Baker-Rodrigo Observation Method Protocol (BROMP) 1.0 Training Manual version 1.0 Oct. 17, 2012

Jaclyn Ocumpaugh, Worcester Polytechnic Institute
Ryan S.J.d. Baker, Columbia University Teachers College
Ma. Mercedes T. Rodrigo, Ateneo de Manila University

History/Purpose of this Method:

The protocol for Quantitative Field Observations (QFOs) discussed here, termed the Baker-Rodrigo Observation Method Protocol (BROMP) 1.0, were initially developed by Ryan Baker and Ma. Mercedes T. Rodrigo . This method is used to record observations of student behavior and/or affect in field settings. Typically, these observations are then synchronized to log-file data compiled by the educational software that students are using at the time of the observations (although they have not been synchronized in all research using these methods). EDM researchers can then use data mining algorithms to determine what patterns in the software interactions correlate with the observational categories noted in the field. In this way, we can build detectors for the educational software that predict when students are in need of intervention because they are bored, frustrated, confused, off-task, etc.

The first publication to use these QFO protocols was Baker, Corbett, Koedinger, & Wagner (2004), which looked at student behavior (e.g., was the student on-task working alone, on-task but participating in conversation, off-task, or gaming the system). The coding schemes for affect (initially developed by Graesser and his colleagues, cf. D'Mello, Graesser, & Picard, 2007) were incorporated later (first in Rodrigo et al., 2007; the best description of this work is found in Baker, D'Mello, Rodrigo, and Graesser, 2010). In this method, behavior and affect are coded separately, because they are at least partially orthogonal (e.g., a student could be gaming the system and bored, or gaming the system and frustrated). Any behavior or affect not fitting in the scheme is noted as being outside the scheme (for instance, we do not typically include "surprise" or "delight" in our coding scheme, but they are occasionally seen, particularly in software environments with a more game-like design).

The observers base their judgment of a student's state or behavior on the student's work context, actions, utterances, facial expressions, body language, and interactions with teachers or fellow students, in line with Planalp et al.'s (1996) descriptive research on how humans generally identify affect, using multiple cues in concert for maximum accuracy rather than attempting to select individual cues. In our experience, attempting to focus on specific cues substantially reduces inter-rater reliability, as compared to more holistic judgment of behavior and affect.

Previous research has produced a number of successful models based on these observations. In terms of behavior, these methods have been used to develop and validate models that can infer gaming the system (Baker, Corbett, & Koedinger, 2004; Baker, Corbett, Roll, & Koedinger, 2008) and off-task behavior (Baker, 2007). In terms of affect, these methods have been used to develop and validate models that can infer confusion, boredom, frustration, and engaged concentration (Baker et al., 2012, San Pedro et al., under review).

These methods have also been used to study affect and the contexts and fashions in which affect emerges and changes over time (cf. Rodrigo et al., 2007, 2008a, 2008b; Baker, Rodrigo, &

Xolocotzin, 2007; Baker, D'Mello, Rodrigo, & Graesser, 2010; Baker, Moore, et al., 2011; San Pedro et al., 2011; Hershkovitz et al., 2012; Rodrigo, Baker, & Rossi, in press).

These methods have been used to study and model affect in a range of computer-based learning environments, including Aplusix, The Incredible Machine, ASSISTments, EcoMUVE, Cognitive Tutor Algebra, Cognitive Tutor Geometry, Ecolab/M-Ecolab, the Middle School Mathematics Tutor/Scooter the Tutor, Science ASSISTments, the SQL-Tutor, BlueJ, Dr. Racket, and the Chemistry Virtual Laboratory. The methods have been used on students from kindergarteners to undergraduate populations. The methods have mostly been used to study learners using technology, but are now being used in other contexts as well, including the study of kindergartners learning in group activities (cf. Fisher et al., in preparation).

Human Affect Recording Tool (HART) for Android:

We have developed an Android application, named HART, the Human Affect Recording Tool, to facilitate researchers in conducting QFOs according to the BROMP 1.0 protocol. This application is discussed in detail in (Baker et al., 2012). It synchronizes your observations to internet time, so that your data can be precisely synchronized to the log file data from the educational software being studied.

The HART application will ask you, as the coder, to input data about the school and classroom you are observing, then ask you to enter student IDs for the students you are observing. Once you have done that, it will present those student IDs to you in the order that you entered them, asking you to code behavior and affect for each student until you tell it that you are finished. It automatically saves this data for you, and you can email it to a pre-designated address when you're done. (See "HART Quick Start" section, below, for more details on this process.)

HART Quick Start:

This section will provide you with a basic understanding of the mechanics of HART, the android application developed to implement the BROMP 1.0 procedure for QFOs.

1. **Always** check to make sure that you aren't in airplane mode before you start observing. If you are, the observations won't sync to internet time. (You will receive a warning from the software.)
2. **Open** the HART program on the phone
3. You will be asked to provide USER information (that's you), as well as information about the software, the school, and the classroom that you're observing.
4. The **password** for HART is "maria" (in all lowercase letters).
5. You'll need to **tell the program how many students** you are going to code in that class session **before you can start entering student IDs or pseudonyms.** Currently, **you cannot add or subtract** from that number without starting over, which can complicate things if students are late or leave early. (Or, if you enter in their user-IDs before they come into the classroom to save time.)
6. You will also be asked to **tell the program which coding scheme** you will be using. Currently there are several pre-programmed into the application, but it is possible for a programmer to customize these to fit new needs. (Customization is NOT something that

you can do when already in the field, however. You should stick with the coding scheme that your team has decided upon ahead of time.)

7. After you have provided a student count, you will need to **enter in the user-IDs for each student**. Usually, this refers to the login ID that students use to access the software you are observing. Usually, this means that you have to go around the room and ask each student what their ID is at the beginning of the observation session. You will need to collect the login information in the order that you're going to observe these students.
8. Once you have entered in all of the students' login information, HART will ask you if you're ready to **start recording.** (Yes!)
9. When you start recording, HART will present the login ID of the first student you entered. You should **enter the behavior and affect** in accordance with the training you have on that coding scheme, then hit "ok" to move on to the next student. HART will automatically present students in the order that you entered them in at the beginning of the session.
10. **Saving:** When you are done, the program will ask you if you are sure, and then it will email the data to the email address you have programmed in. If something malfunctions with the email process (or if the program crashes before this happens), your data will still be saved on the phone. You'll just have to retrieve it manually (via a USB connection).

Observational Tips for Preventing and Correcting Miscoding:

- Seating charts aren't necessary, but they can facilitate the coding process.
  - Having a seating chart as a reference allows a coder to cross-out students who leave, come in late (and therefore weren't entered into HART at the beginning of the session), switch seats, complete the software task for the day, etc. It also allows the coder to make other notes as necessary.
  - Obtaining a seating chart from a teacher ahead of time can speed the process of inputting student logins considerably, though it is important to confirm that no student is absent and that nobody has changed seats. It is also important to be cautious to verify whether the teacher's seating chart accurately matches the classroom layout.
- During coding, errors may occur, where you accidentally code something different than what you intended to code. If you realize you have miscoded a student, you should note the observation number and the corrected code, for later correction of the data file. The observation number can be seen at the top of the screen, next to the student ID.
- If many coding errors are occurring, necessitating correction, it is useful to slow down when entering the data. It is important to enter data promptly, for log synchronization, but not at the cost of entering incorrect data.

Physical Presence in the Classroom:

- Try to be as unobtrusive as possible. Kids are naturally curious, but they're also used to being watched. If you don't draw attention to yourself (and you're not staring at them), they'll probably forget you're there.
- Boring people attract very little attention. The best way to be boring is to look bored yourself.
- In general, it's a good idea to stand diagonally behind the student you're coding. Typically we advise that you stand behind the student that you just observed while making the observation of the next student. However, you may be less obtrusive when you aren't moving at all. If you can find a location in the classroom that requires minimal movement but still allows you to see what's going on with large numbers of students, that is just as good as the technique where you're moving after each observation.
- Try to notice BOTH what is going on with the student's physical presentation and with the computer screen. (If they are highly engaged with a videogame instead of the assigned educational software, we'll want to make sure that they are coded as "off task.") However, if you can't get all three, face and body-language generally more critical than the computer screen.
- If a student asks you what you're doing, a good answer that is truthful and non-problematic is "we're looking at how students are responding to the software." This may prompt them to provide you with complaints about the software. It's probably a good idea to validate their concerns with an apology even if you're not affiliated with the software developers (but make it clear that you are not the developer of the software yourself). If it seems like a particularly important problem (e.g., inability to login), make sure that they've notified their teacher, but keep your interaction with the student as short as possible so that you can deflect attention away from you.
- It may be a good idea to tell teachers what your cover story is before you get to the classroom. This way they don't introduce you as the person watching whether or not a student is behaving! At the very least, it's good to warn them that you're trying to be very unobtrusive. One field coder used to tell teachers ahead of time that "If I'm doing it right, the students will think I'm staring angrily at my phone the whole time."
- If a student notices that they are being coded, it is best to abandon that observation (marking it "?", "?" as discussed below). Occasionally, one student will become sensitized to the observations and will repeatedly look at the observer to see if they are being watched. In this case, it is usually best to drop the student from observation for the rest of the session (e.g. always marking the student as "?", "?").
- During field observations, it can be useful to have a third person (non-observer) to entertain questions and inquiries, if possible. This enables the observers to focus solely on the observations task. Having the observer answer questions while they are observing is distracting.

Typical Coding Schemes (can be modified):

We use a dual coding scheme for this method, coding behavior separately from affect. There are several coding scheme choices to choose from when you start a session, and it is possible for a programmer to customize a new coding scheme that can be installed on the phone.

Once you have selected this scheme on the phone, you will be automatically presented with ONLY the behaviors and ONLY the affects included in that coding scheme. These may include:

1. Behavior:
    a. On task
    b. On task conversation (with either teacher or peer)
    c. Off task
    d. $ (= "gaming the system")
    e. ? (= "other")
2. Affect:
    a. Engaged concentration
    b. Confusion
    c. Boredom
    d. Frustration
    e. Delight
    f. ? (= "other")

Note that "?", "?" typically means a student who could not be coded, either because they were not present, or because they noticed the field observation taking place.

Notes on ambiguous behavior or affect:

You will undoubtedly encounter numerous instances during your observations where a student may be doing more than one thing at a time. In general, you should use your best judgment to determine the predominant behavior/affect that the student is presenting OR you should code a "?" (our "catch-all" category for anything that doesn't fit our coding scheme. Behavior and affect are cultural constructs that we learn to identify as we grow up in that culture. We use clues like facial expressions, body language, vocal expression, and other contextual clues to make holistic decisions about what other people are doing and thinking all day. Most people are competent at doing this, but it is important to be consistent in labeling. This section is meant to give you some guidelines for common questions that new coders have.

Behavior coding is generally pretty straightforward. However, there are a few instances that might be worth discussing. For example:

1. A student is waiting with their hand in the air. They aren't working on their assignment within the software, but that appears to be because they are legitimately waiting for the teacher. This should probably be coded as on-task, even though they are not engaged with the software at the time. Of course, you should use your best judgment. If a student seems to be manipulating the teacher to avoid work, you should code that student as off-task. If you are not

2. Similarly, if a student is doing scratch work on paper, they may not appear to be engaged with the software. However, if they are using that scratch work in a manner that is consistent with the learning task, you should still code them as on-task.
3. Or, you may see two students who are working but sporadically discussing last night's television show. This is definitely a case where you should use your best judgment to determine which behavior is predominant. Remember to consider things like posture, body language, and facial presentations (e.g., are they mainly looking at their own computer screens, or is there a lot of turning to face each other during the conversation). If the student you are coding appears to be mostly concentrated on the educational task, you can code that student as on task. If, however, that student is more oriented towards their classmate, you should probably code them as off-task.
4. A student first appears to be off-task because he or she is staring into space, but then you notice that he or she is responding to a teacher or another classmate who is asking them an on-task question. You should consider coding the student as on-task if their responses indicate that they were paying attention the whole time.

When in doubt, always use the "?" option.

Affect is somewhat trickier to code, since you are more likely to see overlap with it than you are with the behavior categories that we use. For example, you are more likely to see a student who is confused become frustrated than you are to see a student who is both on- and off-task at the same time (although note #3 above!). In cases such as these, you should choose the affect which appears predominant (and if one clearly precedes the other, choose the one which occurs first). If a student appears more focused than confused, code "engaged concentration." If a student's appears confused and frustrated, and the frustration is strong (e.g. a temper tantrum), code frustration. However, if you're not sure, what the student's affect is, use the "?" option.

One reason that many novice coders struggle is that they lack confidence in making these distinctions. Boredom is often obvious, but other affect categories sometimes show considerable overlap in facial expressions and posture. Indeed, a student who is experiencing "engaged concentration" may have a facial expression (a "scowl") that is quite difficult to distinguish between confusion and/or frustration at first glance. If you are not sure, take a few extra seconds to make your decision. You should still code the first affect that you see; if their initial affect is uncertain but then clearly changes (e.g. you're not certain if they are confused or bored, but then an event happens across the room which delights them), you should use the "?" code. However, given a couple of extra seconds you might see sighs, fist pounding, or just general persistence that will help you to determine which affect the student is displaying.

In cases where affects may overlap you should use your best judgment and code the affect that is predominant. There are other times, where it may seem like you have too little evidence to make a decision. If you're really not sure, you should code a "?" for that student's affect, but often you just need to pause and give yourself a few more seconds to make a decision. (Learning to do this can be challenging for novice coders.) **To summarize, here are a few examples that might cause a novice coder to hesitate or question his/her own judgment about a student's affect:**

1. A student who is experiencing "engaged concentration" MAY scowl. Particularly if facial expression is your primary clue for this student's affect, you should take a few seconds to observe the student before jumping straight to deciding they are confused or frustrated.

2. Confusion and frustration may occur at the same time, since one may trigger the other. If you see this, code the one that seems most prominent.
3. Yawning is a good indication of boredom, but it may just mean that the student is tired. Look for what they do before and after yawning.
4. If a teacher is answering questions that indicate a student does not understand part of the material, it might be evidence that the student is confused.
5. A student may be tapping rhythmically because he or she is engaged in concentration. Another student may be doing this because he or she is bored and off task. Use other contextual clues when interpreting such behavior.
6. Sometimes, especially with female students, you may not be able to see a student's face because his or her hair is covering it. If the rest of the body language indicates that they are appropriately engaged in the material, you can code them as 'concentrating,' but if you're in doubt, use the '?' code.

Field Training Process:

We have a relatively short field training process that ensures that field observers are consistently coding using the BROMP protocol for QFOs. In addition to the information presented here, we provide an initial training session outside of the field before taking each coder into the field to practice on students who are using the software. After a couple of practice rounds, we run statistical checks to determine whether or not the novice coder is consistently understanding the protocol and the labeling system. This section provides an overview of the process we use to certify new trainers.

In the initial training session, novice coders are introduced to the general concepts that the field observations are designed to address. (See History/Overview section, above.) They are also given an opportunity to interact with the HART, the android application that we use to record data. Short videos are played, and discussions of the coding scheme, including ambiguous examples, take place so that novice coders have the opportunity to ask questions about the labeling scheme.

Once we enter the field, a novice coder is paired with an experienced, certified trainer. During the first observation session (usually one class), the novice and the trainer practice entering information into HART, and then quietly discuss each student in the order that HART presents them. In this way, the novice has further opportunities to familiarize himself/herself with HART and an opportunity to ask more questions about the labeling scheme. Once the novice appears comfortable with these two issues, the certification process can begin.

In order to be certified, we have to know that the novice not only understands the process but is coding consistently with the trainer. In sessions where we test this, the novice and the trainer will work together to signal which student is being coded. Since the goal is to code the first affect/behavior witnessed, this usually requires some sort of hand signal or countdown so that both coders are beginning their observation at the exact same time. Typically, we like to have at least 60 observations before we check for consistency between the novice and the trainer. In order to do so, we calculate for Kappa (rather than for accuracy), which takes into account the base rate and the number of categories available in the coding scheme. Sixty observations can usually be collected in one class period, but it often takes more than one round of this process before a novice coder can be certified.

Acknowledgements:

This report was prepared in part with funding from the National Science Foundation through the Pittsburgh Science of Learning Center, Award # SBE-0836012, and the Bill and Melinda Gates Foundation, Award #OPP1048577. We would also like to thank the colleagues who have refined the method by learning and using it.

References:

Baker, R.S.J.d. (2007) Modeling and Understanding Students' Off-Task Behavior in Intelligent Tutoring Systems. *Proceedings of ACM CHI 2007: Computer-Human Interaction,* 1059-1068.

Baker, R.S.J.d., Gowda, S.M., Wixon, M., Kalka, J., Wagner, A.Z., Salvi, A., Aleven, V., Kusbit, G., Ocumpaugh, J., Rossi, L. (2012) Sensor-free automated detection of affect in a Cognitive Tutor for Algebra. *Proceedings of the 5th International Conference on Educational Data Mining*, 126-133.

Baker, R.S., Corbett, A.T., Koedinger, K.R. (2004) Detecting Student Misuse of Intelligent Tutoring Systems. *Proceedings of the 7th International Conference on Intelligent Tutoring Systems*, 531-540.

Baker, R.S., Corbett, A.T., Koedinger, K.R., Wagner, A.Z. (2004) Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game The System". *Proceedings of ACM CHI 2004: Computer-Human Interaction*, 383-390.

Baker, R.S.J.d., Corbett, A.T., Roll, I., Koedinger, K.R. (2008) Developing a Generalizable Detector of When Students Game the System. *User Modeling and User-Adapted Interaction*, 18, 3, 287-314.

Baker, R.S.J.d., D'Mello, S.K., Rodrigo, M.M.T., Graesser, A.C. (2010) Better to Be Frustrated than Bored: The Incidence, Persistence, and Impact of Learners' Cognitive-Affective States during Interactions with Three Different Computer-Based Learning Environments. *International Journal of Human-Computer Studies,* 68 (4), 223-241.

Baker, R.S.J.d., Moore, G., Wagner, A., Kalka, J., Karabinos, M., Ashe, C., Yaron, D. (2011) The Dynamics Between Student Affect and Behavior Occuring Outside of Educational Software. *Proceedings of the 4th bi-annual International Conference on Affective Computing and Intelligent Interaction*.

Baker, R.S.J.d., Rodrigo, M.M.T., Xolocotzin, U.E. (2007) The Dynamics of Affective Transitions in Simulation Problem-Solving Environments. *Proceedings of the Second International Conference on Affective Computing and Intelligent Interaction*.

D'Mello, S.K., Graesser, A., Picard, R.W. (2007) Toward an affect-sensitive AutoTutor. *IEEE Intelligent Systems, 22* (4), 53-61.

Hershkovitz, A., Baker, R.S.J.d., Gobert, J., Nakama, A. (2012) A Data-driven Path Model of Student Attributes, Affect, and Engagement in a Computer-based Science Inquiry Microworld. *Proceedings of the International Conference on the Learning Sciences*.

Planalp, S., DeFrancisco, V.L., Rutherford, D. (1996) Varieties of Cues to Emotion in Naturally Occurring Settings. *Cognition and Emotion, 10* (2), 137-153.

Rodrigo, M.M.T., Baker, R.S.J.d., Lagud, M.C.V., Lim, S.A.L., Macapanpan, A.F., Pascua, S.A.M.S., Santillano, J.Q., Sevilla, L.R.S., Sugay, J.O., Tep, S., Viehland, N.J.B. (2007) Affect and Usage Choices in Simulation Problem Solving Environments. *Proceedings of Artificial Intelligence in Education 2007*, 145-152.

Rodrigo, M.M.T., Baker, R.S.J.d., d'Mello, S., Gonzalez, M.C.T., Lagud, M.C.V., Lim, S.A.L., Macapanpan, A.F., Pascua, S.A.M.S., Santillano, J.Q., Sugay, J.O., Tep, S., Viehland, N.J.B. (2008) Comparing Learners' Affect While Using an Intelligent Tutoring Systems and a Simulation Problem Solving Game. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, 40-49.

Rodrigo, M.M.T., Rebolledo-Mendez, G., Baker, R.S.J.d., du Boulay, B., Sugay, J.O., Lim, S.A.L., Espejo-Lahoz, M.B., Luckin, R. (2008) The Effects of Motivational Modeling on Affect in an Intelligent Tutoring System. *Proceedings of International Conference on Computers in Education*, 57-64.

Rodrigo, M.M.T., Baker, R.S.J.d., Rossi, L. (in press) Student Off-Task Behavior in Computer-Based Learning in the Philippines: Comparison to Prior Research in the USA. To appear in *Teachers College Record*.

San Pedro, M.O.C., Baker, R.S.J.d., Rodrigo, M.M. (2011) The Relationship between Carelessness and Affect in a Cognitive Tutor. *Proceedings of the 4th bi-annual International Conference on Affective Computing and Intelligent Interaction*.