Dimensions of Concern: A Method to Use Cognitive Dimensions to Evaluate Interfaces

MARK A. COHEN, Massachusetts College of Liberal Arts
FRANK E. RITTER and STEVEN R. HAYNES, The Pennsylvania State University

Producing useful and usable software often requires continuous and iterative evaluation. This paper introduces a novel usability evaluation method based on the Cognitive Dimensions of Notations framework. The target of our evaluation is Herbal a suite of tools designed to simplify agent development by providing a high-level language and maintenance-oriented development environment. The method introduced here uncovers dimensions of concern, which are used to measure the usability of Herbal and to identify areas for improvement in the design. In this article, we demonstrate how we used dimensions of concern to effectively evaluate and improve usability, and we discuss ways in which our method can be adapted, extended, and applied to improving the usability of other interactive systems.

Categories and Subject Descriptors: H.1.2 [**Models and Principles**]: User/Machine Systems—*Human information processing, software psychology, human factors*

General Terms: Human Factors, Measurement, Design, Theory

Additional Key Words and Phrases: Cognitive dimensions, measuring usability, notational systems, evaluating user interfaces

ACM Reference Format:
Cohen, M. A., Ritter, F. E., and Haynes, S. R. 2012. Dimensions of concern: A method to use cognitive dimensions to evaluate interfaces. ACM Trans. Comput.-Hum. Interact. 19, 2, Article 9 (July 2012), 18 pages.
DOI = 10.1145/2240156.2240157 http://doi.acm.org/10.1145/2240156.2240157

1. INTRODUCTION

Producing useful and usable software requires continuous and iterative evaluation [Pew and Mavor 2007]. This article introduces a novel usability evaluation method based on the Cognitive Dimensions of Notations framework [Blackwell and Green 2000, 2003]. The method introduced here uncovers Dimensions of Concern, which are used to measure the usability of Herbal (an example system) and to identify areas for improvement in the design. The target of our evaluation is The Herbal [Cohen et al. 2010], a suite of tools that simplifies agent development by providing a high-level language and maintenance-oriented development environment. Herbal was developed to provide cognitive modelers with a modern and usable development environment. In this article, we describe how we used dimensions of concern to effectively evaluate

Authors' addresses: M. A. Cohen, Department of Computer Science, Massachusetts College of Liberal Arts, North Adams, MA; email: Mark.Cohen@mcla.edu; F. E. Ritter, College of Information Sciences and Technology, Pennsylvania State University, University Park, PA; email: frank.ritter@psu.edu; S. R. Haynes, College of Information Sciences and Technology, Pennsylvania State University, University Park, PA; email: shaynes@ist.psu.edu.

DOI 10.1145/2240156.2240157 http://doi.acm.org/10.1145/2240156.2240157

and improve usability, and discuss ways in which this method of using cognitive dimensions to evaluate user interfaces can be improved.

1.1 Cognitive Dimensions

Blackwell and Green [2003] define a notational system as a system consisting of a programming language, development environment, and a medium of interaction. According to Blackwell and Green, the problem with notational systems is that "every notation highlights some kinds of information, at the cost of obscuring other kinds" [2003, p. 104]. Like any abstraction, notational systems are rich in trade-offs that constrain their design. To manage these trade-offs, Blackwell and Green introduced Cognitive Dimensions (CDs).

The primary benefit of CDs is that they provide a common vocabulary that designers can use informally to discuss the trade-offs resulting from decisions made while designing notational systems [Blackwell and Green 2003]. This vocabulary can be shared and extended across development teams and projects. In addition, CDs allow for the use of the same vocabulary during both design and evaluation. This ensures that design goals are met and facilitates coummnication between designers, evaluators, and users of the system.

Examples of CDs include closeness of mapping, which is a dimension that measures how closely a notational system maps to the result it represents, and viscosity, which measures how easily a high-level language and environment allow for change (e.g., a language that makes it easy for developers to perform maintenance would have low Viscosity). Additional examples of cognitive dimensions are shown in Table I.

In addition to providing an informal discussion tool, CDs have also been employed as a questionnaire-based evaluation tool that allows actual users, rather than designers, to evaluate the usability of a notational system. By using CDs to structure questionnaires, designers have been able obtain detailed feedback about a notational system using the very same vocabulary that was used during the design of the system. Developers can use this feedback to inform design changes. In this way, CDs can act as both formative design principles, and summative evaluation criteria.

Kadoda et al. [1999] were among the first to use CDs for usability evaluation. Using only the dimensions deemed relevant to their system, Kadoda et al. presented users with a questionnaire that paraphrased the dimensions in terms of the system under consideration. By creating a questionnaire using only the dimensions that are directly related to the system, it can be shorter and easier for users to understand.

Blackwell and Green [2000] took this evaluation technique a step further by creating a questionnaire that presented all of the dimensions, leaving it to the user to decide which ones were relevant. This approach avoids potential bias introduced by the designers of the system as they decide which dimensions are important. In addition, this approach results in a more general questionnaire that does not have to be recreated for each notational system one might wish to evaluate. However, the approach used by Blackwell and Green [2000] can result in longer questionnaires, and may be harder for the users to understand because it has not been tailored to the system being evaluated.

The evaluation method presented here more closely follows the work done by Kadoda et al. [1999] because we have chosen to use only the dimensions that we believe are relevant to the technology we are evaluating. We chose this method because we felt that the need to keep the questionnaire short and easy to understand outweighed concerns about bias.

When using CDs as a tool for evaluating usability, it is important to gather the reasons behind the users' responses. For example, discovering that a notational system has scored poorly in Closeness of Mapping may be interesting, but this information

Table I. Examples of Cognitive Dimensions [Blackwell and Green 2000, 2003]

CD	Description
Visibility	How easy is it to view the elements in a model, including their internal details?
Viscosity	How easy is it to make changes to an existing model? The less the viscosity, the easier it is to change the model.
Diffuseness	How many symbols or how much space does the notation require to produce a certain result or express a meaning?
Hard-mental operations	How much hard mental processing lies at the notational level, rather than at the semantic level? Are there places where the user needs to resort to fingers or penciled annotation to keep track of what's happening?
Error-proneness	How easy is it to make errors using the behavior representation language?
Closeness of mapping	How closely does the behavior representation language match the way that the modeler describes the behavior?
Role-expressiveness	How easy is it to discover why a modeler has chosen a particular design? Explicit support for design rationale improves a systems role-expressiveness.
Progressive evaluation	How easy is it to evaluate and obtain feedback on an incomplete solution?
Premature commitment	How often is the developer forced to make a commitment in the model before there is enough information to make the commitment?

alone cannot properly inform design unless it also reveals the reasons underlying the problem.

The evaluation method presented here is novel because it measures the support of CDs by combining scaled survey questions with open-ended questions and the use of participant observation. The inclusion of open-ended survey questions and participant observation helps the designer assemble the detailed explanations required to inform future design. In addition, we use a novel analysis that identifies the CDs that demand the designer's attention. We call these CDs *Dimensions of Concern*.

The target of the usability study presented here is Herbal. Herbal is an example of a notational system because it consists of a high-level behavior representation language, an agent development environment, and a medium of interaction [Blackwell and Green 2003]. As a result, it is an ideal candidate for a CDs-based evaluation and a useful example to explain our approach.

The primary focus of this article is to assess and report on the effectiveness of our evaluation method. However, to understand our method, and its strengths and weaknesses, it is necessary to provide some details about the system we will be evaluating. To this end, Section 1.2 provides background about Herbal.

1.2 Herbal

To simplify agent programming and cognitive modeling, a high-level behavior representation language and associated parser and compiler were designed and implemented as part of Herbal. This high-level language is based on the Problem Space Computational Model (PSCM) [Lehman et al. 1996; Newell et al. 1991] and is represented using the Extensible Markup Language (XML)[1]. The Herbal language is currently compiled into productions that execute within three popular agent architectures: Soar[2], ACT-R[3], and Jess[4].

An Herbal program is made up of six different types of XML documents, each defining a set of reusable components: types, conditions, actions, operators, problem spaces,

[1]www.w3.org/XML/
[2]sitemaker.umich.edu/soar
[3]act-r.psy.cmu.edu
[4]herzberg.ca.sandia.gov/jess/

Table II. Typical Herbal High-Level Behavior Representation Language Fragment

```
<operator name='driveRight'>
  <if>
    <conditionref condition='okRight'/>
  </if>
  <then>
    <actionref action='moveRight'/>
  </then>
</operator>
```

Table III. Compiling an Herbal Condition to Jess and Soar Equivalents

Destination Architecture	Code Fragment

Herbal High-Level Language	`<condition name='dirty'>`
	`  <match type='vacuum.types.spot'>`
	`  <restrict field='status'>`
	`  <eq>dirty</eq>`
	`  </restrict>`
	`  </match>`
	`</condition>`

Resulting Soar Code	`(<vacuum-types-spot2>` `  ^status <status2>  |dirty| )`

Resulting Jess Code	`(topspace::vacuum.types.spot`
	`  (status ?status1&:`
	`  (eq* ?status1 "dirty")))`

and agents. These documents are referred to as libraries in the Herbal language. Typical usage of Herbal consists of a modeler creating a model of human cognition by defining an agent and the problem spaces it operates in to perform a task. The Herbal suite of tools simplifies this process in many ways.

Table II lists a typical fragment of Herbal source code. The XML shown here declares an instance of an operator called *driveRight*. The *driveRight* operator will be proposed when the condition *okRight* is true, and when the operator is applied an action called *moveRight* will move the agent to the right. The details of the *okRight* condition and the *moveRight* action are given in additional libraries.

Code written in the Herbal high-level language (like the fragment shown in Table II) can be transformed into executable productions for various agent architectures. Table III illustrates how the Herbal compiler transforms the Herbal high-level language for a condition into productions for the Soar and Jess architectures.

Herbal also includes an Integrated Development Environment (IDE) that provides a graphical environment for creating and maintaining agents by leveraging the popular Eclipse platform (eclipse.org). The Herbal IDE allows for the creation of Herbal agents using a visual editor (see the top-left corner of Figure 1, The Herbal GUI Editor), and by programming directly in the Herbal High-Level Language editor (see the top-right corner of Figure 1). Agent programmers can freely switch between these two modes of editing, and view the resulting low-level Soar and Jess code produced because of their code changes (see the bottom two panels in Figure 1, pete.jess and pete.soar).

The Herbal IDE provides integral support for working sets [Ko et al. 2005], which allow the modeler to automatically generate a collection of library components based on keyword searches (see Figure 2). These sets can be saved and recalled as needed during model maintenance.

Herbal supports several different forms of reuse including the creation of source code libraries, instantiation of behavior design patterns, and the ability to capture the

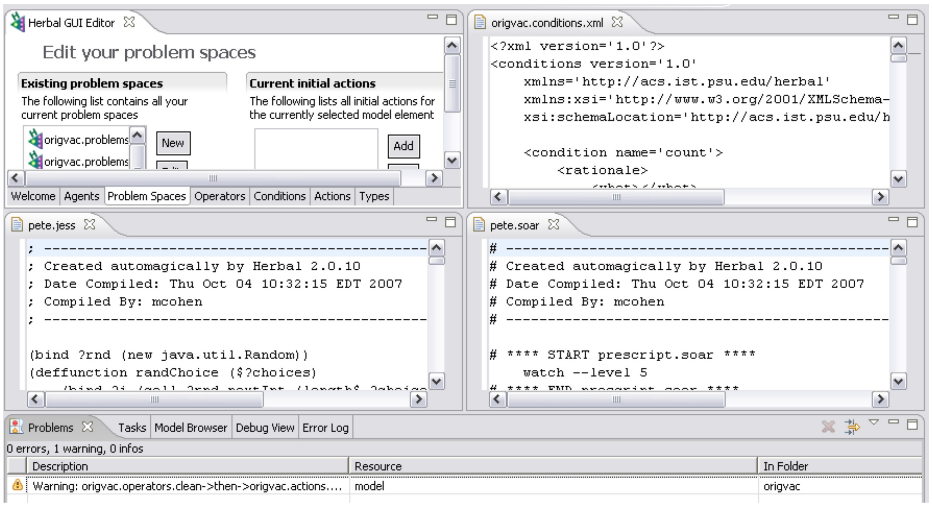

Fig. 1. The Herbal GUI editor.

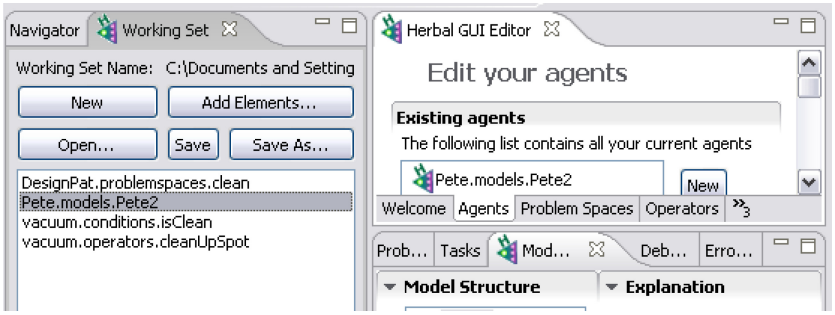

Fig. 2. Support for working sets in Herbal.

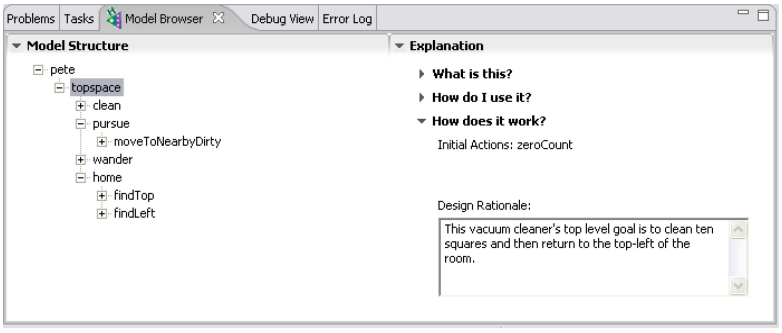

Fig. 3. Capturing design rationale in Herbal.

rationale underlying the design of each component. In Herbal, the design rational is included as part of the component's definition (see Figure 3). The capturing of design rationale in Herbal was informed by the explanation design patterns developed by Haynes et al. [2008].

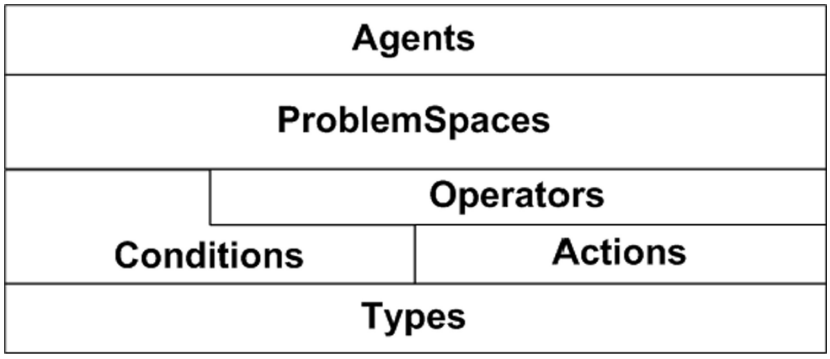

Fig. 4. Dependencies between the different Herbal library types.

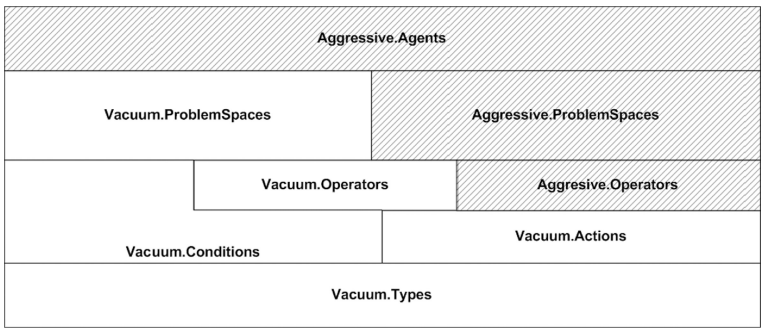

Fig. 5. Constructing agents by reusing libraries of general task behavior.

There are six different kinds of Herbal libraries: types, conditions, actions, operators, problem spaces, and agents. The dependencies between the contents of these libraries are shown in Figure 4. The foundation of all the Herbal libraries is the *types* library. This library contains the set of data types available to the agent programmer. From these types, the programmer can define conditions and actions that can add, edit, remove, or test for the existence of instances of the defined types. Operators are then built from these conditions and actions, and problem spaces are built from a set of conditions and operators that activate a problem space.

Finally, behavior for a specific agent can be defined by reusing existing libraries of general task behavior. This layered approach allows developers to specify behavior at the most appropriate level of abstraction for a given problem. For example, in Figure 5 aggressive vacuum cleaner agents (the Vacuum Cleaner World is described in more detail in Section 2) are built on top of a collection of predefined libraries designed to support all vacuum cleaners.

Reuse is also accomplished in Herbal using behavior design patterns. Structured programming paradigms such as looping constructs are useful in agent programming, but can be a challenge to program in a typical rule-based language. To address this problem and to promote the reuse of meta-behaviors such as looping, a Behavior Design Pattern Wizard was incorporated into the development environment (see Figure 6). This wizard makes it possible for the agent developer to generate instantiations of useful metabehaviors using existing PSCM components.

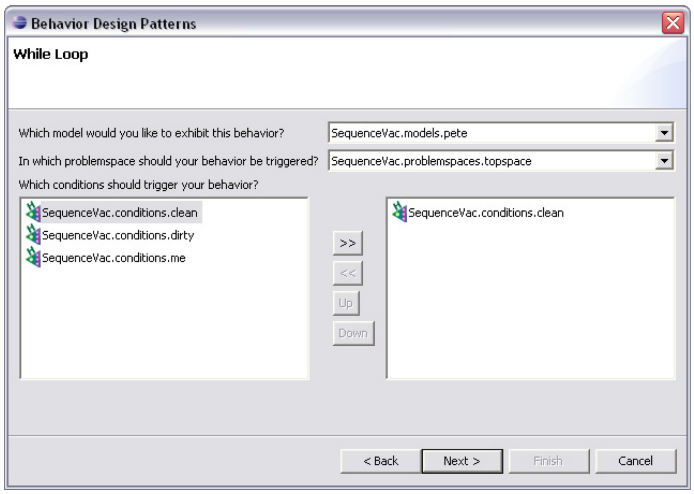

Fig. 6. The herbal behavior design pattern wizard.

Table IV. Calculating a Dimension of Concern Based on Participant Observation

A CD is a Dimension of Concern if:

$$(E_n - E_p) > 0 \text{ AND } (E_n - E_p) > (\tau * N)$$

Where:

E_n is the number of negative events recorded for the CD
E_p is the number of positive events recorded for the CD
N is the number of participants observed
τ is the Dimension of Concern threshold

2. DIMENSIONS OF CONCERN

In this article, we will report the results of a study with the main goal of identifying areas of weakness in the usability of Herbal, as well as ways to address these weaknesses. To help achieve this goal, the concept of a dimension of concern was developed. A dimension of concern is a CD that is poorly supported by the notational system. A CD is considered a dimension of concern when support for the CD by the notational system fails to meet some minimum tolerance threshold for quality.

In the method we present here, dimensions of concern are identified based on questionnaire data and participant observations. A dimension of concern is identified from questionnaire data by identifying the CDs in which the number of participants that gave a negative response related to the CD exceeded some tolerance threshold. We call this threshold the *dimension of concern threshold*.

The process we used to identify dimensions of concern from participant observations was somewhat more complicated. In this case, a dimension of concern was identified by calculating the difference between the number of participants experiencing negative events related to a CD and the number of participants experiencing positive events related to a CD. A dimension of concern arises when this calculation produces a positive score whose value is larger than the dimension of concern Threshold (see Table IV).

An important question when performing this type of analysis is how to determine the dimension of concern threshold. Not having established thresholds to measure usability is a known problem [Pew and Mavor 2007, pp. 324–328].

Blackwell and Green point out that the cognitive dimensions framework "illuminates design maneuvers in which one dimension is traded against another" [2003,

p. 118]. For example, Error proneness can be reduced by requiring users to do things in a specific order, but this can lead to poor support for premature commitment. Choosing a very low tolerance threshold may result in several conflicting Dimensions of Concern that are difficult to satisfy simultaneously. However, a very high tolerance threshold can result in few dimensions of concern but unusable software.

The threshold value decision becomes more complicated because the severity of the design trade-offs can depend on the type of users the system supports. For example, the tradeoff between error proneness and premature commitment described earlier may be less severe for expert users who make fewer mistakes and give more weight to freedom and flexibility.

The difficult decision of what threshold to use should be considered an advantage rather than a disadvantage. While difficult and subjective, the threshold value provides the evaluator with tremendous flexibility. For example, the evaluator may choose to utilize a series of usability studies that employ a spiraling tolerance value. After each study and redesign, a new study with a lower tolerance threshold can be conducted. Iterations can continue until the designers and users are satisfied or the tradeoffs between CDs become too difficult to satisfy at such a low tolerance threshold. Alternatively, experimenters could choose to use a completely different tolerance threshold for each CD. This could reflect the relative importance the designers' place on each CD and the tradeoffs that exist between them. The flexibility and adaptability provided by the dimension of concern threshold is a major strength of our method.

In general, we believe it makes sense to start with a relatively high threshold of tolerance, especially for notational systems designed for use equally by novice and expert users. The study presented here was conducted with participants with limited or no experience developing cognitive models or intelligent agents. Our concern is that a very low threshold of tolerance would lead to design recommendations that compromise the system for expert users. For this reason, we decided to begin with a tolerance threshold of 20%. Once the study results are used to redesign our system, we hope to conduct a second usability study with a slightly lower tolerance threshold and a wider variability of user types.

3. TASK OVERVIEW

To evaluate the Herbal system, users were asked to complete a set of tasks. We recorded the user's behavior and analyzed it with our method to understand Herbal usability, and how it might be improved.

The main study task was to create a working intelligent agent that operates a vacuum cleaner in the Vacuum Cleaner Environment [Cohen 2005]. The Vacuum Cleaner Environment (Figure 7) is based on a very simple world introduced in a widely used Artificial Intelligence textbook [Russell and Norvig 2003]. In the Vacuum Cleaner environment, the goal is to build a vacuum cleaner agent to clean up the dirty squares as quickly as possible. A vacuum cleaner agent can be created using a rule-based program written in one of two popular architectures: Jess and Soar.

In this study the main task is divided into three subtasks: creating a reusable library of agent components; creating a vacuum cleaner agent using this library; and finding and fixing a bug in the resulting vacuum cleaner agent. These three subtasks exercise all of the features of Herbal and closely mirror the different phases of agent/model development: creating reusable components, using these components to build agents, and debugging the resulting agents.

The first subtask exercised Herbal's library creation facilities. In this subtask, participants created a new library and populated it with low-level components needed to build vacuum cleaners. The task instructions encouraged the inclusion of design rationale in the library throughout this task.

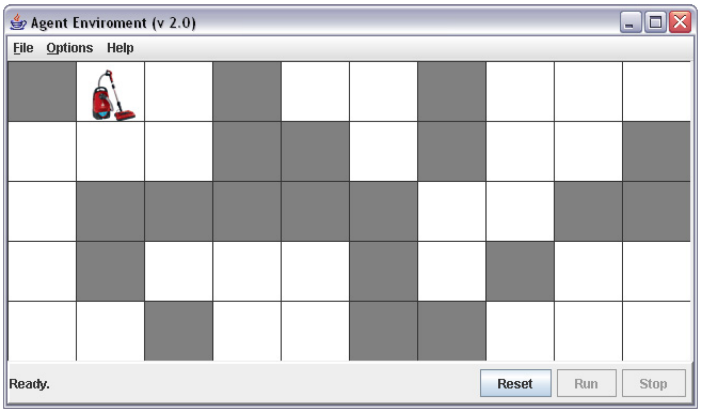

Fig. 7. The Vacuum Cleaner Environment.

The second subtask exercised Herbal's model creation facilities. This task consisted of building a vacuum cleaner agent out of higher-level model components. Participants created these higher-level components by reusing the low-level components created during the first subtask. The design rationale located in the library was available to participants to help them during the task. In addition, participants were encouraged to capture their own design rationale about the newly created components.

The third subtask exercised Herbal's model maintenance facilities. This task consisted of finding and fixing an error in the vacuum cleaner model created during the second task (the experimenter injected this error in this model before the start of the third task). During this task, participants used Herbal's debugger and working set feature to find and fix the bug in the broken vacuum cleaner. Participants were encouraged to make use of design rationale when needed.

4. THE DIMENSIONS OF CONCERN METHOD

Data were gathered during this study using a user reaction survey, and by participant observations. These data measured Herbal's support for CDs. The user reaction survey was based on an earlier CD survey done by Kadoda et al. [1999].

Table I shows the nine CDs that this study used for usability evaluation criteria. These nine dimensions were chosen because we felt they best measure the degree in which Herbal achieves the principles that motivated its design (i.e., embracing high-level languages, enabling reuse, and supporting maintenance-oriented development).

The criteria used to categorize dimensions were different depending on the type of data analyzed: survey or observation. Because of the difficulty involved in observing when a participant is experiencing Hard-Mental Operations, only survey questions measured this dimension.

4.1 Participants

The participants recruited for this study had limited or no experience developing cognitive models or intelligent agents. Participants were recruited from the population of undergraduate students majoring in Computer Science (CS), Computer Information Science (CIS), Management Information Science (MIS), and Psychology (PSYC) at Lock Haven University. These majors represent the diverse population of potential future modelers that Herbal was designed to support. Participants received $10 for taking part in this study, which required approximately an hour of their time.

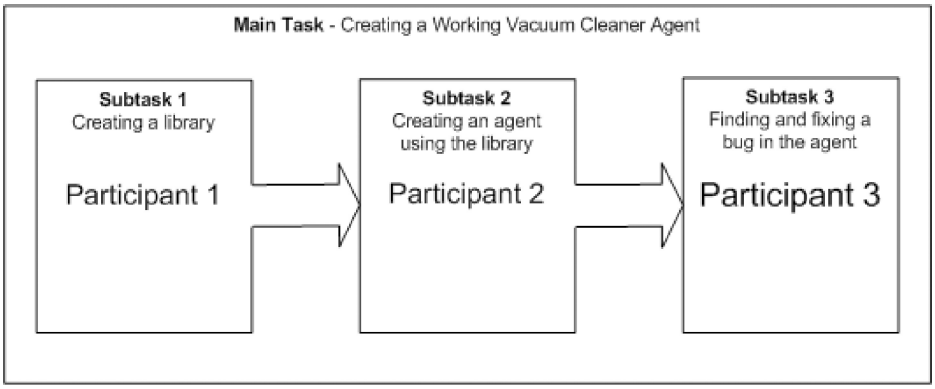

Fig. 8. Different participants work in turn to complete the main task.

Twenty-four students participated: 12 PSYC students and 12 CS/CIS/MIS students. The number of college credits completed by the participants ranged from 42 to 132. The average number of hours per week spent using a computer was 4 for PSYC majors and 10 for CS/CIS/MIS majors. The average number of programming courses previously taken was 0.25 for PSYC majors and 4.75 for CS/CIS/MIS majors. Finally, nine of the 12 PSYC majors were female and one of the CS/CIS/MIS students was female.

4.2 Apparatus

Participants used a Lenovo T60p laptop computer to perform the task. This laptop was docked in a docking station including a keyboard, a mouse, a 17-inch flat screen monitor, and a microphone.

Camtasia Studio 2.0.2 (TechSmith.com) recorded both the screen and audio while the participants perform the subtasks. Additional software that was required for this experiment inlcuded Eclipse (3.2.1), Java (1.5), Herbal (3.0), and the Vacuum Cleaner Environment (2.0).

4.3 Design

The study design placed participants into groups of three. Each group was responsible for completing a working vacuum cleaner agent. Groups contained either all PSYC majors or all CS/CIS/MIS majors. This resulted in eight groups of three participants: four groups of PSYC majors and four groups of CS, CIS, and MIS majors.

As described earlier, the main task for this study was to create a working intelligent agent that operates a vacuum cleaner in the Vacuum Cleaner Environment. This main task consists of three subtasks: creating a reusable library of agent components; creating a vacuum cleaner agent using this library; and finding and fixing a bug in the resulting vacuum cleaner agent. Each group completed the main task by having a different member finish each subtask independently and in turn, as shown in Figure 8.

The first group member was responsible for producing the library, the second group member was responsible for building an agent using the library, and the final group member was responsible for finding and fixing bugs in the agent. The group members did not work simultaneously, the successive work of participants two and three in each triad depended on the work that had occurred before. Because they relied on previous work, mistakes or decisions made by one participant in the group could influence the performance of another participant in the group.

4.4 Procedure

The study began with each participant reading and signing the consent form and completing the User Background Survey, which collected information about his or her background and expectations prior to participating in the study.

Next, each participant watched a 15-minute video before performing his or her subtask. The video provided the participant with a high-level introduction to intelligent agents, the Vacuum Cleaner Environment, the PSCM, and creating and maintaining libraries and agents in the Herbal Integrated Development Environment.

At the completion of the video, the experimenter informed the participant what subtask he or she would be performing and gave each participant the General Task Instructions. These instructions ask the participants to think-aloud during the experiment [Ericsson and Simon 1993; Newell and Simon 1972], and to ask questions if they were confused. After the participant read the General Task Instructions, the experimenter asked them to practice thinking aloud while performing a simple memory recollection exercise to practice verbal protocols.

Next, the experimenter provided each participant with specific task descriptions that provided systematic instructions about how to perform the subtask. The experimenter also started the recording of the computer monitor and audio. At this point, the participants began their subtask.

In addition to the video and audio recordings, the experimenter noted observations about the participant's performance during the execution of the task and entered observations in a data collection form. The experimenter focused on entering events related to the CDs of interest. The same experimenter ran all participants.

Upon completion of the subtask, the experimenter asked the participants to complete a User Reaction Survey. The User Reaction Survey contained 17 questions that mapped directly to the CDs shown in Table I. A five-level Likert scale structured 11 of these questions. The remaining six questions were open-ended and sought information explaining responses to the scaled questions.

5. SURVEY RESULTS

Table V lists the 11 scaled questions. Shading in this table identifies negative responses and the absence of shading identifies positive responses.

Participant responses were analyzed using six different groupings: (1) all participants; (2) participants performing the library creation task; (3) participants performing the model creation task; (4) participants performing the model maintenance task; (5) participants majoring in PSYC, (6) participants majoring in CS, CIS, or MIS. These groupings were created to help make decisions about the support for CDs based on task type and the participants' major.

Dimensions of concern were identified from questionnaire data by identifying questions in which more than 20% of the participants gave a negative response. Table VI lists dimensions of concern based on the survey results.

6. OBSERVATION RESULTS

Upon completion of the study, the experimenter coded all of the screen and audio recordings based on the CDs shown in Table I. In addition, a subset of the recordings was coded by a second experimenter. Cohen's [1960] kappa was used to determine the agreement between the two raters, and an overall agreement 0.83 was calculated. A value that is greater than 0.8 is acceptable, and shows that this set of CD features can be coded.

These coded observations complement the survey results previously presented. Unlike the survey results, information from the context of the observations helps explain

Table V. The Survey Questions Used to Measure Support for Various CDs

Shading indicates the negative response ranges.

Visibility (Q1 & Q2)	How easy was it to see or find the various parts (e.g., problem spaces, operators, conditions) of your agent or library while it was being created, changed, or debugged?				
	very easy	easy	neutral	difficult	very difficult
	If you needed to compare different parts (e.g., problem spaces, operators, conditions) of your agent or library, you could easily see these parts at the same time.				
	strongly agree	agree	neutral	disagree	strongly disagree
Viscosity (Q3)	How easy was it to make changes to your agent or library?				
	very easy	easy	neutral	difficult	very difficult
Diffuseness (Q4)	The elements (e.g., problem spaces, operators, and conditions) you used to build your agent or library allowed you to say what you wanted to say reasonably briefly.				
	strongly agree	agree	neutral	disagree	strongly disagree
Hard-Mental Operations (Q5)	In general, the task you performed did not seem especially complex or difficult to work out in your head.				
	strongly agree	agree	neutral	disagree	strongly disagree
Error Proneness (Q6)	During this task, you often found yourself making small mistakes that irritated you or made you feel stupid.				
	strongly agree	agree	neutral	disagree	strongly disagree
Closeness of Mapping (Q7)	The notation (e.g., problem spaces, operators, and conditions) you used to describe your agent or library was closely related to how you might describe the agent or library naturally.				
	strongly agree	agree	neutral	disagree	strongly disagree
Role Expressiveness (Q8)	During the task, you often did not know what many of the agent or library pieces meant (e.g., problem spaces, operators, conditions) but you put them in anyway.				
	strongly agree	agree	neutral	disagree	strongly disagree
Progressive Evaluation (Q9 & Q10)	It was easy to stop in the middle of creating the agent or library, and check your work so far.				
	strongly agree	agree	neutral	disagree	strongly disagree
	During this task, it was easy to find out how much progress you made, or check what stage in your work you were in.				
	strongly agree	agree	neutral	disagree	strongly disagree
Premature Commitment (Q11)	When working on this task, there were times when you felt like you could have changed the order you performed the steps without breaking the agent or library.				
	strongly agree	agree	neutral	disagree	strongly disagree

the positive or negative participant experiences related to a dimension. This information plays an important role in helping the designers make improvements to the design after the study.

The experimenter identified 37 unique event types during observation, and these types represented either a negative or a positive contribution to a particular CD. A positive contribution to a CD means the software helped a participant in a manner that is consistent with the definition of the dimension. A negative contribution to a dimension means the software was a hindrance to the participant in a fashion consistent with the definition of the dimension.

To generate these event types, the experimenter relied on the participants' actions and utterances. Table VII lists the 37 event categories along with their associated

Table VI. Dimensions of Concern as Measured by All Survey Results

Dimension	All	Library Creation	Model Creation	Model Maintenance	PSYC	CS/CIS/MIS
Visibility	X	X	X			X
Viscosity						
Diffuseness						
Hard-mental operations				X		
Error-proneness			X	X	X	
Closeness of mapping						
Role-expressiveness			X		X	
Progressive evaluation						
Premature commitment	X	X	X	X	X	X

dimensions. It is important to keep in mind that this table shows types of events not actual instances. These event types (or codes) are useful for providing the rationale behind the classification of an observation, and for suggesting improvements to future releases of the software.

For example, while editing model components several participants became confused when specifying properties in the model properties dialog box. This was because the dialog box does not contain the name of the component the participant was editing. This problem became apparent when the participant moved the dialog box out of the way to see what component they had selected before the dialog box appeared. Category 23 (the participant became confused about what specific component they were working on) in Table VII coded this particular observation, and the suggested design change would be to add the component name to the dialog box.

Table VIII lists the Dimensions of Concern based on participant observations.

7. DISCUSSION

Table IX lists dimensions of concern based on both survey results and participant observations. The table indicates concerns based on survey results with the letter S, and concerns based on observations with the letter O. In this table, 54 possible dimension/condition pairings (nine dimensions * six conditions). Of these 54 pairings, 65% (35) show agreement between the survey results and the observations (the absence of a dimension of concern using both a survey and participant observation is considered agreement). According to the surveys and observations, Herbal appears to be strong with respect to Viscosity, Diffuseness, Hard-Mental Operations, Closeness of Mapping, and Progressive Evaluation.

Observations related to Viscosity show that participants found it easy to make changes to components. In addition, the working set feature made it easy for participants to change a collection of related components. Only during the model maintenance task was there an indication (by observation) of difficulty making changes to the model. During this task, several participants encountered problems when editing the operator that was causing the vacuum cleaner to malfunction.

Observations of appreciation for the code automatically created by Herbal provided evidence of positive support for Diffuseness. Several participants mentioned that they were very happy they did not have to generate the Soar code manually. The Design Pattern Wizard also proved to be a compact way of expressing complicated behavior. Problems with Diffuseness took place only during library creation when the participant attempted to enter design rationale. Several participants commented on the redundancy of the design rational task, relying on copy/paste to hasten design

Table VII. Event Codes Used during Participant Observation

ID	Event Description	Dimension	+/-
1	Participant appeared confused by notation used to represent agent	Closeness of mapping	-
2	Participant preferred a term not used by high-level language	Closeness of mapping	-
3	The design pattern wizard allowed participants to create several components using a brief terminology	Diffuseness	+
4	Participant was thankful for code automatically created by Herbal	Diffuseness	+
5	Participant found design rationale to be verbose and/or redundant	Diffuseness	-
6	Participant used copy and paste when entering design rationale	Diffuseness	-
7	The consistency in the Herbal interface helped reduce errors	Error prone	+
8	The design pattern wizard prevented errors creating components	Error prone	+
9	The search feature of the new project dialog lead to errors	Error prone	-
10	Participant had problems distinguishing types of design rationale	Error prone	-
11	Participant confused Eclipse export with Herbal library export	Error prone	-
12	Poor design in the working set dialog lead to errors	Error prone	-
13	Trouble locating the Herbal GUI Editor	Error prone	-
14	Instructions mislead participant to create action instead of type	Error prone	-
15	Participant selected wrong problem space in design pattern wizard	Error prone	-
16	Participant entered literal value in local variable edit box	Error prone	-
17	Participant confused by wire screen when there is nothing to wire	Error prone	-
18	Participant confused by conditions with no restrictions	Error prone	-
19	Lack of required order made it easier to fix mistakes	Premature commitment	+
20	Participant changed order of steps in task without problems	Premature commitment	+
21	Participant was confused by order required to run debugger	Premature commitment	-
22	It was easy for participants to check the status of the model, and for any errors, by looking at what has been done so far	Progressive evaluation	+
23	The participant became confused about what specific component they were working on, or what step they were doing	Progressive evaluation	-
24	Participant viewed rationale to learn more about model	Role expressiveness	+
25	Participant demonstrated strong understanding of the model	Role expressiveness	+
26	Commented that components are self-explanatory	Role expressiveness	+
27	Participant entered quality design rationale	Role expressiveness	+
28	Participant demonstrated poor understanding of the model	Role expressiveness	-
29	Participant had trouble understanding component/subcomponent	Role expressiveness	-
30	Participant viewed rationale but did not find it helpful	Role expressiveness	-
31	Participant misunderstood interface between model/environment	Role expressiveness	-
32	Behavior of agent was easy to see using the debugger	Visibility	+
33	A portion of the Herbal GUI editor was hidden	Visibility	-
34	Easy for participant to make a change to a component	Viscosity	+
35	Working sets helped participant find location of a problem	Viscosity	+
36	Participant had problems editing an action	Viscosity	-
37	Participant had problems editing an operator	Viscosity	-

rationale entry. Perhaps with a more complicated model, entering design rationale would have been a more interesting task.

The system also met the needs of the participants with respect to Progressive Evaluation. Observations confirmed that participants could easily check the progress of

Table VIII. Dimensions of Concern as Measured by Observations

Dimension	All	Library Creation	Model Creation	Model Maintenance	PSYC	CS/CIS/MIS
Visibility						
Viscosity				X		
Diffuseness		X				
Hard-mental operations						
Error-proneness	X			X		X
Closeness of mapping						
Role-expressiveness		X				
Progressive evaluation						
Premature commitment				X		

Table IX. Summary of DoCs Based on Survey Results (S) and Participant Observations (O)

Dimension	All	Library Creation	Model Creation	Model Maintenance	PSYC	CS/CIS/MIS
Visibility	S	S	S			S
Viscosity				O		
Diffuseness		O				
Hard-mental operations				S		
Error-proneness	O		S	SO	S	O
Closeness of mapping						
Role-expressiveness		O	S		S	
Progressive evaluation						
Premature commitment	S	S	S	SO	S	S

the model at any point in time, regardless of task. In addition, the participants were able to browse the model for potential errors when needed.

Closeness of mapping was another dimension that Herbal supported well. Survey results were positive with respect to the high-level language used to describe agents written in Herbal. In addition, only three participants experienced negative events with respect to closeness of mapping. Two participants thought the term "mode" or "state" would be more useful than problem space, and one participant became confused by the notation used to represent the agent and its behavior.

Finally, participants indicated in the surveys that, as a whole, they did not find the tasks particularly complex. This is due in part to the fact that the tasks tested the usability of the system, and were not problem solving exercises (aside from the maintenance task). On the surveys, participants indicated some complexity in the model maintenance task. This is not surprising because this is the one task where participants were asked to solve a problem (i.e., what was wrong with the vacuum cleaner) rather than navigate an interface.

Herbal appears to lack adequate support on two dimensions: error proneness, and premature commitment. Table IX lists these two dimensions as Dimensions of Concern in nearly every column.

The classification of Error Proneness as a concern was due to both observations and survey results. Only the library creation tasks did not suffer from problems with error proneness. Recordings of the observed events give reasons for the problems participants had with error proneness. For example, 13 errors resulted from participants having problems distinguishing between the different types of design

rational. Twenty-one errors resulted from problems with the design and layout of the Working Set, New project, and design pattern Wizard dialogs. Four errors resulted from confusion caused by the use of a condition with no restrictions. Finally, four errors took place because of a problem in the wording of one of the steps in the instructions.

Survey data in all six conditions classified premature commitment as a concern, yet participant observations only classified premature commitment as a concern during the model maintenance task. Due to a lack of useful responses to the open-ended portions of the survey, it is difficult to tell why participants felt restricted to order. Observations provided a somewhat different account with respect to Premature Commitment. Eight participants were able to fix mistakes more easily because Herbal did not enforce order. In addition, nine participants changed the order of the steps in the task, on their own, and without problems.

A task where order appeared to present problems (during observation) was the model maintenance task. Connecting the debugger to the Vacuum Cleaner Environment requires a fixed order and three participants were observed having problems with this rigid order. Because survey data indicated a problem with Premature Commitment, and observations did not, further exploration of this concern is required.

Table IX shows mixed results for role expressiveness and visibility. Three conditions classified role expressiveness as a concern, and four conditions classified visibility as a concern. Role expressiveness was a concern during library creation, where three participants demonstrated a poor understanding of the model in general, and three participants demonstrated trouble understanding the relationship between components and subcomponents. Surveys indicated a problem with role expressiveness during model creation and for PSYC students as a whole. Visibility was a concern in survey data for all conditions except model maintenance and for PSYC students as a whole. However, observations did not indicate Visibility as a concern for any condition.

8. CONCLUSIONS

The method presented here has been shown to be an effective way to measure and summarize the usability of Herbal and has helped us discover areas of strength and areas for improvement in the design.

For example, based on agreement between surveys and participant observation we determined that Herbal is strong in five of the nine dimensions: viscosity, diffuseness, progressive evaluation, closeness of mapping, and hard-mental operations. The use of abstractions such as a high-level language and graphical editor were a conscious design decision in Herbal, and this appears to have helped reduce viscosity. In addition, the variety of views provided in the Herbal Development Environment to make it easy for developers to check the status of a model, and this appears to have strengthened Herbal's support for progressive evaluation. However, it is important to put these results in the context of the dimension of concern tolerance threshold value of 20% and given the experience level of the participants.

Also based on agreement between surveys and participant observation, Error Proneness was shown to be a dimension of concern; only the library creation task did not suffer from problems with error proneness. A major strength of our dimensions of Concern analysis is its use of both questionnaire data and participant observations. There is agreement in both the surveys and observations about a dimension of concern, the observations could be used to determine the cause of the concern and help inform design change.

For example, the observations related to error proneness revealed that 13 negative events related to design rationale were the result of confusion about the difference between the three different design rational types. The explanation behind the problems reported by the users in the questionnaire would not have been discovered without

these observations. Interestingly, this issue with design rationale that our analysis uncovered went unnoticed during design because the designers were all very familiar with the design rational types.

In addition, observations of users performing the working set task revealed the reasons behind the issues with error proneness reported by users in the questionnaires. Users were observed struggling with the working set feature because the search feature is case sensitive, and because it is difficult to refine searches while building working sets. These results can be used to inform a redesign of the working set feature.

However, issues did arise when the surveys and the observations did not agree. In these cases, it was difficult to come to any decisive conclusion on how to improve the design. For example, for many tasks, the survey data indicated poor support for premature commitment, but the participant observations did not agree. The Herbal Development Environment was designed to be library-centric so that users could develop model components without constraining order. Why did the users feel constrained in the sequence that they developed the model? Unfortunately, participants failed to provide sufficient explanations in the open-ended survey questions, which made it difficult to draw conclusions about how to improve design.

It is essential that future evaluations place more emphasis on the open-ended survey questions to ensure that explanations are provided by participants. Actual application stakeholders as evaluators in future studies may help produce open-ended responses that are more useful.

As reported in our results, a large number of error proneness events were reported during the coding process. Error proneness was defined as a measure of how easy is it to make errors using the behavior representation language. However, negative events related to many of the other dimensions can also lead to errors. When coding the participant observations it was not always clear when to use an error proneness code as opposed to (or in addition to) other CD codes. We believe that it is possible participants had the same confusion when responding to the surveys. As a result, error proneness was one of the most common codes employed during coding and we believe that this may have obscured important information related to other CDs. In future evaluations we will provide better guidance on how to differentiate between error proneness in the notation, as opposed to errors caused by deficiencies in other CDs.

The evaluation method presented here more closely follows the work done by Kadoda et al. [1999] because we have chosen to use only the dimensions that we believe are relevant to the system we are evaluating. We chose this method because we felt that the need to keep the questionnaire short and easy to understand, especially considering the participants were undergraduates who were not major stakeholders in the application. However, there is legitimate concern about introducing bias when filtering the CDs presented to the users [Blackwell and Green 2000]. In future studies, if the users are stakeholders in the system being evaluated and have sufficient background and interest in the application, it may be appropriate to include questions in the questionnaire pertaining to all of the CDs.

In conclusion, the usability evaluation method presented here proved helpful for measuring the usability and identifying several areas for improvement to the design of Herbal. By identifying dimensions of concern using both survey responses and participant observation, we were able to identify design improvements that should lead to a more usable system. We believe that this method can be used successfully to evaluate the usability of a variety of notational systems, and that the dimensions of concern threshold provides researchers enough flexibility to handle the complex tradeoffs that exist between CDs in a variety of systems.

REFERENCES

BLACKWELL, A. F. AND GREEN, T. R. G. 2000. A cognitive dimensions questionnaire optimised for users. In *Procedings of the 12th Workshop of the Psychology of Programming Interest Group*.

BLACKWELL, A. F. AND GREEN, T. R. G. 2003. Notational systems: The cognitive dimensions of notations frameworks. In *HCI Models, Theories, and Frameworks*, J. M. Carroll Ed., Morgan Kaufmann, San Francisco, CA, 103–133

COHEN, J. 1960. A coefficient for agreement for nominal scales. *Edu. Psych. Measure. 20*, 37–46.

COHEN, M. A. 2005. Teaching agent programming using custom environments and Jess. *Newslet. Soc. Study Artif. Intell. Simul. Behav. 120*, 4.

COHEN, M. A., RITTER, F. E., AND HAYNES, S. R. 2010. Applying software engineering to agent development. *AI Mag., 31*, 2, 25–44.

ERICSSON, K. A. AND SIMON, H. A. 1993. *Protocol Analysis: Verbal Reports as Data*. MIT Press, Cambridge, MA.

HAYNES, S. R., COHEN, M. A., AND RITTER, F. E. 2008. Design patterns for explaining intelligent agents. *Int. J. Hum.-Comput. Stud.*

KADODA, G., STONE, R., AND DIAPER, D. 1999. Desirable features of educational theorem provers: A cognitive dimensions viewpoint. In *Collected Papers of the 11th Annual Workshop of the Psychology of Programming Interest Group*, T. R. G. Green, R. Abdullah, and P. Brna Eds., Leeds: Leeds University Press, Leeds, 18–23.

KO, A. J., AUNG, H. H., AND MYERS, B. A. 2005. Eliciting design requirements for maintenance-oriented IDEs: A detailed study of corrective and perfective maintenance tasks. In *Proceedings of the International Conference on Software Engineering*.

LEHMAN, J. F., LAIRD, J. E., AND ROSENBLOOM, P. S. 1996. A gentle introduction to Soar: An architecture for human cognition. In *An Invitation to Cognitive Science*, Vol. 4, D. Scarborough and S. Sternberg Eds., MIT Press.

NEWELL, A., AND SIMON, H. A. 1972. *Human Problem Solving*. Prentice Hall, Englewood Cliffs, NJ.

NEWELL, A. YOST, G. R., LAIRD, J. E., ROSENBLOOM, P., AND ALTMANN, E. 1991. Formulating the problem space computational model. In *Carnegie Mellon Computer Science: A 25-Year Commemorative*, R. F. Rashid Ed., ACM Press-Addison-Wesley, Reading, MA, 255–293.

PEW, R. W. AND MAVOR, A. S. Eds. 2007. *Human-System Integration in the System Development Process: A New Look*. National Academies Press, Washington, DC.

RUSSELL, S. AND NORVIG, P. (2003). *Artificial Intelligence: A Modern Approach* 2nd Ed. Prentice Hall, Upper Saddle River, NJ.

Received March 2010; revised March 2011, October 2011; accepted October 2011