# Studying the learning of programming using grounded theory to support activity theory

Maryam Kheir Abadi* and Graham Alsop

*Information Systems and Mathematics, Kingston University, Kingston upon Thames, Surrey, UK*

*(Received 17 May 2011; accepted 20 June 2011)*

Teaching programming to first year undergraduates in large numbers is challenging. Currently, online supported learning is becoming more dominant, even on face-to-face courses, and this trend will increase in the future. This paper uses activity theory (AT) to analyse the use of tools to support learning. Data collection took place during 2008–2010 at Kingston University and involves over one hundred responses. This has been analysed into activity systems offering a detailed analysis of the use of a number of tools being used (in AT these include physical tools, such as technologies including books, and non-physical tools, such as conversation). When teaching programming to large numbers of students it is difficult to offer one-to-one attention and the reliance on such tools becomes more important. For example, in student responses a good integrated development environment (IDE) is shown to make learning easier and more enjoyable, whereas a bad IDE makes the learning experience poor.

Teaching materials, and access to these, were often mentioned positively. These included online communication, discussion boards and video lectures. Using AT offers sufficiently rich detail to identify key interventions and aids the redesign of the learning process. For example, the choice of an IDE for a specific language can have a larger impact than is initially apparent. This paper will report on the data collected to show where simple improvements to the use of tools may have a large impact on students' abilities to learn programming.

**Keywords:** learning programming; activity theory; grounded theory

## Scope

The communities that have been involved in this research are students, staff and the researchers. "Staff" is broadly defined as whoever teaches and helps students to learn. This includes technicians who support students with any difficulty they might have using software and associated development environments. However, this paper presents only the analysis of data from two groups of students. Staff experiences will be reported subsequently.

*Corresponding author. Email: Maryam@kingston.ac.uk

ISBN 978-91-977071-4-5 (print), 978-91-977071-5-2 (online)
DOI: 10.3402/rlt.v19s1/7801

## Methodology

As Alsop and Tompsett (2002) suggest, the methods that researchers use to collect data seem unimportant to students and to stakeholders such as lecturers. However, to achieve accurate and reliable data, the choice of methodology is critical.

Activity theory (AT) was selected because of the nature of the subject being examined. There are multiple communities involved in looking for the same outcome. The outcome could for instance be that of passing a specific assessment. AT allows for a holistic consideration of the multiple perspectives involved. Its ontology requires different research methods depending on the aspect being considered. In particular the choice of methodological approach to study the subject's activity is key. However, AT does not specify any particular research methodology to be used. We have chosen to combine it with grounded theory (GT).

The case studies undertaken here have particular characteristics that need considering. These include:

- The nature of the subject being examined (learning Programming);
- The number of students involved [in our case the numbers being relatively small and so qualitative research methods were chosen to 'provide a rich description of the students' behaviours' (Alsop and Tompsett 2002) during the research];
- The changes in sample during the research life-cycle; and
- The multiple communities involved in looking for the same outcome (in our case passing an assessment).

We now present short introductions to AT and GT to clarify why they were chosen and how they can work together.

### *Activity theory*

Activity theory is a psychological framework used to understand human activities. AT was introduced by Vygotsky (1896, 1934) and developed by Leont'ev (1981). Thereafter, many researchers have used AT in various subject areas. For example, Kuutti (1995) and Nardi (1996) used AT as a potential framework for human computer interactions and for transforming work in Information Systems. Scanlon and Issroff (2002, 2005) specifically utilised AT on the use of technology in Higher Education (HE). Engeström (1999, 2000, 2008) employed AT to examine individual and social transformation. They also developed the concept of an activity system (AS) to illustrate AT. Figure 1 illustrates the generic AS.

Nardi (1996) explains that an activity is the unit of analysis in AT and that the subject is the person or the group involved in the activity. The object stimulates the activity and provides goals and directions to the actions. Tools are the artefacts that can be used in the process of an activity. Other important factors in an AT framework are rules, communities and the division of labour. The whole result of an activity is the outcome or objective of the activity.

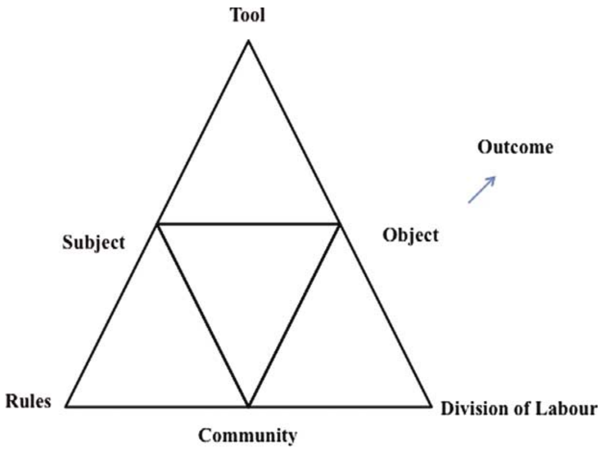

Figure 1. Generic activity system.

### *Combining GT and AT*

The choice of AT was justified by reviewing the characteristics of the case being studied, learning programming using specific tools on undergraduate taught modules involving multiple communities. We required a rich collection of information to ensure that AT ontology is described well. Choosing a method to collect the key data about how and what is used by students in learning and using it to show whether they have "learnt programming" is a challenge. The decision to focus on a qualitative approach was driven by several factors: sample size, accommodating researcher bias, and a changing sample during the research cycle.

### *Grounded theory*

Grounded theory (GT) was first introduced by Glaser and Strauss in social science almost 50 years ago. GT is an inductive qualitative research method that uses a systematic approach to constantly compare collected data and analysis. Here inductive means that there are no initial hypotheses. Accordingly, the researcher has to be as open minded as possible and design the research questions carefully.

Grounded theory interacts closely with data. Any possible hypothesis or theory is driven from the data, as Glaser and Strauss suggest:

> …clearly, a Grounded Theory that is faithful to the everyday realities of the substantive area is one that has been carefully induced from the data. (1967, p.239)

While Glaser and Strauss (1967) believe that new concepts and reality can be discovered from the collected data, Corbin (2008) argues that there is no reality out there waiting to be discovered, rather there are concepts and ideas that can be invented. She continues that humans do not discover reality. For example, Schwandt (1998) states that:

> …constructivist means that human beings do not find or discover knowledge so much as construct or make it. We invent concepts, models and schemes to make sense of

experience and, further, we continually test and modify these constructions in light of new experiences.

Charmaz (2006) also believes that theory is constructed from the data:

Grounded theory involves taking comparisons from data and reaching up to construct abstractions and then down to tie these abstractions to data. It means learning about the specific and the general – and seeing what is new in them – then exploring their links to larger issues or creating larger unrecognized issues in entirety... Grounded Theory methods can provide a route to see beyond the obvious and a path to reach imaginative interpretations. (Charmaz, 2006)

GTM is categorized as an inductive method. Induction can be defined as a type of reasoning that begins with study of a range of individuals' cases and extrapolates from them to form a conceptual category. (Charmaz, 2006)

Other methods and approaches were considered. These included action research and phenomenography. The choice of GT above other methods and approaches appears to have been sound in the light of the initial outcomes. A further paper on the methodological issues is in preparation.

Activity theory helped to break down complicated situations and made them easier to analyse. GT allowed us to have a flexible and open approach to data collection. It also allowed us to decrease the number of presumptions and hypotheses which would have limited the possibilities of findings. However, the decision to use AT implied the need to conform to ontology. This led to data collection using some of the terms required by AT.

## Data collection

Two open-ended questions (adapted from Alsop and Tompsett 2002) were used in this research. Students were asked to write about their best and worst educational experiences of learning programming and to specify the tools they used. They were also asked to summarise their stories in their own words. This was to ensure that the data received were framed in the language of the 'students' rather than the 'researchers'.

## Data analysis

In using GT, the collected data were examined closely. In considering each response, questions were asked such as what has happened and why has it happened? An AS was built for each response as well as associated notes that included the researcher's analysis of the case. The early stages of GT analysis were then used. This included open coding (whilst keeping in mind AT's ontology which includes subject, object, tools, rules, communities and division of labour). In the first instance, the focus taken was on which tools the students have used and which communities have been involved in that event. Thereafter, axial coding was undertaken in two sets; one for the worst experiences and one for the best experiences. For the former this represents, put in the language of AT, 'contradictions' that in an AS need to be overcome for the activity to be successful (the resulting redesign, again specifically in the language of AT, is known as a 'shift' in an AS.)

***Building activity systems***

A first year student (Subject), who attends the "Programming Essential Module" (Object) in the first semester, is taken as a starting point. She/he goes to the lecture and listens to the lecturer who aims to explain the basic concepts of programming in Java. She/he picks up a handout and annotates it (this represents a Tool in AT terms). After the lecture, the student has the opportunity to go to the workshop to put the theory into practice. Here, she/he practices Java in a real environment using several other Tools (TextPad, the WWW, accessing the internet, notes and books.) She/he can also receive help from lab assistants (another Community). She/he is interacting with a machine, reading her/his notes and books and interacting with students, lecturer and assistants in order to achieve her/his goal (Outcome) of "learning programming".

As Engeström (1999) suggests in order to achieve a specific outcome there needs to be a subject, object, rules, tools, division of labour/effort and finally communities. Figure 2 has been annotated from Engeström to illustrate the example.

In the process of learning programming there are very likely to be problems, clashes, breaks and difficulties. Engeström (1999), Nardi (1996) and Roussou, Oliver, and Slater (2007) call these disturbances "contradictions". Analysis of data in this research aims to identify and clarify these contradictions and help identify how they can be solved to make the process of learning programming better, more smooth and enjoyable for students. For example, if the student above begins by writing a simple program called "hello world" in Java, compiles code and then faces syntax errors, she/he could either solve the errors with no help or call on help. This reflects the ability to self-correct or need an intervention from someone else in order to solve the difficulty.

To move away from a general example to something more specific, we give a response citing a student's worst experience:

> "While creating a game in $C^{++}$, using provided engine, I couldn't get it to do what I needed. The program compiled fine, so it was down to my logic. The lecturer suggested using break points to find my mistake – but I could not understand what they were telling me or if I was using them correctly. This was the most frustrating experience of a

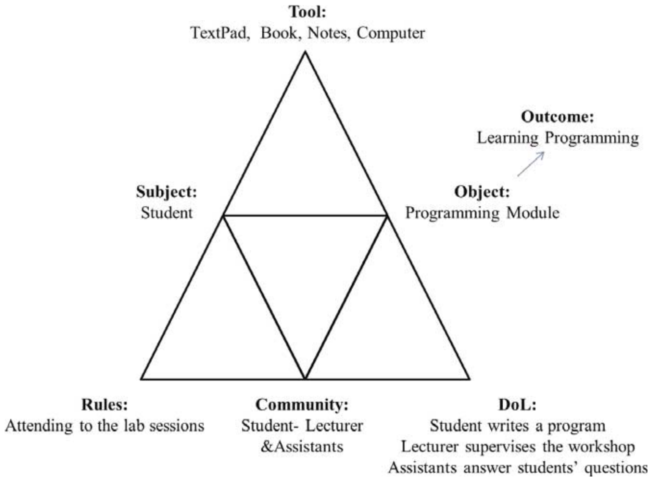
**Tool:**
TextPad, Book, Notes, Computer
**Outcome:**
Learning Programming
**Subject:**
Student
**Object:**
Programming Module
**Rules:**
Attending to the lab sessions
**Community:**
Student- Lecturer &Assistants
**DoL:**
Student writes a program
Lecturer supervises the workshop
Assistants answer students' questions

Figure 2. An illustration of the methodology.

few while using the visual studio suite – I didn't like using such a complex program without understanding how to use it properly, or having a thorough knowledge of the language beforehand." – "Not knowing how to use the program".

We build up an AS. The coding is shown in Table 1.

Figure 3 shows the AS of the above event.

In this example shifts (improvements) are needed in both the tool and communication process. The tool seems to be too complex for the student to use. Why is this the case? It could be because the environment is new or perhaps because Visual Studio is not a good development tool for programming?

We do not know the answer to the latter question. However, we can investigate the former question. The problem of a new environment can be solved by a "short lived goal directed" action (Engeström 1999). A shift in the object of the activity of learning programming in C$^{++}$ to "learn the IDE", in this case Visual Studio, together with a short-term shift in the division of labour by having workshops to learn the integrated development environment (IDE) instead of writing code/ programming could help. In Figure 4, these shifts have been illustrated.

In contrast to a failed AS, an analysis of a good experience follows. This response was chosen from one of the set of best educational experiences from the same group as the previous student. This response, however, was not selected randomly. It was chosen because this student (Subject) shares the same Object with the randomly selected one above. However, the description is of a best rather than a worst educational experience. This shows that the same Object can be the reason for both good and bad experiences.

"The best educational experience when using a programming tool would be the time when I had to program a game using C++. Normally i would find this challenging so I decided to do more independent work using the program such as reading books and practicing on simple programs. By the time I'd finished I had created what I thought exceeded my expectations and for which I received a good mark. This was very satisfying and now I spend longer on independent work." – "Spending time on a program is beneficial and is helpful for work".

This leads to the following coding (Table 2) and AS (Figure 5) developed in the same way as the previous example.

Figure 5 shows the AS of the above event.

Table 1. An activity system coding.

| | |
|---|---|
| Subject | Student |
| Object | Game design |
| Tool | Visual Studio (VS – an IDE from Microsoft to develop programs in C$^{++}$) |
| Community | Student and the lecturer |
| Division of labour (DoL) | Student writes the codes and lecturer helps to find errors |
| Outcome | Confusion, frustration, student did not understand and not happy to use a complex program |

Note: Researcher's interpretation: It seems that the student is not comfortable with the tool (VS), and finds it a to be a complex program. Despite seeking help she/he still does not understand the problem. Is it because the problem has not been fully explained or is it too hard for the student to digest? This student would prefer to understand the concepts before using it in the provided IDE. She/he summarises the story: "Not knowing how to use the program"

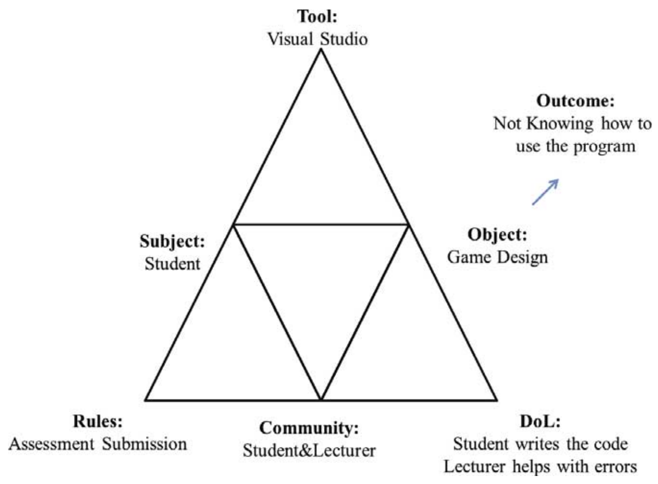

Figure 3. A specific activity system.

Subsequently, ASs were built for each response and these were then categorised. In GT, Corbin (2008) call this process "identify concepts from data". In other words, labelling data with specific words and terms, adding commentary about data analysis, stating comparisons and investigating ideas that appear in data. The result of this process for the failed ASs is shown in Figure 6.

Since the students were asked to summarise their stories, those summaries guided the researcher to label each response and then classify these into groups. Knowledge, structure, tools and programming languages are the four main categories that were driven from the data.

Knowledge contradictions are mostly related to syntax and materials. Some of the students found the syntax, taught for the specific programming language, hard to learn or too much for the duration of the semester. Some other students pointed out that the materials available for the modules did not cover the harder assignments.

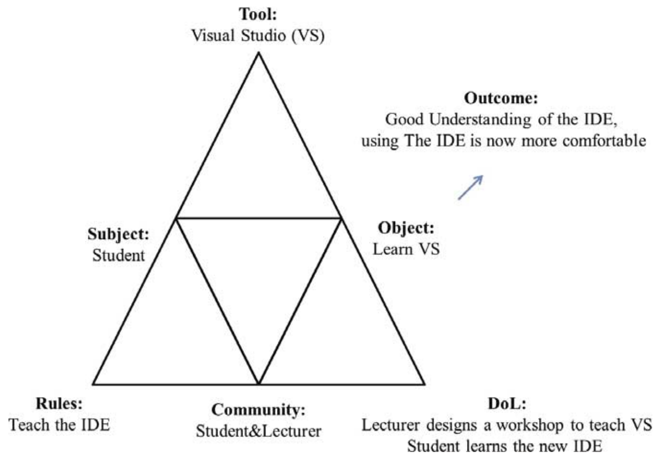

Figure 4. Modified activity system.

Table 2. A further activity system coding.

| | |
|---|---|
| Subject | Student |
| Object | Game design |
| Tool | Books |
| Community | Student |
| Division of labour (DoL) | More independent work and reading books |
| Outcome | Good grade, satisfaction and encouragement |

Note: Researcher's interpretation: From previous experience the student knew that she/he might have problems with designing the program, that is why she/he decided to do more independent work, study more, read related books to improve his/her ability to design the game. The student summary is clear: "Spending time on a program is beneficial and is helpful for work".

Tools are another contradiction in the process of learning programming for these students. IDEs specifically seem to be limited for the work they do. Students also highlighted that some of the IDEs in use are not helpful in terms of solving errors.

The same analysis is applied to the positive experiences in Figure 7, which illustrates the best educational experiences.

## Conclusions

There are two main conclusions. Firstly, using AT with GT has led to the identification of contradictions that require shifts to lead to successful ASs that ensure that students are better able to learn programming. These are that:

1. The choice of IDE is important. Simple IDEs do not provide the required feedback to ensure adequate problem solving.
2. It seems that the tools are not as important as the behaviour/motivation of the student toward learning. Utilising all available tools (such as books, IDEs, online videos, search engines, Blackboard, Study Space, etc.) can

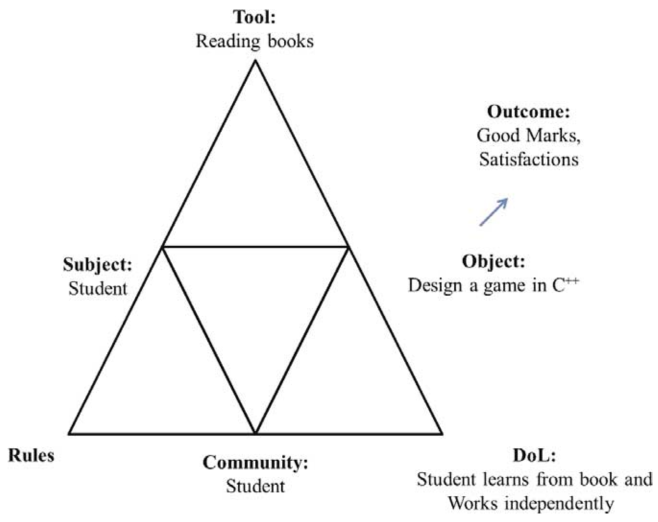

Figure 5. Associated activity system.

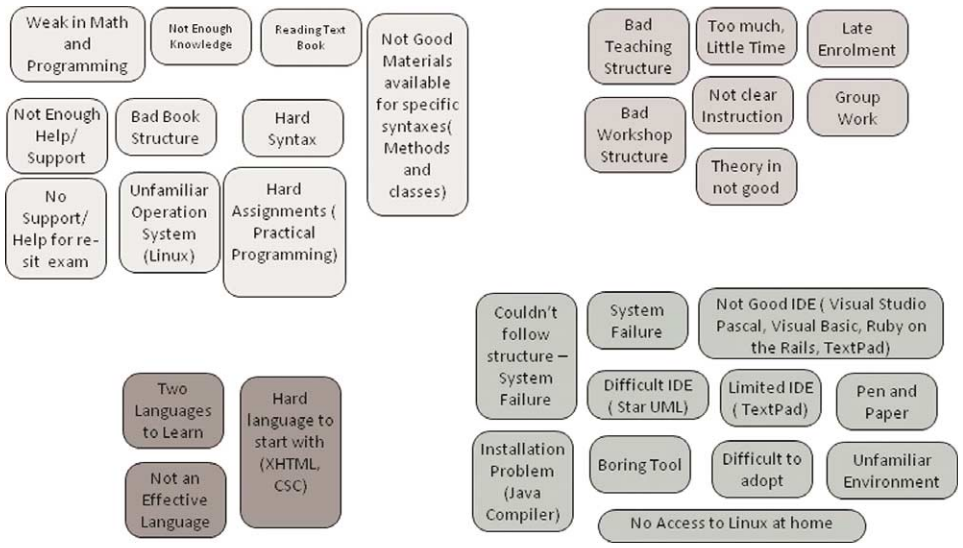
Weak in Math and Programming
Not Enough Knowledge
Reading Text Book
Not Good Materials available for specific syntaxes( Methods and classes)
Not Enough Help/ Support
Bad Book Structure
Hard Syntax
No Support/ Help for re-sit exam
Unfamiliar Operation System (Linux)
Hard Assignments ( Practical Programming)
Bad Teaching Structure
Too much, Little Time
Late Enrolment
Not clear Instruction
Group Work
Bad Workshop Structure
Theory in not good
Couldn't follow structure – System Failure
System Failure
Not Good IDE ( Visual Studio Pascal, Visual Basic, Ruby on the Rails, TextPad)
Difficult IDE ( Star UML)
Limited IDE ( TextPad)
Pen and Paper
Two Languages to Learn
Hard language to start with (XHTML, CSC)
Not an Effective Language
Installation Problem (Java Compiler)
Boring Tool
Difficult to adopt
Unfamiliar Environment
No Access to Linux at home

Figure 6. Analysis of failed activity systems.

increase the motivation to learn. These tools make it easier to learn independently of location and time.

3. The communities involved in the activity of learning programming are another high priority in the responses. The better the level of communication, the more rewarding the activity. The interactions between communities

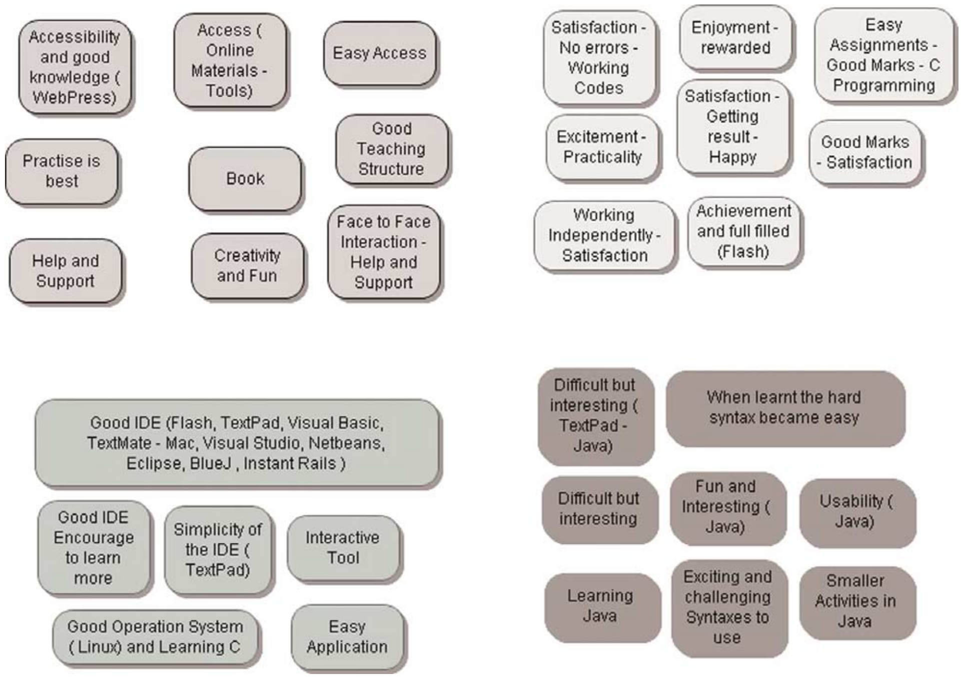
Accessibility and good knowledge ( WebPress)
Access ( Online Materials - Tools)
Easy Access
Practise is best
Book
Good Teaching Structure
Help and Support
Creativity and Fun
Face to Face Interaction - Help and Support
Satisfaction - No errors - Working Codes
Enjoyment - rewarded
Easy Assignments - Good Marks - C Programming
Satisfaction - Getting result - Happy
Excitement - Practicality
Good Marks - Satisfaction
Working Independently - Satisfaction
Achievement and full filled (Flash)
Good IDE (Flash, TextPad, Visual Basic, TextMate - Mac, Visual Studio, Netbeans, Eclipse, BlueJ , Instant Rails )
Good IDE Encourage to learn more
Simplicity of the IDE ( TextPad)
Interactive Tool
Good Operation System ( Linux) and Learning C
Easy Application
Difficult but interesting ( TextPad - Java)
When learnt the hard syntax became easy
Difficult but interesting
Fun and Interesting ( Java)
Usability ( Java)
Learning Java
Exciting and challenging Syntaxes to use
Smaller Activities in Java

Figure 7. Analysis of positive activity systems.

include: student–student, student–lecturer, student–helpers and student–technicians. The involvement of technicians is critical because most of the technical problems occur the first few times that students begin to use a new environment.

4. A large mixed group does not help to support learning. Designing a suitable lecture/workshop for even the majority of a large diverse set of students is very difficult. Examining all students' programming knowledge before the first programming module is one possible solution to enable streaming or, at least, offering more targeted support/advice.

Secondly, in choosing AT as the framework, there was an implied need to find an appropriate approach to collect and aid the analysis of data, which needed also to be compatible with AT. GT was chosen to complement AT. Using GT offered sufficiently rich detail to identify key interventions and ways to redesign the learning process. AT helps to clarify any contradictions in an AS and provides a means to design changes/shifts to solve these contradictions. Using AT with a series of developing ASs can show the history of contradictions, changes and shifts during the process of learning. This aids the development, knowledge, structure and design of a better learning environment for the future. As Engeström (2000) argues, there is never a finished product in the learning process since there is always a moving target.

## References

Alsop, G., and C. Tompsett. 2002. Grounded theory as an approach to studying students' uses of learning management systems. *ALT-J* 10, no. 2: 63–76.

Corbin, J., and A.L. Strauss. 2008. *Basic of qualitative research, techniques and procedure for developing grounded theory*, 3rd ed. London: SAGE.

Charmaz, K. 2006. *Constructing grounded theory, a practical guide through qualitative analysis.* London, Thousand Oaks: SAGE.

Engeström, Y. 1999. Activity theory and expansive design. http://projectsfinal.interactionivrea.org/2004-2005/SYMPOSIUM%202005/communication%20material/ACTIVITY%20THEORY%20AND%20EXPANSIVE%20DESIGN_Engestrom.pdf (accessed June 2009).

Engeström, Y. 2000. Activity theory as a framework for analyzing and redesigning work. *Ergonomics* 43, no. 7: 960–74.

Engeström, Y. 2008. Enriching activity theory without shortcuts. *Interacting with Computers* 20, 2: 256–9. Perseus Digital Library. www.elsevier.com/locate/intcom

Glaser, B.G., and A.L. Strauss. 1967. *The discovery of grounded theory, strategies for qualitative research.* Aldine Pub. Co: Weildenfield and Nicolson

Kuutti, K. 1995. *Activity theory as a potential framework for human computer interaction research, context and conciseness*, 2 vols. 17–44. Cambridge, MA: MIT Press.

Leont'ev, A.N. 1981. *Activity and consciousness in the development of mind, selected works of Aleksei Nikolaevich Leontyev.* Trans. M. Kopylova and ed. A. Blunden. http://marxists.org/archive/leontev/works/development-mind.pdf (accessed 5 July, 2010).

Nardi, B.A. 1996. *Studying context: A comparison of activity theory, situated action modules, and distributed cognition*, 4 vols. 35–52. Cambridge, MA: Massachusetts Institute of Technology.

Roussou, M., M. Oliver, and M. Slater. 2008. Exploring activity theory as a tool for evaluating interactivity and learning in virtual environments for children. *Congnition Technology & Work* 10: 141–53.

Scanlon, E., and Issroff, K. 2002. Educational technology: The influence of theory. *Journal of Interactive Media in Education* 6, p. 1–13. [www-jime.open.ac.uk/2002/6]

Scanlon, E., and K. Issroff. 2005. Activity theory and higher education: Evaluating learning technologies. *Journal of Computers and Learning* 20, no. 6: 430–9.