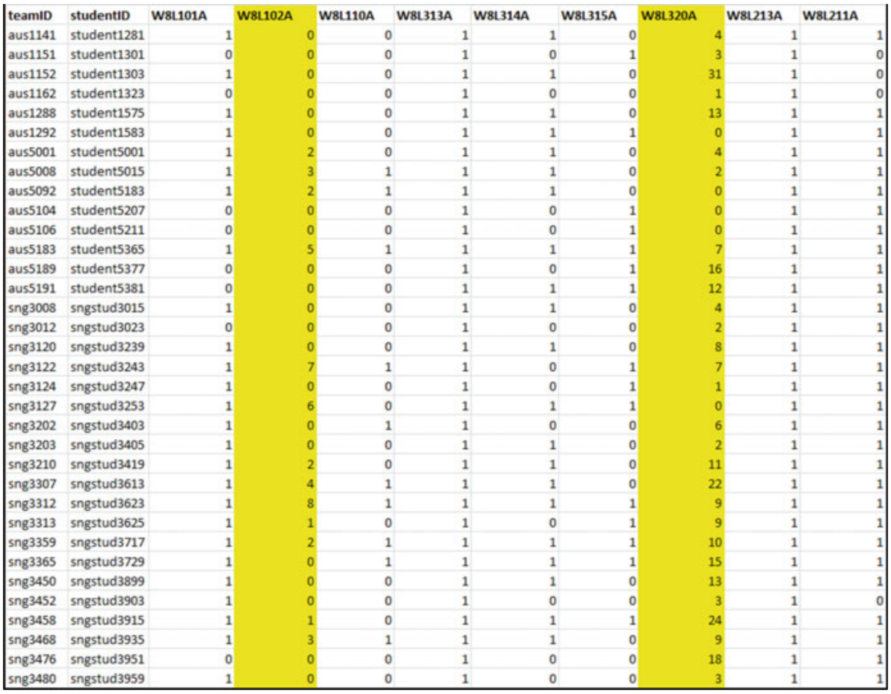
The outputs from each of the indicators based on the algorithms are saved in a 'coded file'. The coded file presents the output values relevant to the algorithm. For example, if the indicator observes a count of actions, the raw numerical value will be present in this file. Indicators highlighted in yellow in Fig. 6.4 are still in raw counts (or frequencies). These indicators are later converted into either a dichotomy or partial credit.

Mapping

Each indicator was mapped onto an element of the conceptual framework (outlined in Hesse et al. 2015; Chap. 2). which consists of five strands – three comprising the social aspect and two comprising the cognitive aspect. The main purpose of this mapping process was to identify an underlying skill. To reduce judgment error in the mapping process, it was undertaken several times by different teams. An iterative process was used. Several panels of researchers reviewed the indicators and mapped them onto the conceptual framework. The process was repeated for each set of indicators within each task until a stable allocation was agreed upon. When the changes and revisions to the allocation of indicators to elements fell to a minimum, the element mapping was then considered to be stable and the interpretation process proceeded to the next step. As an example, the indicator U2L004A records whether a student covers all positions with their balls. This is assessed by the presence of three 'dropShute' actions in the process stream for student A – one for each of the three positions L (left), M (middle), and R (right). This indicator was mapped onto systematicity in the framework, suggesting that the student had explored the task through a strategic sequence of actions. An excerpt from a session log on how this is captured can be seen in Fig. 6.5.

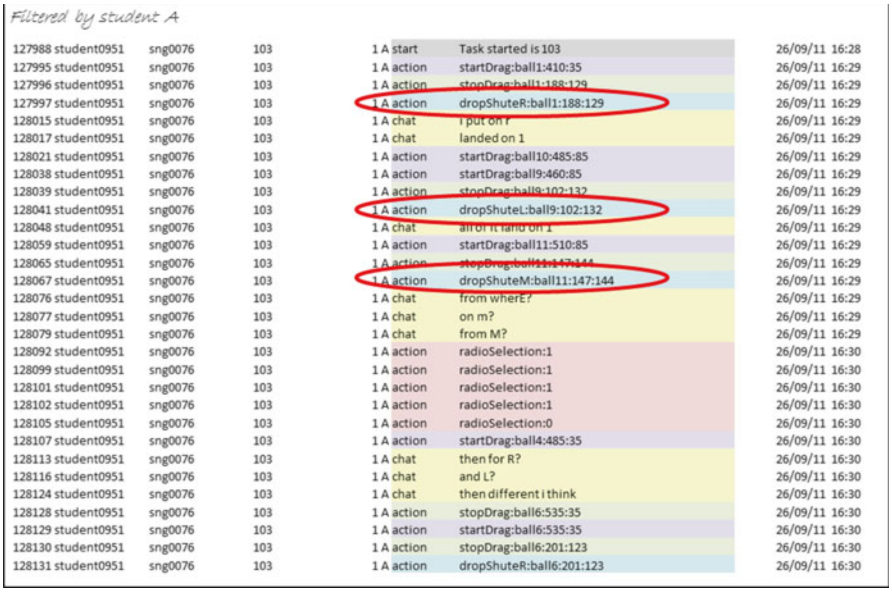
Table 6.3 Example of algorithms to the corresponding indicator

Indicator name	Details	Algorithm	Output
U2L004A	Systematic approach. All positions have been covered.	Step 1: Find all drop ball occurrences captured as dropShute and their corresponding positions as dropShuteL, dropShuteR, dropShuteM.	Count values
U2L004B	Scoring rule: threshold value.	Step 2: Then count all the occurrences of the action recorded under 'dropShute' and their unique positions from the log.	
	Task name: Laughing Clowns.	Step 3: Increase the value of the indicator by one if one or more 'dropShute' occurs in the form of dropShuteR, dropShuteL, or dropShuteM.	
		Step 4: If the total number of unique dropShutes (dropShuteR, dropShuteL, and dropShuteM) from the log is less than three then the value of the indicator is defined as −1 to indicate missing data.	
Global001A	Acceptable time to first action given reading load.	Step 1: Find the starting time when a student joins a collaborative session.	Time
Global001B	Time (in seconds) spent on the task before first action (interpreted as reading time)	Step 2: Find the previous record of the first action.	
	Scoring rule: Threshold time.	Step 3: Find the time of that previous record (from step 2).	
		Step 4: Calculate the time difference obtained (from step 1 and step 3), indicating the time before first action.	
Global005A	Interactive chat blocks: Count the number of chat blocks (A, B) with no intervening actions. Consecutive chats from the same player counts as one (e.g., A,B,A,B = 2 chat blocks; A,B,A,B,A,B = 3 chat blocks; AA,B,A,BB = 2 chat blocks)	Step 1: Find all the consecutive chat from student A and B without any intervening action from A or B. Treat two or more consecutive chats from a single student as one chat.	Count values
Global005B	Scoring rule: threshold number.	Step 2: Increase the value of the indicator by one if one block is found.	



teamID	studentID	W8L101A	W8L102A	W8L110A	W8L313A	W8L314A	W8L315A	W8L320A	W8L213A	W8L211A
aus1141	student1281	1	0	0	1	1	0	4	1	1
aus1151	student1301	0	0	0	1	0	1	3	1	0
aus1152	student1303	1	0	0	1	1	0	31	1	0
aus1162	student1323	0	0	0	1	0	0	1	1	0
aus1288	student1575	1	0	0	1	1	0	13	1	1
aus1292	student1583	1	0	0	1	1	1	0	1	1
aus5001	student5001	1	2	0	1	1	0	4	1	1
aus5008	student5015	1	3	1	1	1	0	2	1	1
aus5092	student5183	1	2	1	1	1	0	0	1	1
aus5104	student5207	0	0	0	1	0	1	0	1	1
aus5106	student5211	0	0	0	1	0	1	0	1	1
aus5183	student5365	1	5	1	1	1	1	7	1	1
aus5189	student5377	0	0	0	1	0	1	16	1	1
aus5191	student5381	0	0	0	1	1	1	12	1	1
sng3008	sngstud3015	1	0	0	1	1	0	4	1	1
sng3012	sngstud3023	0	0	0	1	0	0	2	1	1
sng3120	sngstud3239	1	0	0	1	1	0	8	1	1
sng3122	sngstud3243	1	7	1	1	0	1	7	1	1
sng3124	sngstud3247	1	0	0	1	0	1	1	1	1
sng3127	sngstud3253	1	6	0	1	1	1	0	1	1
sng3202	sngstud3403	1	0	1	1	0	0	6	1	1
sng3203	sngstud3405	1	0	0	1	1	0	2	1	1
sng3210	sngstud3419	1	2	0	1	1	0	11	1	1
sng3307	sngstud3613	1	4	1	1	1	0	22	1	1
sng3312	sngstud3623	1	8	1	1	1	1	9	1	1
sng3313	sngstud3625	1	1	0	1	0	1	9	1	1
sng3359	sngstud3717	1	2	1	1	1	1	10	1	1
sng3365	sngstud3729	1	0	1	1	1	1	15	1	1
sng3450	sngstud3899	1	0	0	1	1	0	13	1	1
sng3452	sngstud3903	1	0	0	1	0	0	3	1	0
sng3458	sngstud3915	1	1	0	1	1	1	24	1	1
sng3468	sngstud3935	1	3	1	1	1	0	9	1	1
sng3476	sngstud3951	0	0	0	1	0	0	18	1	1
sng3480	sngstud3959	1	0	0	1	0	0	3	1	1

Fig. 6.4 Excerpt from a coded data file



Filtered by student A

```
127988 student0951   sng0076   103    1 A start    Task started is 103          26/09/11 16:28
127995 student0951   sng0076   103    1 A action   startDrag:ball1:410:35       26/09/11 16:29
127996 student0951   sng0076   103    1 A action   stopDrag:ball1:188:129       26/09/11 16:29
127997 student0951   sng0076   103    1 A action   dropShuteR:ball1:188:129     26/09/11 16:29
128015 student0951   sng0076   103    1 A chat     i put on r                   26/09/11 16:29
128017 student0951   sng0076   103    1 A chat     landed on 1                  26/09/11 16:29
128021 student0951   sng0076   103    1 A action   startDrag:ball10:485:85      26/09/11 16:29
128038 student0951   sng0076   103    1 A action   startDrag:ball9:460:85       26/09/11 16:29
128039 student0951   sng0076   103    1 A action   stopDrag:ball9:102:132       26/09/11 16:29
128041 student0951   sng0076   103    1 A action   dropShuteL:ball9:102:132     26/09/11 16:29
128048 student0951   sng0076   103    1 A chat     all of it land on 1          26/09/11 16:29
128059 student0951   sng0076   103    1 A action   startDrag:ball11:510:85      26/09/11 16:29
128065 student0951   sng0076   103    1 A action   stopDrag:ball11:147:144      26/09/11 16:29
128067 student0951   sng0076   103    1 A action   dropShuteM:ball11:147:144    26/09/11 16:29
128076 student0951   sng0076   103    1 A chat     from wherE?                  26/09/11 16:29
128077 student0951   sng0076   103    1 A chat     on m?                        26/09/11 16:29
128079 student0951   sng0076   103    1 A chat     from M?                      26/09/11 16:29
128092 student0951   sng0076   103    1 A action   radioSelection:1             26/09/11 16:30
128099 student0951   sng0076   103    1 A action   radioSelection:1             26/09/11 16:30
128101 student0951   sng0076   103    1 A action   radioSelection:1             26/09/11 16:30
128102 student0951   sng0076   103    1 A action   radioSelection:1             26/09/11 16:30
128105 student0951   sng0076   103    1 A action   radioSelection:0             26/09/11 16:30
128107 student0951   sng0076   103    1 A action   startDrag:ball4:485:35       26/09/11 16:30
128113 student0951   sng0076   103    1 A chat     then for R?                  26/09/11 16:30
128116 student0951   sng0076   103    1 A chat     and L?                       26/09/11 16:30
128124 student0951   sng0076   103    1 A chat     then different i think       26/09/11 16:30
128128 student0951   sng0076   103    1 A action   stopDrag:ball6:535:35        26/09/11 16:30
128129 student0951   sng0076   103    1 A action   startDrag:ball6:535:35       26/09/11 16:30
128130 student0951   sng0076   103    1 A action   stopDrag:ball6:201:123       26/09/11 16:30
128131 student0951   sng0076   103    1 A action   dropShuteR:ball6:201:123     26/09/11 16:30
```

Fig. 6.5 Excerpt from a process stream log file for the Laughing Clowns task

Scoring

Indicators can be thought of as the equivalent of items in a conventional test. In order to obtain an estimate of student ability from the scored indicators it is necessary that the status of one indicator does not affect or depend on the status of others. Requiring indicators to be stochastically independent also avoids the complexity of scoring the absence of an indicative behaviour when it is dependent on another event. For instance, if indicator 002 is dependent on indicator 001, and both are dichotomous, the assessment of indicator 002 = 0 will differ depending on whether indicator 001 = 0 or 1.

Dichotomously Scored Indicators

Most indicators of behaviours in the AC21S tasks are designed to be indicative only of the presence or absence of an observable behaviour. This would provide for each student a coded value of '1' to the indicator if it is present and a coded value of '0' to the indicator if it is absent. Through the forcing of most of the indicators into a dichotomy, the interpretation of indicators becomes simpler than is necessary for partial credit coding and scoring. In the Laughing Clowns task, for example, a player needs to leave a minimum number of balls for his/her partner in order for the task to be completed successfully. If the process data shows that this minimum number was satisfied, the indicator can be scored as 1. If it is not satisfied, it is scored as 0.

Frequency-Based Indicators – Partial Credit Scoring

In cases where a particular indicative behaviour is monitored for frequency of occurrence, recording the frequency counts is useful (as indicated in Table 6.3), especially when the cut-off for a qualitatively differentiable interpretation of the behaviour is not clear. For example, the total time taken on a task and the time taken for a player to respond to a partner query can range from a few seconds to several minutes. A dichotomy-based score cannot capture the subtlety of differences in such a case. In the Laughing Clowns example given above, the cut-off value is well-defined because success on this task is impossible beyond the minimum number of balls retained. However, in other tasks this situation may not be recordable in such clear-cut values. There will be an intuitive interpretation that more errors mean less problem solving ability, but it might not be clear where to place a cut-off point for scoring purposes. In these situations, the counts of indicative behaviour are recorded and used to construct a frequency distribution of values for later scoring.

Frequency-based indicators need to be converted into polytomous score values by setting cut-off or threshold values. The distribution typically takes the form similar

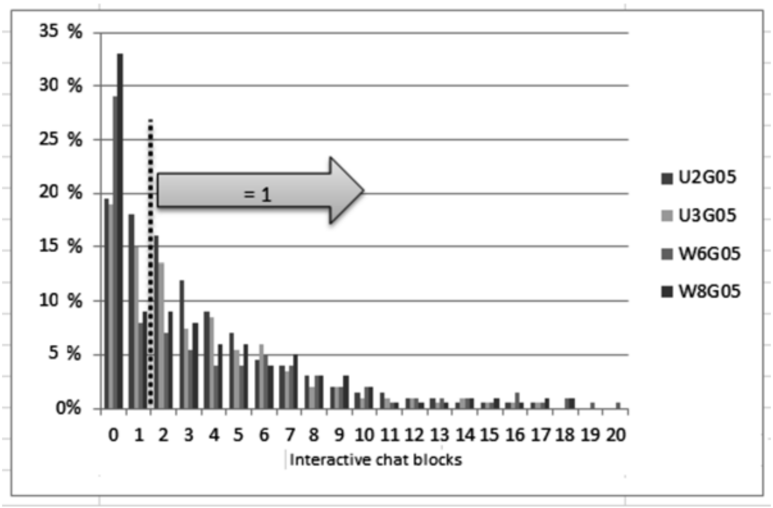


Fig. 6.6 Frequencies of inferred interactive chat blocks across four tasks for setting dichotomous categories

to an exponentially decreasing function or a unimodal function with a positive skew. An example of a decreasing function is shown in Fig. 6.6, where the distribution of inferred interactive chat blocks (chat A-chat B-chat A) for four tasks is illustrated. It shows a similar pattern of decreasing numbers of blocks, although the rate of decrease differs among the tasks to some degree. This type of distribution is scored by deciding where to put a cut-off point that divides the values into a dichotomy (high-low performance levels). If the cut-off value is set at 2, students who have interactive chat blocks of 0–1 get a score of 0, while those who have more chat blocks ($n \geq 2$) get a score of 1. The dichotomous scores can then be interpreted similarly to the presence-absence type of indicators where chat blocks ≥ 2 are taken as evidence of interaction (conversely, less than 2 chat blocks would be taken as insufficient evidence of interaction).

A second example, illustrated in Fig. 6.7, shows the distribution of response time on a question for the Hot Chocolate task. In this example, the mode is around 12 s, with the majority of elapsed time measures falling between 6 and 20 s. Deciding which range of values is qualitatively better is more difficult than in the previous example. Unlike the first example, where the scores were dichotomous, a unimodal distribution can have partial credit assigned to more than two different value ranges. Deciding the various value ranges and their score conversion equivalents can be done using empirical distributions and information obtained from relevant literature. For example, the period that elapses between chat and a following action could be regarded as 'wait time' and, although the concept of wait time in collaboration differs in intention and meaning to the 'wait time' in the literature, it can be used as a guide. The original concept of 'wait time' in a classroom setting refers to the time between a teacher-initiated question and a response from students (Rowe 1972). In

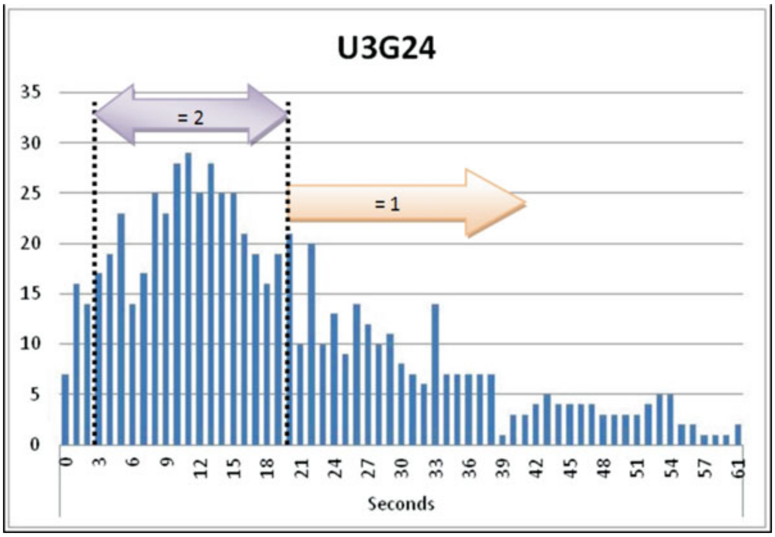


Fig. 6.7 Example histogram of a response time indicator for polytomous categories in indicator U3G24

that field of study, Tobin (1987) and Stahl (1990) suggested a minimum of three seconds wait as a threshold for positive student outcomes, such as increased chances of correct responses and higher cognitive levels of response. The context of their 'wait time' is different from the online setting, and their method of measurement was different from that of the collaborative tasks in ATC21S, but their concept provides a possible lower threshold for a reasonable score bracket (e.g., 0–3 s = 0, 3–20 s = 2, >20 s = 1).

Due to the unique nature of the ATC21S scoring approach, there was very little existing literature that could be used as a guide in setting the cut-off values for most of the process stream data. Since the empirical data for this variable were being captured for the first time in this project, setting the threshold cut-off values and assigning the partial credit scores was necessarily exploratory, and adjustments were made iteratively after calibration and interpretation. Setting the initial cut-off values was a precursor to calibration. The values were regarded as tentative descriptions of (qualitative) levels which were then checked for model fit and meaning during the calibration and scaling process.

Evidence of Collaboration Within Indicators

The evidence of collaboration in a task is primarily based on communication between the players. But it is more than simple communication. Student communication is not necessarily collaborative, or even cooperative. Such an interpretation at best would be simplistic and at worst incorrect. The ATC21S (Hesse et al. 2015; Chap. 2), and the PISA (OECD 2013) definitions of collaborative problem solving are clearly more nuanced than this. Using such simplistic definitions cannot help teachers develop their students' skills. Collaboration involves sharing, perspective taking, joint planning, decision making and shared goals. This cannot be summarised by a single indicator – 'students communicated'. It will involve both direct and indirect communication.

Indirect communication is inferred through actions that can be observed by collaborative partners. With this in mind, a specific approach to capturing chat was adopted. In the problem solving context, portions of the messages were recorded using a series of identifiable keywords. For collaboration, the *presence* of chat was recorded and the content of the chat was not taken into account. Chat linked to action – pre and post a chat event – was used to infer collaboration. This approach had the advantage of simplifying the data collection directly from the process stream while recognising the complexity of the collaboration itself. The presence/absence of chat, coupled with response time and action sequence data (i.e., when the chat occurred with respect to other actions or events), allowed a process to be used to infer collaboration. It was cross-checked by a separate panel of approximately 20 graduate students directly interpreting patterns of chat and action. This process made it clear that a simplistic APP approach which merely identifies the presence of communication is unlikely to enable collaboration to be accurately inferred.

There were several combinations of chat and action that could be interpreted as evidence of collaboration. Communication was inferred from patterns of chat or a combination of chat and action. If there was a presence of chat in the Laughing Clowns task after the last ball had been used and before the question had been answered then it was inferred that the students were discussing the potential answer. This was supported by the analyses of chat content.

The pattern of player-partner (A-B) interaction was also important to capture. For every pattern of chat-action possibility, player-partner combinations were also captured. The length (and hence the number of combinations) of player-partner interaction is unlimited (i.e., A, B, A, B, B, etc.). Hence, a limit of three sequences was adopted. With this limit in place, only the following player-partner combinations were possible: (1) A, B, A; (2) A, B, B; and (3) A, A, B. These combinations apply only to the action of the initiating student (A). Each student was coded separately in the data file, so the perspective changed when the other student (B) was scored. Only an interaction that was initiated by a student was scored for each student (i.e., we only scored for A the player-partner combinations that began with A, and vice-versa). Examples of combinations of interactions that can be captured are summarised in Table 6.4. In this table, the type of interaction (column 1) refers to all possible combinations of chat and action in a three-event block; the perspective (columns 4 and 5) refers to the sequence of player interaction (column 3) for these blocks from the perspective of the scored player (thus, it always begins with the scored player).

Table 6.4 Examples of inferred interactive chat-action combinations

Type	Measurement	Combination	Perspective from student A	Perspective from student B
Interactive **chat-action-chat** blocks	count	player + player + partner	AAB	BBA
	count	player + partner + partner	ABB	BAA
	count	player + partner + player	ABA	BAB
	count	player + player + player	AAA	BBB
Interactive **chat-action-action** blocks	count	player + player + partner	AAB	BBA
	count	player + partner + player	ABA	BAB
Interactive **chat-chat-action** blocks	count	player + partner + partner	ABB	BAA
	count	player + partner + player	ABA	BAB
Interactive **action-action-chat** blocks AAC	count	player + partner + partner	ABB	BAA
	count	player + partner + player	ABA	BAB

Defining the Skills Progression

After the rule-based indicative behaviours were identified, coded, and scored, the empirical data were examined to determine whether the mapping was consistent with the relevant skill in the conceptual framework (Hesse et al. 2015). This preliminary empirical analysis was undertaken to check if the relative difficulty of each indicator was consistent with the skill levels in the conceptual framework (Hesse et al. 2015). For example, an indicator that was interpreted and mapped to a simple level of participation in a task was expected to be less difficult (i.e., have a higher probability of being observed) than an indicator matched to systematic and exhaustive participation in optional activity in the problem space (a lower probability of being observed). Indicators were also reviewed by a panel to check the mapping of each indicator was relevant to the skill it was intended to measure. This panelling process also refined the definition of each indicator so that there is a clear link between the algorithm and the measurement construct. For example, an indicator algorithmically defined as "number (count) of resets (for the game)" can be refined and specified by extending the definition with "exploration activity and initial understanding of problem space". The refined conceptual descriptors were completed for all indicators independent of the empirical quantification of the item's relative position along the construct continuum (i.e., before they were placed into a hierarchical order of item difficulty [delta] based on a scaling under the Rasch

Model). After the indicators were ordered, based on empirical parameter estimates of their deltas, the hierarchy of the descriptors was again assessed to check that they make sense within a broader collaborative problem solving framework. This review process was completed several times to ensure that the conceptual descriptors are supported by empirical item location, which in turn informs the construct continuum. In the same process, the review clarifies which items have deltas that do not fit the theoretical model, and thus are not informative or meaningful within the overall structure of the construct.

After the skills progression was developed, levels of progression were identified in order to help teachers to cluster students more effectively and aid their instruction of CPS skills. The indicators were split into their two dimensions – social or cognitive – based on their previous mapping. Cognitive and social dimensions were each assessed independently to define a continuum and levels within each. Skills within each dimension were identified to represent the progression from novice to expert.

At this point, indicators which proved to have little value or influence on the interpretation were removed. The deletions were based on extensive item review, psychometric characteristics, and mapping to the theoretical continuum. The pruning is due to some indicators not matching the conceptual framework vis-a-vis their placement as expected from the theoretical progression. Also pruned were some indicators with coding issues which couldn't be resolved after extensive review.

Multiple calibrations allowed for comparison and analysis of item parameters. The stability of these parameters remained, even if the number of indicators was reduced considerably. As a result of the refinement process, the number of indicators was reduced from over 450 to fewer than 200. The removal of poorly 'fitting' indicators reduced the standard errors of the item parameters, while maintaining the reliability of the overall set.

Challenges and Future Directions

Even the most successful projects have lessons from which we can learn. The purpose of this section is to describe some of the lessons learned during development and deployment of the collaborative problem solving task design and delivery. What follows are descriptions of measures that are recommended as good practice to improve the design and implementation of such assessment tasks and data structure.

Design of the session log is crucial. The importance of leveraging complex and interactive assessment tasks not only to implement assessment delivery but also to establish automated scoring has been highlighted by many researchers (Mills et al. 2002; Williamson et al. 2006). The format in which data points are captured ensures efficient interpretation of user responses for establishing reliable scoring rules based on the evidence of interactive patterns from the logs. To validate the scoring rules, log files should be structured to allow human interpretation without obscuring their understanding. For example, each user action or response should be recorded as separate attributes in human readable format and as single instances with corresponding user identification, task and present state, timestamp, record index and other data as required for the task. In addition, it is imperative to ensure the optimum level of detail capture for both analysis and processing of data for automating the scoring process. Through the delivery of logs from one developer for ATC21S, it was apparent that the contents of the responses captured should be recorded under several attributes in a well-structured database to optimise the processing time for scoring complex data and to ensure uninterrupted traffic load on the system. Timestamping was found to be essential for logging response data from the assessment tasks. Timed data, along with database indexing, proved to be useful in sequencing user interactions with the task environment. In the current case, database design allowed the capture of user responses only in corresponding seconds. From the accumulated data it was observed that more precise times (i.e. milliseconds) when users respond may often be required to differentiate sequences of actions that occur almost simultaneously. Multiple actions can be recorded as occurring at the same time (in seconds), but actions do occur consecutively and this should be more accurately reflected in the way they are captured and arranged in the database.

Event types described across different tasks should be defined in a uniform method. Consistency in event definition is important for future developers of similar tasks and for understanding the events they represent. In the present context, the assessment tasks were initially designed by different developers. As a result, the language and format used to define the same event – for instance ‘chat’ – were quite different and had different naming conventions across the various tasks (e.g. ‘Send message’, ‘Type message’, ‘Enter text’ etc.).

Development of interactive tasks and the capacity to automatically score responses is a resource intensive undertaking, even in traditional and well-defined educational domains (Masters 2010). Due consideration should be given to future analysis needs while designing complex assessments of this nature. Emphasis should be given to understanding the intended use of the data to support inferences to the diagnostic richness that can be pertained through interpretation and analysis. This is important, since extension towards more complex data accumulation in less concisely defined educational domains, such as interactive problem solving, may challenge conventional approaches to scaling educational assessment data and may be inadequately handled (Rupp 2002).

While the content of actions can be assessed, assessing the content of chat is currently beyond the limitations of this project. There are some robust automated text analysis programs that analyse large-volume texts – for example, essays, formal open-ended items and reports. One application of these is the Coh-Metrix (Graesser et al. 2004), a computational linguistics tool, which can analyse text for cohesion, language/discourse, and readability. However, the challenges posed to ATC21S by the use of such a tool were too great. To begin with, as the project is international, there are several different language translations involved, which could lead to translation issues within automated text analysis programs. The automated text analysis software would also need to be quite sophisticated to classify the text blocks into the predefined activity type – for example, chat/action/chat. A further difficulty is the quantity and quality of text that may be present within a task's chat box. Students may provide single word answers or low volumes of text, and the type of software available is designed for large quantities of text. The quality of chat is likely to present problems, including grammatical errors, non-standard syntax, abbreviations, and synonyms or 'text-speak' – all of which involve non-standard spelling that would not be recognised by current software designed for more formal language. A key consideration for future deployment is the identification of ways to capture these text data in an understandable coded form or to translate them into a uniform language (such as English) before they are recorded.

References

Arroyo, I., & Woolf, B. P. (2005). *Inferring learning and attitudes from a Bayesian Network of log file data.* Conference paper presented at the Artificial Intelligence in Education: Supporting learning through intelligent and socially informed technology. Amsterdam, The Netherlands.

Bennett, R. E., Jenkins, F., Persky, H., & Weiss, A. (2003). Assessing complex problem solving performances. *Assessment in Education, 10*(3), 347–359.

Care, E., Griffin, P., Scoular, C., Awwal, N., & Zoanetti, N. (2015). Collaborative problem solving tasks. In P. Griffin & E. Care (Eds.), *Assessment and teaching of 21st century skills: Methods and approach* (pp. 85–104). Dordrecht: Springer.

Chung, G. K., de Vries, L. F., Cheak, A. M., Stevens, R. H., & Bewley, W. L. (2002). Cognitive process validation of an online problem solving assessment. *Computers in Human Behaviour, 18*, 669–684.

Fu, W. T. (2001). ACT-PRO action protocol analyzer: A tool for analyzing discrete action protocols. *Behavior Research Methods, Instruments, and Computers, 33*(2), 149–158.

Graesser, A. C., McNamara, D. S., Louwerse, M. M., & Cai, Z. (2004). Coh-metrix: Analysis of text on cohesion and language. *Behavior Research Methods, Instruments, and Computers, 36*(2), 193–202.

Greiff, S., Wüstenberg, S., & Funke, J. (2012). Dynamic problem solving: A new assessment perspective. *Applied Psychological Measurement, 36*(3), 189–213.

Hesse, F., Care, E., Buder, J., Sassenberg, K., & Griffin, P. (2015). A framework for teachable collaborative problem solving skills. In P. Griffin & E. Care (Eds.), *Assessment and teaching of 21st century skills: Methods and approach* (pp. 37–56). Dordrecht: Springer.

Masters, J. (2010). Automated scoring of an interactive geometry item: A proof-of-concept. *Journal of Technology, Learning, and Assessment, 8*(7). Retrieved April 12, 2013, from http://escholarship.bc.edu/jtla/, http://www.jtla.org

Mills, C. N., Potenza, M. T., Fremer, J. J., & Ward, W. C. (2002). *Computer-based testing: Building the foundation for future assessments*. Mahwah: Lawrence Erlbaum Associates.

OECD. (2013). *PISA 2015: Draft collaborative problem solving framework*. http://www.oecd.org/pisa/pisaproducts/Draft%20PISA%202015%20Collaborative%20Problem%20Solving%20Framework%20.pdf. Accessed 7 July 2014

Pelligrino, J., Chudowsky, N., & Glaser, R. (2001). *Knowing what students know: The science and design of educational assessment*. Washington, DC: National Academy Press.

Polya, G. (1945). *How to solve it* (1st ed.). Princeton: Princeton University Press.

Polya, G. (1957). *How to solve it* (2nd ed.). Princeton: Princeton University Press.

Rowe, M. (1972). *Wait time and rewards as instructional variables, their influence in language, logic, and fate control*. Paper presented at the National Association for Research in Science Teaching, Chicago, IL.

Rupp, A. (2002). Feature selection for choosing and assembling measurement models: A building-block-based organisation. *International Journal of Testing, 2*(3/4), 311–360.

Schoenfeld, A. H. (1985). *Mathematical problem solving*. New York: Academic.

Stahl, R. (1990). *Using "think-time" behaviors to promote students' information processing, learning, and on-task participation. An instructional module*. Tempe: Arizona State University.

Tobin, K. (1987). The role of wait time in higher cognitive level learning. *Review of Education Research, 57*(1), 69–95.

Williamson, D. M., Mislevy, R. J., & Bejar, I. I. (2006). *Automated scoring of complex tasks in computer-based testing*. Mahwah: Lawrence Erlbaum Associates.

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