Evaluating the Usability of Visual Programming Environments for Educational Robotics: An Activity Walkthrough Approach

Steven Robert Harris

Hypermedia Research Unit
University of Glamorgan, Wales, UK
srharris@glam.ac.uk

Mike Reddy

Centre for Astronomy and Science Education
University of Glamorgan, Wales, UK
mreddy@glam.ac.uk

Abstract

In order to address the difficulties that arise when applying the cognitive walkthrough method (Lewis & Wharton, 1997) to non-trivial interface design problems, Bertelsen (2003, 2004) has developed the activity walkthrough, a usability inspection method which adapts the cognitive walkthrough for use within the theoretical framework of activity theory (AT). In its current version, the activity walkthrough is based on the most general form of activity theory; this paper proposes that the efficacy of the method may be improved through the integration of concepts and principles drawn from the systemic-structural theory of activity (SSTA), an activity-theoretical approach specifically tailored to the study of human work and learning. As a first step in this direction, a prototype SSTA-based activity walkthrough was developed and used to carry out a usability inspection on 7 of the visual programming environments currently used in educational robotics. It was found that this amended form of the activity walkthrough proved effective in identifying design-dependent differences in the learnability and usability of the programming environments with regard to a benchmark task.

1 Introduction

With the growing availability of educational robots and robot kits, practical robotics is increasingly becoming part of the science education curriculum in schools, colleges and universities. Typical educational activities involve participants building and using robots to carry out basic tasks such as obstacle avoidance and line following, with observation and discussion of the process and results providing the basis for teaching and learning across a range of topics. In order to connect learning in science, technology, engineering, mathematics and information technology, pedagogical activities around physical robots also frequently include the creation of control programs on personal computers; finished code can be downloaded to the physical robot via cable or infra-red links for execution. Most educational robotics packages are accompanied by programming software designed for this purpose. Currently, the majority of such applications provide a visual programming environment, where programs are assembled through the direct manipulation of icons and controls representing code snippets, program parameters, and physical robot components such as sensors and motors. Some also include a simulation facility, allowing users to review and amend programs prior to download. The ease or difficulty with which these software tools can be learned and used has the potential to significantly affect the planning, conduct and outcomes of educational robotics activities.

This paper reports findings from the usability inspection of seven visual programming environments for educational robotics, which makes use, in an adapted form, of the activity

walkthrough method recently described in Bertelsen (2003, 2004). The evaluation approach is explicitly task-oriented, assessing the usability of applications with regard to the extent and manner in which they support the solution of a pedagogically relevant task-problem. While it is hoped that the outcome of the usability inspection may also be useful to interested educators, the principal aim of the exercise is to investigate (1) the general effectiveness of the activity walkthrough as a usability inspection method and (2) the potential for improving its effectiveness through the refinement of its underpinning concepts and principles.

2 Method & Related Work

2.1.1 The Cognitive Walkthrough

In the work reported here, an adapted version of Bertelsen's activity walkthrough method (Bertelsen, 2003, 2004; Bertelsen & Godsk, In Press) provided the basis for what is commonly known as a usability inspection (Nielsen, 1994) or structured expert review (Preece et al., 1994). The activity walkthrough is a variant of the cognitive walkthrough (Wharton, Rieman, Lewis, & Polson, 1994; Lewis & Wharton, 1997), a usability inspection method which has proven popular in design practice, where it can be applied to the expert review of a design specification without, or as a precursor to, actually building an interface or involving users. The basic technique of a cognitive walkthrough involves:

1. the identification of a typical task or tasks to be accomplished with the software;
2. the breaking down of selected task into a series of steps, taking into account issues raised by specific task cases and the user profile, especially with relation to user knowledge of the task domain;
3. an inspection process where evaluators use the software to work through a selected task with the application, at each step seeking answers to a series of user-related questions as a means for evaluating issues such as comprehension and ease of learning.

The evaluators' questions attempt to identify whether, at each step:

- the appropriate action is obvious to the user;
- if the user is able to connect the correct action with the desired outcome;
- if the action will trigger appropriate feedback from the interface.

2.1.2 Adapting the Activity Walkthrough to SSTA

In its original formulation (e.g. Polson, Lewis, Rieman, & Wharton, 1992), the walkthrough technique took a cognitive stance, and was based on a theory of exploratory learning (Carroll, 1982; Shrager & Klahr, 1986). Researchers in activity-theoretical HCI have tended to reject the cognitive walkthrough as inappropriately reductive; its use of hypothetical task analyses broken down according to the sequence of machine operations conflicts with AT views on the dynamic structure and conditions of actual activity. In particular, difficulties arise with regard to understanding the sense-making activity of the user in the absence of any real context for interaction (Bertelsen, 2003).

However, effective use of the walkthrough technique has proven not to be tightly linked to any particular theoretical framework; by structuring the inspection process around specific questions, other types of psychological theory may be embedded into the walkthrough. This is the course followed by Bertelsen; the activity walkthrough attempts to retain some of the practical advantages of the method while drawing on AT concepts and principles to provide more

systematic support for consideration of the real context of interaction. To underpin this adaptation Bertelsen draws on Leont'ev's General Theory of Activity (Leont'ev, 1978, 1981) and its interpretation in the Scandinavian School of information systems design (see Bertelsen & Bødker, 2003 for an overview). However, this general form of activity theory does not directly suggest any detailed analytical methods, making it difficult to apply to design problems (Bedny & Harris, 2005). For the purposes of the usability inspection reported here it was determined that the activity walkthrough method could be usefully further developed through the integration of concepts and principles drawn from the Systemic-Structural Theory of Activity (SSTA), an activity-theoretical approach specifically tailored to the analysis and design of human work and learning (Bedny & Meister, 1997; Bedny, Seglin, & Meister, 2000). SSTA provides an integrated activity-theoretical methodology that includes carefully developed units of analysis and methods to support their application to the holistic, multidimensional analysis of human performance; this suggested that the integration of SSTA methods with the activity walkthrough might result in further improvements in its effectiveness.

Table 1: Phases of prototype activity walkthrough in SSTA terms. *Adapted from Bertelsen (2003)*

Phase	Description	Steps
1	Preparation	• Identify appropriate task(s) to analyse
2	Contextualization	• Situate the application under evaluation in the context of use by identifying relevant users and use activities, user expectations, other artifacts mediating activity etc. (Qualitative analysis).
3	Verification of tasks	• Check that chosen task(s) are appropriate to the activities in which the application is going to be embedded
4	Task analysis	• Describe the main features of the task (is it deterministic, algorithmic etc.) and identify relevant task conditions. • Describe and classify the actions required during the performance of the chosen task(s) at an appropriate level of detail.
5	Walkthrough	At each step of task performance ask: 1. Does the interface support orientation toward the task-goal? 2. What is the user supposed to do? 3. Are the appropriate tools and objects visible to the user? 4. Can the user connect actions involving available tools and objects with the task-goal? 5. Does the interface support the development of new actions & operations as required (i.e. is it learnable in use)? 6. When appropriate, does the interface trigger established actions & operations? 7. When actions are performed, can the user perceive that progress is being made toward the task-goal (is there appropriate feedback)?
6	Task analysis verification	• Critically review the task analysis

Consideration of the walkthrough in SSTA terms suggested adjustments to each of its 6 main phases, presented in their amended form in Table 1. While space precludes detailed discussion, some general points should be noted. Firstly, while AT-HCI methods based on general activity theory have tended to dismiss the notion of *task* as an inherently system-oriented concept, in the systemic-structural approach tool-mediated work and learning activity is understood as an inherently task-based, problem-solving endeavour (Harris, 2005). Consequently, activity during task performance is a major object of study, and SSTA has developed useful taxonomies of tasks

and types of work process (see e.g. Bedny & Harris, 2004) which can be used to inform phases 1, 3, 4 and 6. Situating the activity walkthrough in relation to the four-stage methodology of systemic-structural activity analysis described in Bedny & Meister (1997 pp. 237-297) suggests that it is a "quick and dirty" approach that combines aspects of both the morphological and functional description of activity during task performance. Phases 1, 2 and 3 roughly correspond to the initial, qualitative stage of morphological analysis. In the amended version shown here, Phase 4 combines qualitative description with the identification of discrete actions; depending on the requirements of the analysis, this phase could be used as a basis for developing the detailed taxonomies of task action, algorithmic descriptions of the logical structure of a task performance, and time-structure analyses described in Bedny & Meister (1997), Bedny, Karwowski, & Seglin (2001), Bedny & Karwowski (2003), Sengupta & Jeng (2003), Bedny & Harris (2005), and Harris (2005).

2.1.3 The Benchmark Task

1. Identify & utilize method for creating new program
2. Identify and interpret visual representation of program & physical robot components
3. Identify relevant program & physical robot components
4. Identify & utilize method of adding components to program
5. Identify and interpret representation of flow of control in program
6. Identify & utilize method for linking program components
7. Identify & utilize method for setting relevant program parameters i.e. timers
8. Visually check program flow and rearrange as required
9. Where available, transfer code to simulation environment and execute
10. Evaluate results
11. Amend program as necessary
12. Download code to physical robot
13. Execute code on physical robot
14. Evaluate results
15. Amend program as necessary

Figure 1: Stages of the benchmark task

During the inspection each application was evaluated on its support for the completion of a basic programming task, chosen to represent typical introductory activities in educational robotics: preparing code that will switch on a physical robot's motor(s) for a given, finite, amount of time, causing it to move (by default, forward) and then stop. This benchmark task was chosen as representing a basic goal common to many introductory educational robotics activities: the demonstration of direct control of the physical robot via the programming environment, without the inclusion of environmental feedback from sensors. Our formulation of the test task embodied a number of assumptions: firstly, that an educator was present to support the learner, and that the educator has provided an objective formulation of the task-goal; secondly, that the PC-robot communication interface was enabled (found to be a non-trivial task during test setup); thirdly, that the programming environment was launched, active, and in the appropriate modality (again, a non-trivial task with some applications); fourthly, that the physical robot was powered up, ready, and fully enabled for motion; and finally, that the user had some information technology skills but would be considered a first-time user with regard to the specific programming application in use. This careful definition of the task as situated within a teaching and learning activity satisfies phases 1 and 2 of the activity walkthrough which were identical for all applications evaluated.

Phase 3 was addressed through the generation of a general purpose, idealized task description, shown in Figure 1, which was used as a basis for comparison with the actual actions required during task performance for each application. Where appropriate, this task description could be further decomposed into a detailed taxonomy of actions using the methods described in Bedny & Harris (2004).

In systemic-structural terms the benchmark task can be classified as probabilistic-algorithmic: steps must be carried out in a definite order, but each step requires multiple decisions. In common with many computer-mediated activities, task solution involves a significant proportion of exploratory activity requiring perceptual, orienting, and decision-making actions. The assumed presence of a more experienced mentor (the tutor) is considered as providing an additional channel for task-relevant feedback, acting to reduce user uncertainty and supporting sustained focus on the task-goal.

3 Evaluation Conditions & Inspected Applications

Table 2: Applications evaluated in the usability inspection

Name	Description	Full-screen?	Simulator?
Lego® RoboLab v.2.5.2	Visual programming environment distributed with Lego® Mindstorms for Schools™ education packs.	Yes on start-up, then reverts to windowed	Only available via WWW
Lego® Robot Invention System (RIS) v.2	Iconic programming environment which does not handle variables. Bundled with commercial Lego® Mindstorms™ kits, widely used in schools and colleges as (unlike RoboLab) does not require extra licence.	Yes	No
PicoBotz v.1	Visual programming environment to support the Open University Tribotz kit robot.	No, windowed	No
T184 RobotLab	Educational application developed by Open University. Procedural programming language. Not linked to specific physical robot, will run on Lego® Mindstorms™ robots.	No, windowed	Yes
Real Robots 3	Visual programming environment supplied with Cybot self-build robot kits.	Yes	Yes
Robot Works	Programming environment to support the Hasbro Wonderborg kit robot.	Yes	Limited simulation provides preview of the function of sense and command block functions, not of the final program.
Terrapin Logo	All-purpose combination command-line and visual programming environment incorporating Lego® RCX brick robot controller command functionality.	No, windowed	No

4 physical robots were used during the evaluation. These were simple twin-motor vehicles with variable combinations of touch, light and ultrasonic sensors. One was constructed using the Lego® Mindstorms™ kit; the others were commercially available robots with less flexibility in construction, but with some degree of assembly required. These were a Hasbro Wonderborg™, an iBotz PicoBot™ and a Real Robots Cybot™. Prior to carrying out the activity walkthrough, the programming applications were installed on the common platform of a Pentium 4 multimedia PC (clock speed 2.4 GHz, 512 MB RAM), running the Widows XP operating system, and equipped

with both serial and USB facilities for communication with physical robots. During the preliminary stages of the evaluation, observations were recorded on the installation and setup process for each application. It was found that the complexity of these tasks varied considerably between packages. The evaluators then applied the 6 phases of the activity walkthrough (listed in Table 1) to each application in turn.

3.1.1 Applications Inspected

Table 2 presents an overview of applications evaluated during the study. These provide a representative sample of physical robot programming environments currently in use at various levels of the UK education system, ranging from the widely used and commercially successful "off the peg" Lego® applications to bespoke educational packages such as Open University T184 RobotLab. The evaluation also included one representative of the earlier generation of widely used LOGO-based robot programming environments developed in conjunction with the Lego® RCX programmable brick (see Resnick, 1994 for an overview of the development history of Logo and Lego). There are a number of task-relevant functional aspects by which applications may be categorized. Firstly, the handling of windowing: for example, four of the applications default to full-screen displays in a manner similar to a computer game, thus blocking access to other applications. Another is whether or not simulation of program execution is supported, prior to download to the physical robot; in the sample, two applications – RobotLab (Lego) and Real Robots (Cybot) – include fully integrated simulation, while RoboLab (Lego) offers the opportunity to upload programs to Web-based simulator.

4 Findings

Main points emerging from the activity walkthrough are summarized in Table 3. The benchmark task was successfully completed – i.e. steps 1-14 of the idealized task description fulfilled – with six applications, with the degree of difficulty encountered during task solution varying across packages. Table 3 ranks the applications according to their performance in the inspection. T184 RobotLab emerges as the most learnable and usable. This application presents the user with a Guide menu offering pre-built, generic programs, specifically arranged around tasks; e.g. "Move a robot". Selecting this task from the Guide menu launches the programming environment with code and simulator panes open, and an appropriate stub program loaded and ready to run. The code window offers a hierarchical representation of program flow and commands, with decomposition of the task solution into primitives mapped to physical robot components. Invoking the run command from a pull-down menu executes the code in a simulator, which is active by default. The user is then able to activate the physical robot connection, download, and execute the program by invoking choices from the pull-down menu, completing steps 1-10 of the benchmark task. While both Robot Works and PicoBotz proved almost as usable, some problematic interface design elements tended to hamper task solution in both cases. Despite not being purpose-built for robotics programming, Terrapin Logo proved reasonably usable in supporting task solution, and might be recommended as a suitable option for educators who wish to have more all-purpose programming facilities available than is offered by the specifically robot-oriented applications.

One unexpected finding was that the official Lego® applications, currently widely adopted in the UK, proved among the least usable. The orientation of RIS toward various physical robot types, coupled with difficulties in accessing programming functionality proved disadvantageous to task completion; however, once in use the programming interface proved both learnable and effective. In the case of the specifically educationally targeted RoboLab, its arbitrary design orientation

toward different types and levels of user and overly complicated visual programming metaphor rendered it highly unsatisfactory in use. Despite persistent efforts, the evaluators were unable to complete the task with the final application, Real Robots. This software presented multiple difficulties connected with its primary design orientation toward the physical robot's onboard sensors; sensor-actuated program events proving difficult to map to the task level.

Table 3: Summary points from the usability inspection

Name	Task completed?	Design orientation	Visual metaphor	Comments
T184 RobotLab	Yes, easily.	Task/action	Command-line hierarchy plus simulator	Pre-programmed tasks available.
Robot Works	Yes, easily.	Program events	Command blocks and program threads	Idiosyncratic interface makes use of modality. Intrusive sound.
PicoBotz v.1	Yes. Problems at steps 9, 10.	Robot actions.	Icons, text labels, buttons, menus. Nested hierarchies.	Modal interface. Tutorial produces "fake" interface and overrides user control of application.
Terrapin Logo	Yes, with some difficulty. Problem at steps 3, 8, 9.	Program events.	Primarily command-line interface – Graphics window offers Cartesian coordinate space with "turtle" representing robot	Not primarily designed for physical robot programming. Uses logical switches (true/false) rather than on/off values.
Lego® Robot Invention System (RIS) v.2	Yes, with some difficulty. Problem at steps 1, 9, 10, 13.	Physical robot types.	Blocks with inbuilt connectors.	Lengthy introductory screens. Programming options initially locked out.
Lego® RoboLab v2.5.2	Yes, with great difficulty. Problems at steps 1, 4, 5, 6, 8, 9, 10.	Arbitrary user type.	Blocks, wires, plugs and sockets	No tool tips. Problems with creating and saving program files. Differing modalities available according to different user types and levels.
Real Robots 3	No. Problems at all steps.	Physical robot sensors.	Blocks and connectors.	Assumes always-running motors and control only via sensors. No auto-connect facility when assembling program blocks.

5 Conclusion

The usability inspection revealed marked, design-dependent differences in the learnability and usability of the visual programming environments when evaluated with regard to accomplishing a benchmark task designed to reflect the real context of interaction. The design-oriented insights gained through the use of the activity walkthrough lend support to Bertelsen's suggestion that adapting the cognitive walkthrough to the framework of general activity theory may address the need of usability inspection methods to provide more systematic consideration of the real context of interaction and improved understandings of the sense-making activity of users. This paper has also suggested further developments to the activity walkthrough as a first step toward integrating

more specifically design-oriented activity-theoretical concepts and methods drawn from the systemic-structural theory of activity into usability practice.

6 Acknowledgments

The authors would like to thank O. W. Bertelsen and G. Z. Bedny for helpful comments on earlier versions of this paper. We also thank iBotz (a division of Instruments Direct) and the software author, Peter Balch, for permission to review a development version of PicoBotz; and the UK Open University and author Jon Rosewell for permission to review the T184 RobotLab software.

7 References

Bedny, G. Z., & Harris, S. R. (In Press). The Systemic-Structural Theory of Activity: Applications to the Study of Human Work. *Mind, Culture and Activity, 12*(2).

Bedny, G. Z., & Karwowski, W. (2003). A Systemic-Structural Activity Approach to the Design of Human-Computer Interaction Tasks. International Journal of Human-Computer Interaction, 16(2), 235-260.

Bedny, G. Z., Karwowski, W., & Seglin, M. H. (2001). Activity Theory as a Basis for the Study and Redesign of Computer Based Task. In M. J. Smith & G. Salvendy & D. Harris & R. J. Koube (Eds.), Usability Evaluation and Interface Design: Cognitive Engineering, Intelligent Agents and Virtual Reality (Vol. 1). Mahwah, NJ: Lawrence Erlbaum Associates.

Bedny, G. Z., & Meister, D. (1997). The Russian Theory of Activity: Current Applications to Design and Learning. Mahwah, NJ: Lawrence Erlbaum Associates.

Bedny, G. Z., Seglin, M. H., & Meister, D. (2000). Activity theory: history, research and application. Theoretical Issues in Ergonomics Science, 1(2), 168-206.

Bertelsen, O. W. (2003). Activity Walkthrough: a cognitive walkthrough in activity theory terms. In M. Hertzum & S. Heilesen (Eds.), Writings in Computer Science: Proceedings of the Third Danish HCI Research Symposium (Vol. 98, pp. 17-20). Roskilde, Denmark: Roskilde University.

Bertelsen, O. W., & Bødker, S. (2003). Activity Theory. In J. M. Carroll (Ed.), HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science (pp. 291-234). London: Morgan Kaufmann.

Bertelsen, O. W., & Godsk, M. (In Press). WAW - an activity theory based tool for early website usability assessment. Accepted for publication in the International Journal of Human Resources Development and Management.

Bertelsen, O. W. (2004). *The Activity Walkthrough: an Expert Review Method Based on Activity Theory.* Paper presented at the Third Nordic Conference on Human-Computer Interaction (NordiCHI) 23-27 0ctober 2004, Tampere, Finland.

Carroll, J. M. (1982). The adventure of getting to know a computer. *IEEE Computer, 15*(11), 49-58.

Harris, S. R. (2005). *Design, Development, and Fluency: An Activity-Theoretical Approach to Human-Computer Interaction Research.* Unpublished PhD. Thesis, University of Glamorgan, Wales, UK.

Leont'ev, A. N. (1978). Activity, Consciousness and Personality (M. J. Hall, Trans.). Englewood Cliffs, N. J.: Prentice-Hall.

Leont'ev, A. N. (1981). Problems of the Development of the Mind. Moscow: Progress.

Lewis, C., & Wharton, C. (1997). Cognitive walkthroughs. In M. G. Helander & T. K. Landauer & P. V. Prabhu (Eds.), Handbook of human-computer interaction (2nd ed., pp. 717-732). Amsterdam: Elsevier/North-Holland.

Nielsen, J. (1994). Usability Engineering. London: Academic Press.

Polson, P., Lewis, C., Rieman, J., & Wharton, C. (1992). Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. International Journal of Man-Machine Studies (36), 741-743.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., & Carey, T. (1994). Human-Computer Interaction. Wokingham, UK: Addison-Wesley.

Resnick, M. (1994). Turtles, Termites and Traffic Jams. Cambridge, Ma.: The MIT Press.

Sengupta, T., & Jeng, O.-J. (2003). Activity Based Analysis for a Drawing Task. In * (Ed.), Proc. Ergonomics in the Digital Age, IEA Congress 2003 (Vol. 6, pp. 455-458). Seoul: The Ergonomics Society of Korea.

Shrager, J., & Klahr, D. (1986). Instructionless learning about a complex device: the paradigm and observations. *International Journal of Man-Machine Studies, 25*(2), 153-189.

Wharton, C., Rieman, J., Lewis, C., & Polson, P. (1994). The Cognitive Walkthrough Method: a practitioner's guide. In J. Nielsen & R. Mack (Eds.), Usability Inspection Methods (pp. 105-140). New York: John Wiley & Sons.