Appendix A

Figure 1. Criteria for evaluating methods.

Figure 2. Criteria for evaluating method implementation.

Figure 3. Criteria for evaluating experiment design.

Figure 4. Criteria for evaluating what the experiments told us.

1. How is the method an improvement over existing technologies?
    - Does it account for more situations? (input)
    - Does it produce a wider variety of desired behaviors? (output)
    - Is the method expected to be more efficient? (space, solution time, development time, etc.)
    - Does it hold more promise for further development? (for example, due to the opening up of a new paradigm)
2. Is there a recognized metric for evaluating the performance of your methods? (e.g., normative, cognitively valid, etc.)
3. Does it rely on other methods? (Does it require input in a particular form or preprocessed? Does it require access to a certain type of knowledge base or routines?)
4. What are the underlying assumptions? (known limitations, scope of expected input, scope of desired output, expected performance criteria, etc.)
5. What is the scope of the method?
    - How extendible is it? Will it easily scale up to a larger knowledge base?
    - Does it address exactly the task? portions of the task? a class of tasks?
    - Could it, or parts of it, be applied to other problems?
    - Does it transfer to more complicated problems? (perhaps more knowledge intensive or more/less constrained or with more complex interactions)
6. When it cannot provide a good solution, does it do nothing or does it provide bad solutions or does it provide the best solution given the available resources?
7. How well is the method understood?
    - Why does it work?
    - Under what circumstances, won't it work?
    - Are the limitations of the method inherent or simply not yet addressed?
    - Have the design decisions been justified?
8. What is the relationship between the problem and the methods? Why does it work for this task?

Figure 1. Criteria for evaluating methods.

1. How demonstrative is the program?
    - Can we evaluate its external behavior?
    - How transparent is it? Can we evaluate its internal behavior?
    - Can the class of capabilities necessary for the task be demonstrated by a well-defined set of test cases?
    - How many test cases does it demonstrate?
2. Is it specially tuned for a particular example?
3. How well does the program implement the methods?
    - Can you determine the program's limitations?
    - Have parts been left out or kludged? Why and to what effect?
    - Has implementation forced a more detailed definition or even re-evaluation of the method? How was that accomplished?
4. Is the program's performance predictable?

Figure 2. Criteria for evaluating method implementation.

1. How many examples can be demonstrated?
    - Are they qualitatively different?
    - Do these examples illustrate all the capabilities that are claimed? Do they illustrate limiations?
    - Is the number of examples sufficient to justify the inductive generalizations?
2. Should program performance be ocmpared to some standard? its tuned performance? other programs? people (cognitive validity)? experts and novices (expert performance)? normative behavior? outcomes? (either from the real world or from simulations)
3. What are the criteria for good performance? Who defines the criteria?
4. If the program purports to be general (domain-independent),
    - Can it be tested on several domains?
    - Are the domains qualitatively different?
    - Do they represent the class of domains?
    - Should performance in the initial domain be compared to performance in other domains? (Do you expect that the program is tuned to perform best in domain(s) used for debugging?)
    - Is the set of domains sufficient to justify inductive generalization?
5. If a series of related programs is being evaluated,
    - Can you determine how differences in the programs are manifested as differences in behavior?
    - If the method was implemented differently in each program in the series, how were these differences related to the generalizations?
    - Were difficulties encountered in implementing the method in other programs?

Figure 3. Criteria for evaluating experiment design.

1. How did program performance compare to its selected standard? (e.g., other programs, people, normative behavior, etc.)
2. Is the program's performance different from predictions of how the method should perform?
3. How efficient is the program? time/space? knowledge requirements?
4. Did the program demonstrate good performance?
5. Did you learn what you wanted from the program and experiments?
6. Is it easy for the intended users to understand?
7. Can you define the program's performance limitations?
8. Do you understand why the program works or doesn't work?
    - What is the impact of changing the program even slightly?
    - Does it perform as expected on examples not used for debugging?
    - Can the effect of different control strategies be determined?
    - How does the program respond if input is rearranged, more noisy, or missing?
    - What is the relationship between characteristics of the test problems and performance (either external or internal if program traces are available?
    - Can the understanding of this program be generalized to the method? to characteristics of the method? to a larger class of tasks?

Figure 4. Criteria for evaluating what the experiments told us.