Behavior Research Methods
2006, 38 (3), 382-389

Developing a coding scheme for detecting usability and fun problems in computer games for young children

W. BARENDREGT and M. M. BEKKER
Eindhoven University of Technology, Eindhoven, The Netherlands

This article describes the development and assessment of a coding scheme for finding both usability and fun problems through observations of young children playing computer games during user tests. The proposed coding scheme is based on an existing list of breakdown indication types of the detailed video analysis method (DEVAN). This method was developed to detect usability problems in task-based products for adults. However, the new coding scheme for children's computer games takes into account that in games, fun, in addition to usability, is an important factor and that children behave differently from adults. Therefore, the proposed coding scheme uses 8 of the 14 original breakdown indications and has 7 new indications. The article first discusses the development of the new coding scheme. Subsequently, the article describes the reliability assessment of the coding scheme. The any-two agreement measure of 38.5% shows that thresholds for when certain user behavior is worth coding will be different for different evaluators. However, the any-two agreement of .92 for a fixed list of observation points shows that the distinction between the available codes is clear to most evaluators. Finally, a pilot study shows that training can increase any-two agreement considerably by decreasing the number of unique observations, in comparison with the number of agreed upon observations.

Testing products with representative users is one of the core aspects of user-centered design. A common goal of a user test is to identify the parts of a system that cause users trouble and need to be changed. When computer games for children are evaluated, both usability and fun problems can occur, and it is important to fix both. However, no coding scheme of behavior that indicates these problems in computer games for children is available yet. The coding scheme proposed in this article is based on a list of breakdown indication types from the detailed video analysis method (DEVAN; Vermeeren, den Bouwmeester, Aasman, & de Ridder, 2002). This method was developed to detect usability problems in task-based products for adults. However, the new coding scheme for children's computer games should take into account that in games, fun, in addition to usability, is an important factor and that children behave differently from adults. Therefore, the definitions of existing breakdown indications probably need to be changed, new breakdown indications need to be added, and some indications have to be removed. The article starts from the list of DEVAN breakdown indications. Subsequently, the influence of the non-task-based nature of games on the coding scheme is discussed. Furthermore, this article describes new breakdown indications that reflect the observed behavior of children indicating problems in games. Finally, the article discusses how the reliability of the final coding scheme was assessed.

This work was supported by a grant from the Innovation-Oriented Research Program Human–Machine Interaction of the Dutch government. We thank all the evaluators who participated in the experiments. Correspondence concerning this article should be addressed to W. Barendregt, Faculty of Industrial Design, Eindhoven University of Technology, Den Dolech 2, P.O. Box 513, 5600 MB Eindhoven, The Netherlands (e-mail: wolmetb@hotmail.com).

The Coding Scheme

As a starting point for the coding scheme, the list of breakdown indication types of the DEVAN method (Vermeeren et al., 2002) was used. This list is one of the most detailed lists of behaviors indicating usability problems. The original list is given in Appendix A.

Nature of games. The aim of the list of breakdown indications of DEVAN is to show problems during user tests with task-based products, and thus, many breakdown indications on this list also relate to tasks. However, it was decided not to give explicit tasks during the evaluation of games, because a major problem with the giving of tasks for the evaluation of computer games is that both the given task and the game can provide goals. Games provide goals in order to create a challenge (Malone & Lepper, 1987). The goals set by the tasks may interfere with the intended goals of the game because the children feel obliged to fulfill both the goals provided by the tasks and the goals provided by the game (Barendregt, Bekker, & Speerstra, 2003). Therefore, the literal wording of some of the breakdown indications may make them seem not applicable to games. However, games usually do have subgames, and these subgames can be considered tasks, although they are not given to the child by the facilitator. By replacing the term *task* with *subgame*, the indications that refer to tasks can still be used.

When we tried to code some of the gathered video material with the DEVAN coding scheme, it became clear that the breakdown indications *discontinues action*, *repeated action*, and *corrective action* were very hard to determine. For example, is repeated clicking of a button a repeated action? Or, is making a character run back to a safe place a corrective action? It was decided to remove them from the list of breakdown indications. For the breakdown indication *wrong action*, it was decided that the types of actions that could be considered wrong should be clearly defined. Clicking on a part of the screen that cannot be manipulated is considered to be a wrong action. Furthermore, actions that are clearly not what the child wants to do—for example, clicking a button to quit the game before the test is over—are also considered wrong actions. Later on in the analysis, these behaviors may still be regarded as part of the challenge and, therefore, not problems, but they should be noted first.

Fun. Having fun is a key factor in a computer game (Pagulayan, Keeker, Wixon, Romero, & Fuller, 2003), but fun problems are not explicitly covered by the DEVAN breakdown indications, because their aim was not to detect problems in products that do not aim to be fun. Malone and Lepper's (1987) taxonomy for intrinsically motivating instructional environments was used as a starting point to detect fun problems. The taxonomy contains four main heuristics: *challenge*, *fantasy*, *curiosity*, and *control*. On the basis of observations of children playing several computer games (Barendregt & Bekker, 2004), we considered what verbal or nonverbal behavior children would display if these heuristics were violated.

Challenge. When the provided challenge in a (sub)game is too high, a child will want to quit the (sub)game or ask help from the facilitator. The first indication (*quit*) is already present in the original list; asking for *help from the researcher* has to be added to the list. When the provided challenge is too low, the child may want to stop playing the (sub)game or become bored (Rauterberg, 1995). The first indication (*quit*) is present in the list; the second indication, *bored*, needs to be added. Providing the right challenge level throughout the game is also called *balancing the game* (Rouse, 2005).

Fantasy. Computer games have the potential to immerse players in a fantasy world (Rouse, 2005). However, when the child is not pleased with the provided fantasy, he or she may express dislike—for example, because the fantasy is too childish or too scary. This indication *dislike* needs to be added.

Curiosity. Curiosity is aimed more at learning through playing than at playing just for fun. However, even for noneducational computer games, a lack of curiosity can make a game less pleasurable to play. According to Rouse (2005), "once players have accomplished a goal in the game, they do not want to have to repeat their accomplishment" (p. 15). Children may specifically signal that they are frustrated or demotivated by repetition or by a lack of progress and new experiences. Making and keeping players curious about new parts of the game is part of *balancing the game* (Rouse, 2005). The breakdown indication *doubt*, *surprise*, *frustration* (*DSF*) is already present in the list. The breakdown indication *bored*, added to detect possible challenge problems, can also be used as an indicator for a curiosity problem. Children may also want to quit a subgame before the goal is reached because they are not curious to explore any further. The indication *(sub)game stopped* can be used as a breakdown indication for this type of problem. Curiosity problems and problems related to too low a challenge level are very similar.

Control. Control problems may occur when the game takes over control—for example, during feedback or a story—and the user cannot regain control even though he or she wants to. According to Rouse (2005), "players expect to do, not to watch." Although control, in the sense of game mechanics, is necessary for playing a game, specific control problems can arise when the game takes over control for too long and this frustrates the user because he/she wants to interact. For example, a control problem occurs when the user cannot skip a long introduction or feedback, even though the user already knows the content and wants to go on playing. Children will show impatience when such a control problem occurs. The *impatient* indication should be added to the list of breakdown indications.

Behavior of children with games. Preliminary versions of the coding scheme were used to code the behavior of children playing different adventure games—for example, "Robbie Rabbit: Fun in the Clouds" (Mindscape, 2003), "Rainbow, the Most Beautiful Fish in the Ocean" (MediaMix Benelux, 2002), and "World in Numbers Group 3" (Malmberg, 2003). While we were trying to code these user test data, some behavior was discovered that could not be coded with the existing breakdown indications. Further breakdown indications for this behavior were added to the coding scheme. These breakdown indications will be explained below.

Perception problem. Children sometimes complained that they had difficulties hearing or seeing something properly. Examples are texts that are too small to be read or verbal explanations given in a very low volume. Another example is a situation in which the goal of a subgame is explained verbally by one of the characters in the game. Sometimes, another character would talk through this explanation of the goal, making it hard to hear. The latter problem could also be called an *attention problem*, because the information is perceivable but the child is unable to attend to the right kind of information. However, because the coding is not performed to classify problems but just to locate breakdown indications, the single breakdown indication type *perception problem* was added to the list of indications. This breakdown indication type covers all situations in which children explicitly complain about visibility or audibility.

Passive. Some children would stop interacting with the game when they did not know how to proceed. They would just sit and stare at the screen. Furthermore, games are often dialogues between the player and some of the characters; the child has to respond to questions and requests of the characters. However, it was not always clear to the children that an action was required of them. The children

would remain passive when an action was necessary. It was therefore decided to include *passivity*, when children were supposed to act, as an indication of a problem.

Wrong explanation. Sometimes, children at first did not seem to have a problem playing a subgame, but later, they gave an explanation of something that had happened that was not correct and could lead to other problems. For example, in "Rainbow, the Most Beautiful Fish in the Ocean" (MediaMix Benelux, 2002), children can decorate pictures of fishes with stamps (see Figure 1). When a child clicks outside the lines of the picture, the chosen stamp is deactivated.

However, one of the children in the user tests clicked outside the picture without noticing it and then said, "I've run out of stamps! I have to get new ones." Because of his wrong explanation, it was clear that the boy did not understand the deactivation of the stamps outside the picture. Furthermore, he kept clicking outside the picture, having to select the same stamp many times. It was decided to add *giving a wrong explanation of something that happened in the game* as an indication of a problem. This is similar to what Lavery and Cockton (1997) call a "knowledge mismatch problem." However, it should be noted that the breakdown indication type *wrong explanation* refers only to a verbal expression made by the child, not to a problem type, since this needs further analysis. This analysis is performed after the initial detection of the breakdown indications.

Adaptations to prevent overlap. Some of the breakdown indications in DEVAN have two similar versions: one as an observed action on the product and one as a verbal utterance or nonverbal behavior. When the original coding scheme was used, it became clear that evaluators usually code only one of the two versions. Therefore, it was decided to merge these breakdown indications into one. This holds for the indications *execution problem* and *execution difficulty* and for the indications *stop* and *quit*. Furthermore, the distinction between *searches for function* and *puzzled* was unclear. It was decided to remove the indication *searches for function* because the *puzzled* indication could usually cover these situations even when both occurred.

The proposed coding scheme. The proposed list of breakdown indication types is still similar to the one used in DEVAN. The new list is depicted in Appendix B. It contains seven new breakdown indications (*passive*, *impatience*, *wrong explanation*, *bored*, *perception problem*, *help*, and *dislike*), and six breakdown indication types are removed from the original list (*repeated action*, *corrective action*, *discontinues action*, *execution difficulty*, *searches for function*, and *stop*). *Passive*, *wrong explanation*, and *perception problem* were added because children displayed this behavior but this could not be captured by the DEVAN codes. The breakdown indication types *impatience*, *bored*, and *dislike* were added in order to capture behavior that is an indication of a fun problem. *Help* was added because in the setup of the user tests, the children could ask help from the facilitator. The breakdown indication types *repeated action*, *corrective action*, and *discontinues action* were removed because it was too hard for evaluators to decide when to code these indications. The breakdown indication types *execution difficulty*, *searches for function*, and *(sub)game stopped* were collapsed with other breakdown indication types because they were actually nonverbal versions of the breakdown indication types *execution problem*, *puzzled*, and *quit*. Evaluators often used only one of the two codes.

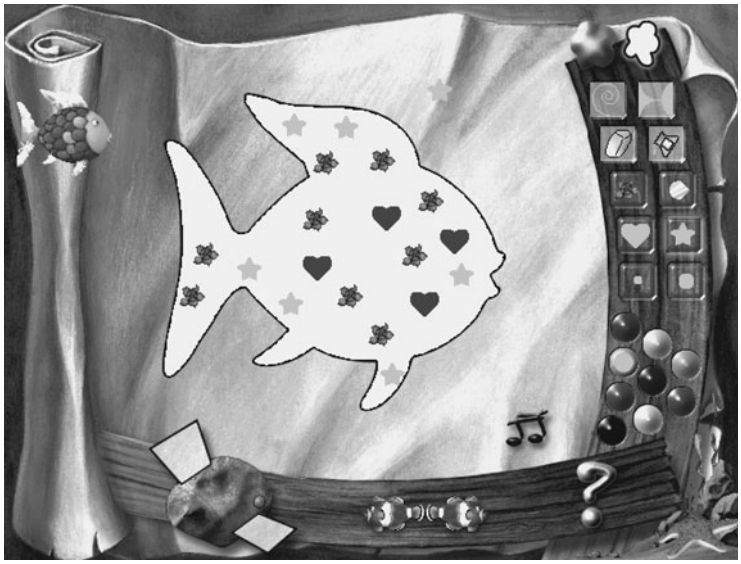

Figure 1. The paint studio in "Rainbow, the Most Beautiful Fish in the Ocean," MediaMix Benelux, where children can decorate pictures of fishes with colored stamps. When a stamp is used outside the white area, the stamp is deactivated and has to be selected again. Copyright 2002 by MediaMix Benelux.

Measuring the Reliability

To determine the reliability of a coding scheme, several measures are available, such as the intraclass correlation coefficient (ICC), Cohen's kappa, and the any-two agreement measure. However, such measures as ICC are numerical, whereas the breakdown indications are categorized into nonordered classes. Furthermore, both the ICC and Cohen's kappa are based on each evaluator's classifying the same observation points. In the case of free detection of all breakdown-indicating behavior, not all the evaluators may have noticed certain behavior, resulting in different observation points (see Figure 2).

Finally, both ICC and Cohen's kappa assume that the total number of breakdowns that need to be coded is known, or can reliably be estimated. Since it is possible that all the evaluators have failed to observe certain behavior, this is probably not true. Therefore, we used the any-two agreement measure, similar to that of Hertzum and Jacobsen (2001), to determine the reliability of the coding scheme:

$$\frac{\left|P_i \cap P_j\right|}{\left|P_i \cup P_j\right|} \text{ (over all } \tfrac{1}{2}\, n(n-1) \text{ pairs of evaluators).}$$

Here, P_i and P_j are the sets of problem indications detected by Evaluators i and j, respectively, and n is the number of evaluators. By replacing the sets of problems with sets of breakdown indication type/time pairs, this measure can also be used to determine the agreement of coded observation points.

Procedure. To determine the any-two agreement for the proposed coding scheme, three evaluators and one of the authors coded a 10-min piece of videotape of a child playing a computer game called "Rainbow, the Most Beautiful Fish in the Ocean" (MediaMix, 2002). The child was asked to talk as much as possible about playing this game but was not reminded to think aloud. Furthermore, the child was not asked to perform any tasks, because this was assumed not to be a representative way of using a game (Barendregt et al., 2003).

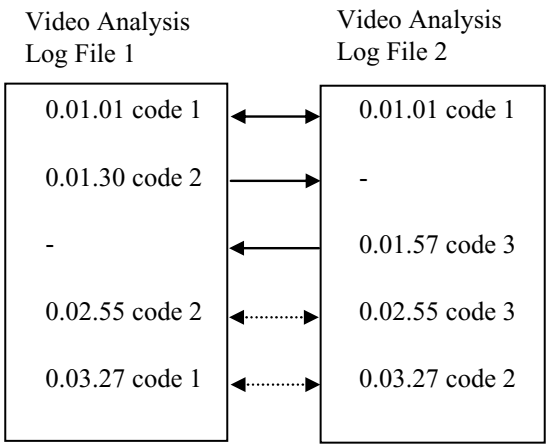

Figure 2. Different observation points from two different evaluators.

Before the actual coding, all the evaluators attended a classroom meeting in which all the breakdown indication types were explained. After the explanation, the individual evaluators could borrow a laptop on which the Noldus Observer (Noldus, 2002) was installed, along with the coding scheme. They also received a CD-ROM with the game used in the user test, so they could become familiar with it before the coding.

After the evaluators had completed their observations individually, all the observations were compared with each other to determine the number of agreements, disagreements, and unique observation points. On the basis of preliminary analyses with the coding scheme, it was decided that observation points within 4 sec of each other should be counted as the same observation points. This 4-sec interval was chosen because some evaluators coded behavior after it had happened, whereas others coded behavior by reviewing the data and going back to the beginning or the middle of a certain behavior to code it (especially for sentences). By using 4 sec, these codes were still considered to relate to the same situation.

When two evaluators had the same observation point and the same code at this point, it was counted as an agreement. When one of the evaluators had an observation point and the other did not, it was counted as a unique observation for the evaluator who had coded it. When two evaluators had the same observation point but unequal codes, this was counted as a disagreement.

Results. Table 1 shows the results of the comparison for each pair of evaluators.

The average any-two agreement was 38.5%, which is rather low. However, it is in the range of any-two agreement measures reported in an overview study by Hertzum and Jacobsen (2001), although these numbers were based on problem detection, instead of breakdown indication detection. A qualitative analysis was performed to determine the causes for the unique observations. A major cause for unique observation points was that sometimes, the evaluators had not just coded all indicating behavior but had made a decision about the severity or multiple occurrence of a breakdown. For example, when a child made the same error more than once, some evaluators had stopped coding this behavior because they reasoned that the problem would be reported anyway. However, this was not the intention of the coding scheme, because it was meant to just detect problem-indicating behavior. Reasoning about problems and multiple occurrences of problems is something that should be done in a later phase of the analysis.

It also became clear that some of the unique observation points were caused by unintended additional interpretations of the breakdown indication types. An example of an additional interpretation is that one of the evaluators had coded *recognition of error or misunderstanding* also when a child said something like "I have been here before," which is not a recognition of an error or misunderstanding but a recognition in general. Training the evaluators more intensively about how to apply the coding scheme could

Table 1
Any-Two Agreement Measure, Number of Agreements, Number of Unique Observation Points, and Number of Disagreements for Each Evaluator Pair

Evaluator A × Evaluator B	Any-Two (%)	Agreements	Unique A	Unique B	Disagreements
Evaluator 1 × Evaluator 2	50	37	15	16	6
Evaluator 1 × Evaluator 3	33	21	28	9	8
Evaluator 1 × Evaluator 4	47	31	25	7	3
Evaluator 2 × Evaluator 3	27	21	39	10	8
Evaluator 2 × Evaluator 4	45	31	24	6	9
Evaluator 3 × Evaluator 4	29	16	17	16	7

probably decrease the numbers of unique observations. This assumption will be investigated more closely in the Influence of Training section.

Another cause for many unique observations is that it is very hard to manually keep track of all clicks that children perform. For example, if a child clicks objects that are not clickable, the evaluator has to register each click of the child by clicking the *wrong action* code in the Observer. When the child is fast, it is very hard for the evaluators to register each click. Often, one of the evaluators will miss one or more clicks, resulting in unique observations. A good solution would be if the clicks were automatically recorded and imported in the Observer file so that the evaluators had only to code the behavior, instead of detecting and coding it. Although missing repeated clicks had a negative influence on the any-two agreement, it should be noted that these clicks would probably be grouped into one problem in a later phase of the analysis, as is also done in SUPEX (Cockton & Lavery, 1999).

Finally, it was discovered that part of the real disagreements was related to the codes *impatient* and *wrong action*. When a child clicked an object rapidly and frequently, it could be coded as *impatient*, because it showed impatience, or it could be coded as *wrong action*, because it usually involved an object that could not be clicked (and therefore, did not respond, resulting in impatience). Two other codes that seemed to lead to disagreement were *puzzled* and *DSF*. *Puzzled* is meant for confusion before an action is executed, *DSF* after an action is executed. However, sometimes it is difficult to determine whether the confusion occurs before or after an action. For example, incomprehensible feedback can lead to confusion about the performed action, but also to confusion about what is expected next. In both cases, we think it is not really important which code is used, since the codes are not used for classification purposes, as long as all the evaluators notice the behavior of interest.

Reliability for a Fixed List of Observation Points

To determine the extent to which unclear breakdown indications contributed to the low any-two agreement, another small study was set up. In this study, it was possible to calculate Cohen's kappa properly, because the evaluators received a list of observation points for which they had to pick a code. This list was created by taking the lists of all four evaluators in the first experiment. When at least three out of the four evaluators in the first experiment agreed on an observation point (but not necessarily on the code), it was included in the list of observation points, resulting in a list of 26 fixed observation points.

Two experienced new evaluators received the latest list of breakdown indications with explanations, a list of observation points, the game, and a CD with the video data. Independently, they had to code all 26 observation points by picking one of the breakdown indications. When they thought that an observation point should be coded by more than one breakdown indication, they could add these additional codes between brackets, but they had to pick one code as being the most important.

Of the 26 given observation points, 24 were coded identically, resulting in an any-two agreement of .92. The agreement between the two evaluators was thus very high. For the two points that were not coded identically, one of the evaluators had actually given the code of the other evaluator as a secondary code.

These results give a clear indication that the coding scheme does not contain unclear breakdown indicator descriptions but that evaluators use different thresholds to indicate a certain behavior as a breakdown indication.

The Influence of Training

The first two experiments showed that differences in the coding were caused mainly by unique observations of the evaluators and less by disagreements about the same behavior. To determine whether the number of unique observations could be decreased by providing more training and discussion about the exact use and purpose of the coding scheme, we set up another exploratory study with one of the authors and a student assistant as evaluators. In this study, both evaluators first coded a 20-min piece of videotape with the coding scheme, after a short introduction of about half an hour. The results of the first coding are depicted in the first row of Table 2. The any-two agreement was again very low.

After the first session, the evaluators had a 1-h discussion about their observations and when to code specific behavior. For example, the author explicitly explained that indicators that appear more than once for the same problem still need to be recorded. Both evaluators now coded another 20-min piece of videotape. The results are depicted in the second row of Table 2.

In the second session, the any-two agreement was doubled, and it is clear that this was due to the much lower proportion of unique observations. Thus, it appears that

Table 2
Number of Agreements, Number of Disagreements, and Number of Unique Observation Points of Both Evaluators for the First and the Second Sessions

Any-Two (%)	Agreements	Disagreements	Unique Observations 1	Unique Observations 2
25	15	6	30	10
55	54	5	21	18

after only 1 h of additional training after a practice session, the any-two agreement becomes much higher.

Discussion

Although created with the greatest care, on the basis of analyses of many pilot test sessions, it is not guaranteed that the coding scheme will help to detect all possible types of problems. Therefore, it is possible that some problems were missed. However, during all the analyses we performed with the coding scheme, it was never felt that another breakdown indication type that was not present in the coding scheme yet was needed.

The coding scheme is specifically created to code the behavior of young children with computer games in the adventure genre in order to analyze test sessions. To use this coding scheme to analyze video data of other users playing games of other genres, it should be checked whether all codes occur, whether additional codes are needed, and whether evaluators are able to reliably identify the relevant behaviors.

Conclusions

In this article, the development of a coding scheme for detecting usability and fun problems in computer games for young children has been described. The coding scheme is based on the DEVAN method (Vermeeren et al., 2002) and is adapted according to the theory of fun in computer games from Malone and Lepper (1987) and from observations of children playing games. Six breakdown indication types were removed from the original list. Three breakdown indication types were removed because evaluators were unable to recognize these when analyzing video data of children playing a game. Three other breakdown indication types were removed because they were related to a very similar verbal breakdown indication type—in which case, the types were joined into one. Seven breakdown indication types were added. Three breakdown indication types were added to be able to detect fun problems. Three other breakdown indication types were added because they described behavior that children had showed when playing a game. One breakdown indication type was added because it was decided that children could ask for help and that, therefore, asking for help would be a clear indication of a problem. The average any-two agreement measure of 38.5% for paired comparisons of four evaluators using this coding scheme shows that thresholds for when a certain user behavior is worth coding are different for different evaluators. However, the any-two agreement of .92 for a fixed list of observation points shows that the distinction between the available codes is clear to most evaluators. Furthermore, it was shown in a pilot study that training the evaluators more intensively about how to apply the coding scheme on the basis of the mistakes that they had made in a previous coding decreases the numbers of unique observations and, therefore, increases the any-two agreement considerably.

REFERENCES

Barendregt, W., & Bekker, M. M. (2004). Towards a framework for design guidelines for young children's computer games. In M. Rauterberg (Ed.), *Proceedings of the IFIP ICEC Conference* (pp. 365-376). Berlin: Springer.

Barendregt, W., Bekker, M. M., & Speerstra, M. (2003). Empirical evaluation of usability and fun in computer games for children. In M. Rauterberg, M. Menozzi, & J. Wesson (Eds.), *Proceedings of the IFIP 8th International Conference on Human–Computer Interaction*, (pp. 705-708). Amsterdam: IOS Press.

Cockton, G., & Lavery, D. (1999). A framework for usability problem extraction. In *Proceedings of the IFIP 7th International Conference on Human–Computer Interaction*, (pp. 344-352). Amsterdam: IOS Press.

Hertzum, M., & Jacobsen, N. E. (2001). The evaluator effect: A chilling fact about usability evaluation methods. *International Journal of Human–Computer Interaction*, **13**, 421-443.

Lavery, D., & Cockton, G. (1997). Representing predicted and actual usability problems. In H. Johnson, P. Johnson, & E. O'Neill (Eds.), *Proceedings of the International Workshop on Representations in Interactive Software Development* (pp. 97-108). London: University of London, Queen Mary and Westfield College.

Malmberg Uitgeverij (2003). Wereld in getallen, Groep 3: Het Pretpark [World in numbers, Group 3: The Theme Park] [Computer software]. Amsterdam: Author.

Malone, T. W., & Lepper, M. R. (1987). Making learning fun: A taxonomy of intrinsic motivations for learning. In R. E. Snow & M. J. Farr (Eds.), *Aptitude, learning, and interaction III: Cognitive and affective process analysis* (pp. 223-253). Hillsdale, NJ: Erlbaum.

MediaMix Benelux (2002). Regenboog, de mooiste vis van de zee [Rainbow, the most beautiful fish in the ocean] [Computer software]. Overijse, Belgium: Author.

Mindscape (2003). Robbie Konijn, Groep 3: Pret in de Wolken [Robbie Rabbit, Group 3: Fun in the Clouds] [Computer software]. Haarlem, The Netherlands: Author.

Noldus (2002). The Observer Pro [Computer software]. Wageningen, The Netherlands: Noldus.

Pagulayan, R. J., Keeker, K., Wixon, D., Romero, R., & Fuller, T. (2003). User-centered design in games. In J. Jacko & A. Sears (Eds.), *Handbook for human–computer interaction in interactive systems* (pp. 883-906). Mahwah, NJ: Erlbaum.

Rauterberg, M. (1995). About a framework for information and information processing of learning systems. In *Preprints of Proceedings of "International Conference on Information System Concepts"* (pp. 4.1-4.15). Marburg: Phillipps University.

Rouse, R., III (2005). *Game design: Theory and practice* (2nd ed.). Plano, TX: Wordware.

Vermeeren, A. P. O. S., den Bouwmeester, K., Aasman, J., & de Ridder, H. (2002). DEVAN: A detailed video analysis of user test data. *Behaviour & Information Technology*, **21**, 403-423.

APPENDIX A
Breakdown Indications From DEVAN, With Descriptions
(Vermeeren, den Bouwmeester, Aasman, & de Ridder, 2002)

Code	Short Description	Definition
	Breakdown Indication Types Based on Observed Actions With the Game	
ACT	wrong action	An action does not belong in the correct sequence of actions. An action is omitted from the sequence. An action within a sequence is replaced by another action. Actions within the sequence are performed in reversed order.
DIS	discontinues action	User points at function as if to start using it but then does not. User stops executing action before it is finished.
EXE	execution problem	Execution of action not done correctly or optimally.
REP	repeated action	An action is repeated with the same effect.
COR	corrective action	An action is corrected with a subsequent action. An action is undone.
STP	task stopped	User starts new task before having successfully finished the current task.
	Breakdown Indication Types Based on Verbal Utterances or Nonverbal Behavior	
WGO	wrong goal	The user formulates a goal that cannot be achieved with the product or that does not contribute to achieving the task goal.
PUZ	puzzled	The user indicates: Not knowing how to perform the task or what function is needed for it. Not being sure whether a specific function is needed or not.
RAN	random actions	The user indicates that the current action(s) are chosen randomly.
SEA	searches for function	User indicates: Not being able to locate a specific function. Searching for a function which the analyst knows does not exist.
DIF	execution difficulty	User indicates: Having physical problems in executing an action. That executing the action is difficult or uncomfortable.
DSF	doubt, surprise, frustration	The user indicates: Not being sure whether an action was executed properly. Not understanding an action's effect. Being surprised by an action's effect. The effect of an action was unsatisfactory or frustrating.
REC	recognition of error or misunderstanding	The user indicates: Recognizing a preceding error. Understanding something previously not understood.
QUIT	quit task	User indicates recognizing that the current task was not finished successfully but continues with a subsequent task.

APPENDIX B
Definition of Proposed Breakdown Indication Types

Code	Short Description	Definition
	Breakdown Indication Types Based on Observed Actions With the Game	
ACT	wrong action	An action does not belong in the correct sequence of actions. An action is omitted from the sequence. An action within a sequence is replaced by another action. Actions within the sequence are performed in reversed order.
EXE	execution/motor skill problem	The user has physical problems interacting correctly and in a timely manner with the system.
PAS	passive	The user stops playing and does not move the mouse for more than 5 sec when an action is expected.
IMP	impatience	The user shows impatience by clicking repeatedly on objects that respond slowly, or the user expresses impatience verbally.
STP	subgame stopped	The user stops the subgame before reaching the goal.
	Breakdown Indication Types Based on Verbal Utterances or Nonverbal Behavior	
WGO	wrong goal	The user formulates a goal that cannot be achieved in the game.
WEX	wrong explanation	The user gives an explanation of something that has happened in the game, but this explanation is not correct.
DSF	doubt, surprise, frustration	The user indicates: Not being sure whether an action was executed properly. Not understanding an action's effect. The effect of an action was unsatisfactory or frustrating. Having physical problems in executing an action. That executing the action is difficult or uncomfortable.
PUZ	puzzled	The user indicates: Not knowing how to proceed. Not being able to locate a specific function.
REC	recognition	Recognition of error or misunderstanding: The user indicates recognizing a preceding error or misunderstanding.
PER	perception problem	The user indicates not being able to hear or see something clearly.
BOR	bored	The user verbally indicates being bored. The user nonverbally indicates being bored by sighing or yawning.
RAN	random actions	The user performs random actions, indicated verbally or nonverbally.
HLP	help	The user cannot proceed without help and either asks for it or the researcher has to intervene in order to prevent serious problems.
DIS	dislike	The user verbally indicates disliking something.

(Manuscript received October 1, 2005;
revision accepted for publication March 6, 2006.)