

Formal Approach to Modelling Learning Patterns

Cat Kutay
University of New South Wales
Sydney, 2052

Dec 19, 2005

Abstract

This work provides a formal development of a language for specifying agents to perform the analysis of group activities and learning support within a groupware system, based on modelling patterns of learning. The parameters in this language are updated by the agents in the individual or group models and used to select optimal feedback to provide to students learning activities. While the agents discussed have been developed within a specific software system, the language provides a generic formal approach to agent development in the domain of learning using groupware.

The domain considered is an open learning environment where students are solving an open-ended problem. In contrast to an Intelligent Tutoring System, there are no correct answers, only the development of concepts and processes that improve the learning outcome. However, given the computer is still the mediator in the communication, we aim to provide scaffolding for the learning that takes place. Hence we need to develop a model of the learning efficiency or how to identify breakdowns. The results of this modelling are open to the user through the scaffolding provided rather than direct access to the blackboard style communication database. Firstly if the feedback is aimed at reducing behaviour that reflects poor learning, the agent is considered effective if the behaviour reduces. Secondly the feedback can be directly assessed by the user and marked as appropriate or otherwise.

The formal language is used to describe rules of learning activities which can be analysed by computer. Through the formal technique of refinement, rules can be combined and enhanced. Through storage over time in an individual and a group model, the analysis can be extended to more complex rules. The formal language is used to describe the agents which analyse the learning events and their modelling parameters which provide suitable scaffolding within the groupware.

Introduction

In this paper we describe a formal language used to develop agents for the analysis of learning activities. The agents are designed to provide scaffolding for the students' activities. The paper starts with an overview of the activities that are to be analysed to develop the semantics of the language.¹

¹Terms in this language are written in italics to distinguish them.

The idea of formal modelling as proposed here is designed to provide a structure for modelling student activity in an open learning environment rather than a tutoring system. The formalism is designed for analysis of student processes to improve estimation of the level of understanding reached by a group. Similarly formal rules are developed for analysing patterns of interaction and participation.

Research done by Constantino-González and Suthers [13] on participation and by Sollers and Lesgold [31] on interaction patterns have produced methods to extract patterns from known good and poor learning interactions. Given such patterns have been extracted, a learning support system is designed here to match such patterns against groups in action.

The domain of learning is a course at the University of New South Wales where students are required to develop Software Engineering Requirements and Specification documents, using basic drawing and text editing tools to describe the system. ²

The approach of this paper is to look at learning events rather than learning states, since the latter cannot be deduced in open ended learning environments. Work by Soller and Lesgold [31] uses Hidden Markov Models to locate optimal learning interactions, or sequences of activities. The Hidden model is used due to the inability to determine the state after each event or activity within the software interface. In the present work the formalism is based on the events and their sequences, using the event based formalism of Abrial [2].

Once the language to analyse events is developed, the outcome of the analysis provides feedback to students. There are two options to verify the analysis. Firstly the feedback may result in improved learning or interaction activity by the students. Secondly the feedback can be accessed by the student for relevance and this response used to evaluate future feedback.

Prior Work on Developing a Formalism

Akhras and Self [8] and [5] have previously used Situation Theory as the basis of their analysis of learning activities. Situation Theory differs from the present Activity Theory approach (also used in the work by Fjeld et al [17]) in that the emphasis of the former is on the emergence of human

²The groupware system used is described in Kutay [23]

activity out of the particulars of a given situation. This forms a state based analysis of the learning activity.

In a Situational analysis the goal of the activity is a factor specified in the context, rather than a coherent plan. The formalism developed by Akhras and Self [8] analyses the learning environment and process as two factors and adds the dimension of time, or extended action. This change to Situation Theory was suggested by the development of Activity Theory [24]. Akhras and Self formalised the processes in an Interactive Learning Environment by considering:

1. The context of learning including content, dynamics and development
2. Analysis of learning interactions through patterns, relation to context and learner's knowledge and patterns between learning situations
3. Analysis of the time-extended processes which are either cumulative, constructive, self-regulatory or reflective

This analysis was further developed by Mühlenbrock [25] who developed *Activity Recognition* based on this situation calculus and then including plans. Mühlenbrock uses the plan and task recognition developed by Kautz [22] and Hoppe [21].

While the present work is similar to the activity analysis of this previous work, it avoids the need for the initial situation calculus, by citing preconditions, background, and constraints in terms of activity and events only. Also postconditions are in terms of activity to be taken rather than situations. In this work the problem of multiple non-occurring actions, or the frame problem, is avoided by considering the consequences of the actions, rather than attempting to define all possible actions.

Mühlenbrock describes his formalism in the following sections:

1. Actions and operators within the environment
2. Situations representing workspace properties
3. Operators that relate actions to situations and other actions with preconditions, postconditions, backgrounds, decompositions and contratins,
4. Complex actions based on a sequence of simple actions

5. Composed actions which are not yet completed
6. External actions are included, either previous asynchronous actions, or messages from another user's interface

This paper uses a similar structure but uses an Activity Theory analysis of the learning to developing suitable scaffolding for the learning context and so enhance the learning process.

Outline of Approach

The formalism is designed to explicitly develop scaffolding that is relevant to the learning context at the present, and to define, encourage and enhance three aspects of learning (see Dillenbourg and Self [16]):

1. Verbalising strategic decision through conflictive actions which require explanation.
2. Acquire reflective skills through conflict between individual knowledge and individuals experience which requires resolution.
3. Acquire better models of learning through conflict between different individuals' knowledge which requires resolution.

The focus is not so much on modelling the learning environment or learning states, as in modelling the state of the learner indirectly through the events they have gone through or actions they have taken as discussed in the section on Event-based Formalism. The formalism use the event analysis of the Activity Theory model to characterise the learning processes as discussed in the section on Activity Analysis.

Context

The context for analysis is an groupware learning environment where students are solving an open-ended realistic problem and producing a document or report describing their solution. Also graphical components for producing domain specific illustrations of their solution are considered in the analysis. In contrast to an Intelligent Tutoring System, there are no correct answers,

only the development of concepts and processes that improve the learning outcome.

The physical interface for which the formalism was enacted as agent support is described by Kutay [23]. The groupware allows students to open and view with their group tools that edit documents, make notes and draw graphics, as well as providing a standard Chat tool. Chat contributions can be augmented with token and topic descriptors to assist analysis. Another tool that has been included in this model is a planner in which the student may plan their work, or a course planner can be provided.

The significance of the domain is that the process of learning is very open. There is no correct solution to act as a guide in analysing contributions. However there are rules for diagram and document formats. There are no restrictions on the chat contributions. However there are patterns of interaction which are optimal for learning. Finally there is no knowledge of individual students differing views except where these are expressed in the group. The analysis of the learning process as developed below, relies on this generic data.

Event-based Formalism

The sequence of events as they are enacted can be described in the form of Hidden Markov Chains describing the linking events rather than outcome states. The agent implementation rules analyse the chains for effective or ineffective learning. The rules can be developed from simple two event chains, and then added to improve the complexity of the analysis. The outcome states are probabilistic in nature.

The event system formalism developed by Abrial [2] is used to model discrete transition systems. The formal model consists of constants, variables and the invariant of the model. The events or transitions consist of a guard or necessary condition of the transition and an action or effect of the transition. The initialisation is the initial state or condition of the variables or actions. The complexity of the formal model is handled by refinement, decomposition and generic instantiation (See Abrial [3]).

In this paper we consider the following aspects of the learning:

1. Context. The invariant and the variables and constants of the domain and their initialisation are used to describe the context.

2. Learning and Group Interactions. The events include guards or conditions and actions which are used to describe the interactions.
3. Cognitive Process. These are described by functions of variables and constants which are linked between agents through a learner model.
4. Affordances. These are handled by refinement (related to agent extension), decomposition (related to agent implementation) and generic instantiation (related to generic agent types).

Although like Akhras and Self [4] we are describing the learning domain or situation, we are describing this in terms of events or actions, hence there is a direct translation from students' learning activity to the computer analysis of the learning used to derive scaffolding.

Domain Specific Agent Language Concepts

While the formalism is designed for translating generic learning goals and strategies into software agents, the following aspects would be entirely domain specific and hence would have to be developed for each domain of learning through the provision of such tools as templates and a lexicon of the significant jargon of the domain:

Timing An outline plan of the course should be provided to enable analysis of the timing component of learning. For instance, different skill and concepts may be introduced later in session, so should not be assumed at the start of the work.

Form of Scaffolding The form which the scaffolding or advice takes should suit the level of learning of the student, starting at the basic level then gradually builds up to more advanced concepts (see Brown, Collins and Duguid [9]). For instance one approach is to use the course temporal progression to set learning level. Then groups might initially be advised to supply the basic required document format. Later in the semester it will be suggested that particular sections may be too long. Finally towards the end of the semester they will be prompted with general advice for finalising the document in relation to the assessment criteria.

Semantics The semantics for the graphical format specific for that course should be supplied. In the case of specific diagrammatic forms used

in the particular domain, there are rule which can be verified from analysing properties of the diagram, and some rules that can be provided as a guide only, as they would require sophisticated language analysis to verify.

Resources To enable student self-regulation of their learning various material resources should be directly linked to the software. Also a version tree structure to support a group document trace of changes provides a tool for reflection and rework.

Concepts The important concepts of the domain could be provided as a keyword list or lexicon for students to enable the tool to follow the development of concept use by the students, and to provide a cumulative reference to each concept.

We now proceed with the development of the formalism under the sections shown in Figure 1). Examples of each part of the formalism are shown and related to the learning scaffolding to be provided through agent analysis using this formalism.

Activity Sets and Functions

The basis of the formal modelling is operations on sets. In this domain we are dealing with activities of students in groupware, and wish to analyse these activities and their function attributes. The activities and their attributes make up the elements of the sets to analyse. The present work covers the sets: Users, Text records, Graphical records and File handling. Implementations in other learning domains may add to this list.

Activity Analysis

The context is represented by the variables and constants which may be generic or specific to the domain. The invariant and initialisation of the system define the context, so the language must be developed to describe the range of contexts with which we are dealing.

The context of the learning is in two parts. There is the context in which the group is working, the combined experience and information they can draw on, which is classified as Learning Activity. Also there is the context

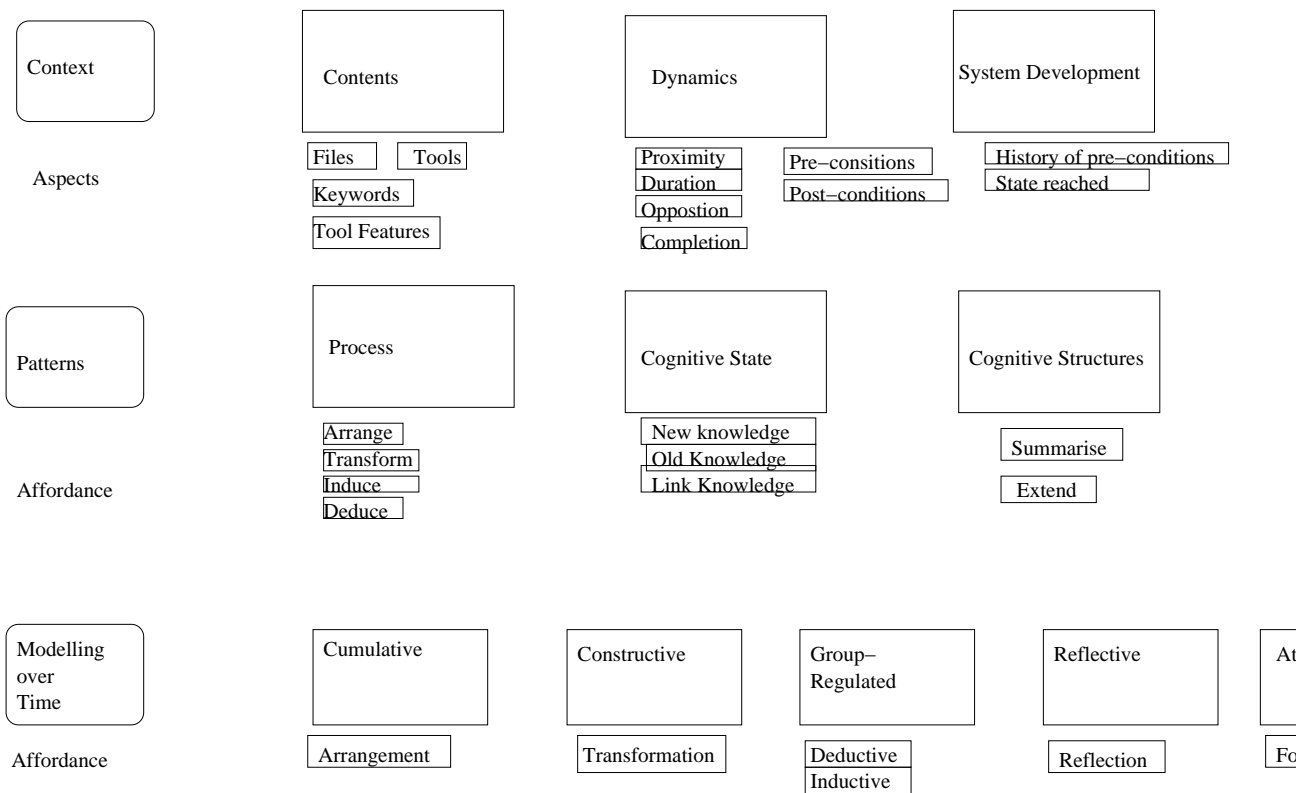


Figure 1: Student Learning Model - aspects for formalism

for the individual and how they fit into the group, or the group learning domain and is classified as Collaboration Activity.

The context is then a model based on the existing information on the screen or the content of files (the information the group has chosen to open in a tool or add to a tool), the information in focus (such as the topic of discussion), the sequence of changes or the dynamics of the action (such as the particular tool selected for reading or editing) and historical links that the group has made between various pieces of knowledge or the system development. This information is available to some extent for the analytical agents to access. Also, the context can be updated by adding information and tools to the screen, for instance changing the particular student's software interface to suit a change in role. Finally, the resources or Community, such as previous years' students' projects, form part of the database of information which the agents can draw from to describe the context.

The context language covers the aspects of the Activity Theory Model described in Figure 2 and links them to the formalism as follows:

1. Interface Content (Subject, Object and Community) – Considered as the variables, constants and their initialisation.
2. Activity of interest (Tools, Rules and Division of Labour) – Considered as the functions including invariant and guards.
3. Objective of the Learning (Outcome) – Considered as events acting on the variables to produce the outcome object.

Interface Content

The process of learning can be represented by actions on the entities of the learning environment and process entities through the system interface. Environmental entities are the concepts, actions, physical objects and information related to the subject matter in the domain of learning. Process entities represent information about the occurrence of these entities in the process of interactions. These are the content of the learning activity and are carried out on entities of the interface, such as the interface to the document file or graphics file.

The entities are represented by constants and variables in the formalism. In most cases the entities that are analysed by agents are attributes of other

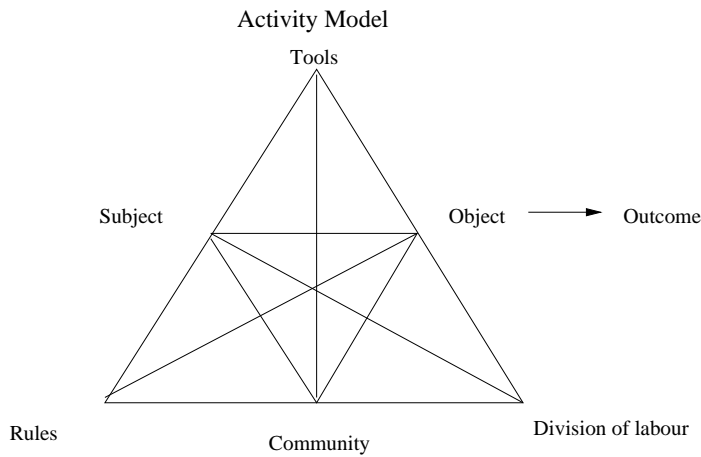


Figure 2: Activity Theory Model - aspects for formalism

larger entities as described below in Section on Content Formalism. Their value is established at initialisation and varies within the invariant parameters.

According to Akhras and Self modelling environment entities corresponds to specifying the types of situations the learner may encounter during the course of their interactions with the environment. From the Activity Theory approach the environmental entities are the objects of the environment that may be acted upon. These are the files, the text or diagrams and their specific features (such as views of documents, token in Chat, etc) and the specific keywords from the Lexicon of the domain provided as part of the agent system for that domain.

Activity of Interest

To represent the dynamics of actions we define events, context of events and use the invariant and the event guard to set the conditions and limit the effects of events. We need to develop a language of relations between entities to set these guards and invariants. Also we need constants that can be used between states and agents to describe the common attributes, which are stored in the Learner Models.

The process entities are still not located in time, although they may be analysed for sequence of occurrence. The way a learning process develops is by analysis of entities and their interactions through proximity, duration,

concept occurrence, opposition, completion or absence in an interaction. To carry out this analysis the agents need to have encoded rules for these interactions.

It should be noted that the actions we are analysing at present are those initiated by the learner. Automatic analysis by the system is a further extension of the work not yet attempted. The present formalism deals with the invariant relations that are maintained or through the activity as well as desired functions leading to improved learning.

Objective of Learning

The learning is carried out by a series of events. These series of operations and actions develop into interactions that can be analysed for plans, in a sophisticated system, or in this case, analysed for patterns. As examples of sophisticated systems, the analysis of spatial patterns in human-computer tactic games using post-analysis of games logs with Hidden Markov Models has produced some pattern recognition opportunities (see Sukthankar and Sycara [32]). Also Bui [10] has proposed an implementation of the Abstract Hidden Markov model to recognise plans or policies in online activities.

Agents can recognise predetermined patterns as predicting an achieved state, if pre-conditions have been satisfied and then the expected action occurs. The agent can then assume the post-condition which may involve supplying advice or making other changes in the interface. This agent process is described in the next sections as a means to analyse the cognitive structure of the learning.

An important aspect is that the sequence of actions involved in a learning pattern, may not be sequential, but separated in time, maybe even spread over more than one group session. The agents need to keep track of the point in any sequence the group has reached. In his plan recognition approach, Bui [10] also assumed that the Hidden Markov Model has a memory, to enable the model perform the task. In this present formalism, the Individual and Group Models are basic data storages of attributes relating to past interface contents and activities, stored as probable states of the individual user or the group as a whole (See Section on Learning Objective).

Structure of Formalisation

The formal language developed here enables the specification of agent support for scaffolding in learning. The following sections provide an outline of the aspects of analysis that can be performed by the agents on the learning context. This is not complete, but provides a view of the scope of this analysis. This formalism has been designed and implemented in an existing system, hence some of the difficulties of definition and categorisation have been resolved.

Content Formalism

The entities that are analysed by the agents are the variables and the constants of the agent language. The entities provide a way to model the information on the screen or in files. This forms a representation of the learning state of the student or group.

1. User: The first variable is the *user* which is an element of the set of *Users* who are online. This set is developed by the server. The attributes of a user are *User.name* and any other data that is stored in their Individual Model.
2. Files: The files that belong to the group are the entities *File* and refer only to the files read/writeable by the tool agent that is requesting them. The file attributes can be specified as *File.open* or *File.closed* or *File.all*, to select those files that are active at present or to check all local files. The *File.group* form is used to specify those files that are saved in the repository as group-verified files.
3. Initialisation: The pattern analysis is initialised when the tool starts (Tool.onstart), when a file is opened (Tool.onopen) or closed (Tool.onclose), or periodically (Tool.permanent) on a given time period, or when called by Course Planner (Tool.planned).³
4. Time: Time is a variable that can be related to Now, beforeNow and afterNow.

³The planner initiates the plan agents, when a task is reached that matches the chat Topic or a milestone is reached.

5. Objects: The Objects created (eg text or diagram) are recorded as records. The format of the records is listed in configuration files to be used at initiation of each tool.

These records give the form of the data records for the current tool, such as *Line* for the Chat (including attributes *Line.token*, *Line.tokenNumber*, *Line.topic* and *Line.username* to extract the token, tokenNumber, topic⁴ or username of the line).

All records are recorded by the Activity log with a time and Record.create or Record.delete component giving the username of the user doing this activity.

6. Template: Any template file is developed using the record structure from above, so that the template and the student's files can be compared using these rules and entities. For instance a Course Plan is in the form of *Plan.tasks* and *Plan.milestones*.

Entities or objects can be combined in many different ways. For instance they can be searched for relationships between entities (\rightarrow , \leftarrow , $<$, $>$, \equiv , \neq). Also they can be searched for entity attributes such as *Line.topic*==“Executive Summary”.

Each Also we need to represent the constant attributes. For instance *Line.token* is an element of the set of tokens as specified for the particular groupware chat tool.

$$Line.token \in \{ "RequestInfo", "Explain", "Rephrase", "Disagree", "Acknowledge", "Clarification" \}$$

Activity of interest

In this formalism the events associated with learning are the conditionals that are searched in the interaction and learning process. These conditionals form the guard of an event. When the event is achieved, action is taken by the agent. For instance, the iterative form of event guard is given as:

$$pre(Conditional(process, step - 1), t - 1) \wedge \\ occurs(Conditional(process, step), t)$$

⁴topic refers to selected topic of contribution, which is usually a list of tasks in the planner

$\Rightarrow \text{exists}(\text{file}, \text{process})$

When a sequence or pattern of activity is established, such as a student asks for information in Chat, then the agents can verify if the pattern is completed with an “Explain” or no “Explain” over a certain interchange time.

The files are searched for conditionals relating to their record entity structure. These conditionals are associated with the objects of a particular tool so each tool is described as an attribute of the generic Tool, such as Tool.Editor. For instance:

$\text{conditional}(\text{Tool.Editor}, \text{ExecutiveSummaryLength})$
 $\Leftarrow \text{exists}(\text{file}(\text{Tool.Editor}, \text{File.open}) \wedge$
 $\text{Section.header} == \text{“ExecutiveSummary”}) \wedge$
 $\text{length}(\text{Section.header} == \text{“ExecutiveSummary”}) > 20[\text{lines}]$
or
 $\text{length}(\text{Section.header} == \text{“ExecutiveSummary”})$
 $< \text{length}(\text{Section(header)} == \text{“Requirements”})$

This is the same as the *event* (x,y) form used in Akhras and Self [8].

Each event provides the role of that object in the activity. In this case the role of the section is to have a certain length. It would also have a certain content description, which would be provided to the system in the Document Template pattern.

Entities can be searched to find comparisons of their records. In DFD the shapes can be searched for proximity:

$\exists \text{variable}.(\text{variable} \in \text{Shape} \wedge$
 $\text{variable.bounds} - \text{variable.bounds} < 10[\text{pixels}])$ (pixels assumed)

Objects which are added then deleted over a short period can be verified by analysing entities changed and comparing the time record.

$\text{change}(\text{Tool}, \$\text{record}, \text{user}, t) \equiv$
 $\text{add}(\text{Tool}, \$\text{record}, \text{user}, \text{entity}, t)$
 $\vee \text{delete}(\text{Tool}, \$\text{record}, \text{user}, \text{entity}, t + \delta)$
 $\vee \text{move}(\text{Tool}, \$\text{record}, \text{user}, \text{entity}, t + \delta)$

These searches or checks are used by the agents to set the conditionals for action rules. Once a conditional is reached, the agent provides a *return* object. The return may be in the form of a list of items found satisfying the conditional. The return is used to determine the action of the agent to be proposed. The final action taken may also depend on the Group Model which stores the accumulation of past actions in terms of probable states.

Sometimes the actions require further pre-conditions or invariants, where the action also depends on various state or process factors. These are expressed as iterations of conditionals, but depend on values stored in the Group Model. For instance in the analysis of use of a particular Concept in the Lexicon:

$Conditional(Action(concept), depth(concept) - 1)$

$\Rightarrow write(depth(concept), all)$

or

$Conditional(Action(concept), context(concept))$

$\Rightarrow write(context(concept) + 1, all)$

where the previous *depth* of activity reached on that concept is retrieved from the Group Model. The rules for development of the database for the Group Model are the constants of the formalism.

Learning Objective

The Learner Models (Individual and Group) keep track of all the action data over time and is used to specify the exact response of agents in a given context over time. This includes the states that may have been reached, in learning, such as *depth of activity* on concepts as $depth(keyword)$ and level on DFD as $level(DFD)$. The Learner Models are created by the agents in the Blackboard style (see Chaib_Draa et al. [11]) using a communication database for recording the probable state.

The information about the Learning Objective and the state in this planned process is used by agents to assist in deciding the level of feedback to be given, the priority of certain feedback relating to processes which may be near completion, and feedback that should be combined if possible.

In the modelling formalism as expanded below, the time variable is discrete. In fact each step in time is signified by the pointer on the log file moving to the next line and the new line being read.

- Depth: $occurs(depth(concept), t)$
 $= occurs(depth(concept) - 1, t - 1) \wedge occurs(concept, t)$
- Level: $level(diagram) = \max_{Tool.DFDTool} Process.level$
- State: $occurs(state(process), t) \Leftarrow occurs(pre(state(process)), t - 1)$
 $\wedge occurs(conditional(All, process), t)$

- Combine Action: $Action(event1) \subseteq Action(event2)$
 $\Leftarrow summarise(Action(event2).role, Action(event1).role)$
 $\veertextend(Action(event2).role, Action(event1).role)$

Another aspect of the agent language that relates to the history or development of the modelling database, is the functions that define the searches to be made. Searches are made of entire files to verify activity in terms of various rules. For instance, if a document is being searched for records that satisfy a conditional, the function $find(conditional, list)$ finds elements from the list (which may be files) which have the conditional being true and returns them as the *return* list.

In addition, $length(return)$ provides data on how many items in a list, once a search has been made and the return of a search saved as a list. Also $last(x, return)$ and $first(x, return)$ returns the last and first x elements of the return list.

Learning Activity and Group Interactions

In this section we look at the analysis of the actions in terms of the learning activity and group interaction which is able to be extracted for agent analysis. These are described by functions of variables and constants.

The cognitive structure of the students is what a student or group *knows*. This can be treated as a summary of concepts and experience that have been encountered, and possibly repetitively. This provides the opportunity for them to modify their cognitive structures (following approach of von Glaserfeld [19]). Linkages of events through time sequence or linkage of object through proximity (such as opening two levels of a DFD at once to compare) show this learning activity.

The areas of Activity theory covered in this section are the tasks, including the Steps performed which suggest cognitive links or structures, the Division of Labour or interaction of the students and the Co-ordination of labour in the sense of learning processes analysed.

Analysis of Cognitive Structures

This is a model of the knowledge relations students make, by adding information or links. This occurs as events through editing the objects, or through

thought by opening records of past events and reviewing.

The graphical tools incorporate the symbols used in that course, for instance DFD's use labelled and unlabelled arrows, ovals and rectangles. These form the entities of the rule analysis, where the semantics of the graphical representations and important aspects in the diagram can be checked and recommended.

As for graphical constructions, there is a set of rules for a general text document layout which should be adhered to, but these should be made more flexible by using an adaptive model to analysis the content of the sections. This model would be applicable to any report writing domain.

Analysis of Interaction Division

The process analysed the division or difference between the interactions of students, for instance their different participation levels. The learning primitives look at participation aspect in terms of how much an individual contributed to events or object editing, or within what sort of interaction environment the learning occurred (conflictive, rapid exchange or long reflective exchange).

From the Collaborative Learning Taxonomy used for providing token in the Chat tool, we get interaction patterns which may be present or absent in various combinations. These can be combined to deduce the value of the interaction for learning. In particular conflict and participation are aspects which affect learning.

Analysis of Learning Co-ordination

The analysis so far has been collecting the data on student acts, and the next step is to analyse these acts for threads that may link to specific conceptualisations of the more complex learning. These include working in a group, decision making, designing graphics, developing the documents and linking diagrams with documents.

Analysis of Cognitive Change

The simplest forms of cognitive change to analyse are those relating to the use of new or old concepts or linking new with old. This usually occurs as a result of reflection.

A more appropriate approach is to look at the cognitive change that occurs as a result of the output of the agent. If the agent has correctly identified the possible learnign state and provided suitable feedback, the student will gain in their learning. The proof of the agent or its verification would come from analysing a cognitive change that matches the pattern of the required cognitive change specified in the pattern of that agent. Hence we must look at the language used to specify the action for change.

Cognitive Structures

The type of links between knowledge that need to be represented are presented in formal language here as:

Common threads running between the tools

$$file1 \mapsto file2 \Leftrightarrow find(file1, concept) \equiv find(file2, concept)$$

and

$$concept1 \mapsto concept2 \Leftrightarrow$$

$$\exists token.(token \mapsto concept1 \wedge token \mapsto concept2)$$

Timing compared to course planner

$$Task.description = concept \wedge occurs(concept, t) \wedge t < Task.end$$

Templates used to compare to student documents and diagrams $\forall var.(var \in$

$$Records \wedge var \in file1 \wedge file1 \equiv template1$$

$$\wedge template1 \in Template \Rightarrow var \equiv Template.record$$

Rules adhered to in documents including graphics

$$\forall file.(file \in Tool.Editor \Rightarrow Editor.rule(file))$$

The rules are defined as sets of rules for each tool. The rules in diagrams include aspects from the design pattern. If a design pattern record is names “Search” and its attributes as: Tool searched == tool, object found = record, search similar = attribute1, search dissimilar = attribute 2. Rules can be expressed as

$$\forall var.(var \in Search.tool \wedge var \in Search.record$$

$$\wedge var.attribute1 = Search.attribute1 \wedge var.attribute2 = Search.attribute2$$

Interaction Division

There are interaction patterns that can be analysed simply from the Chat tokens. For example these patterns include occurrences of “Request Info” followed (or not) by “Explain” or “Rephrase” from another student with a different username.

$$\begin{aligned} & \forall a.(a \in Line \wedge \\ & a.name = user \wedge user.token \equiv \text{“RequestInfo”} \wedge \\ & \exists u.(u \in Line \wedge u.name \neq user \\ & \wedge u.token \equiv \text{“Explain”} | \text{“Rephrase”} \wedge \\ & u.target \equiv user.token) \Rightarrow append(a, return)) \end{aligned}$$

Other patterns such as Criticism can be seen in document and drawing tools from the pattern of insertion and deletions.

$$\begin{aligned} & (\sum_a(a \in Shape \vee Paragraph \wedge a.create = user \wedge \\ & \exists u.(u \in User, u.name \neq user \wedge a.delete \equiv user)) > 10) \\ & \Rightarrow append(a, return) \end{aligned}$$

Learning Co-ordination

The modelling of the group from the individual models is now described for each of the attributes of the Learner Model.

1. State:

$$\textbf{Decision State } decisionState.group \equiv min(decision.individual)$$

$$\textbf{Decision Contribution } decisionContrib.group \equiv user.name \mapsto decisionContrib.individual$$

$$\textbf{Course State } courseState.group \equiv max_{time} courseState.individual$$

$$\textbf{Design Complexity } design.group \equiv max(design.individual)$$

$$\textbf{Agent Weight } agentWeight.group \equiv max(agentWeight.individual)$$

2. Context:

$$\textbf{Role } role.group \equiv user.name \mapsto role.individual$$

$$\textbf{Concepts } concept.group \equiv \cup_{user.name} concept.group$$

$$\textbf{Depth } depth.group \equiv average(depth.individual)$$

3. Action:

Interactions $participation.group \equiv min(participation.group)$

Files $files.group \equiv files.individual_{librarian}$

Windows $windows.group \equiv \cap_{user.name} windows.individual$

These values are available for the user to view, however the more useful aspect of the open modelling comes with the ability of the student to comment on the feedback given based on the system modelling of their learning. The difficulty is that the data as stored in the model is not directly informative, however the actions taken by the system based on this data are either useful or not, and this is where student comment is sought. We now look at the cognitive change as a result of agent action.

Cognitive change

The basic change is the replacement of old ideas with new.

$new(concept, user, t) \Leftrightarrow \exists change(tool, concept, user, t)$

$\wedge \forall tj.(ti < t \Rightarrow \neg occurs(context(concept), tj))$

$\Rightarrow \neg occurs(depth(concept), ti))$

The cognitive aspects that are changing are the depth of activity and the context in which concepts occur. These constant functions are developed in terms of the language variables and the Learner Model entities.

Depth of activity is a mapping from the total occurrence of a concept to the number of states of learning for that concept. For instance if the concept *Requirements* has been given 4 levels then the depth of understanding by a group of that concept is a mapping from the number of occurrences of *Requirement*, or the number of edits of the Requirements section in the report, to the numbers 0 to 3.

$depth(concept) \equiv total(concept) \mapsto numLevels$

$wheretotal(concept) \equiv \sum_{t1} \sum_{Tool.all} (occurs(concept, t1))$

Also the change initiated by agents is a result of the action by the agent. The various actions developed so far are:

1. $write(target, tool, token, textString)$ — writes “token: textString” to tool in target screen
2. $mailto(target, textString)$ — mails textstring addressed to target
3. $write(model, attribute, textString)$ — saves textString to attribute in Learner Model

4. `implement(toolAgent, textString)` — implements agent for tool with `textString` for information

Cognitive Process

As the activity is modelled over time, the entities or variables change with the events. To preserve the history of the interactions, or the process and patterns of the activity, we focus on the sequence of actions. These patterns are the model of the students activities in the context and the cognitive structures that are developed as a result of these activities. It is the linking of context and cognitive structures by activities that provides the opportunity for student learning, and for computer analysis.

The level of scaffolding that the system should provide for students to guide their intentional learning should diminish throughout the course (see Brown, Collins & Duguid [9]). Also, there are aspects of the learning that may be introduced to encourage further development when part way into the course. In particular, the need to encourage conflict should diminish as the design is developed and the report construction progresses.

However, there would need to be some assessment of where the group is at, what level of understanding of the course concepts has been achieved, or simply where they are at in their design. This could then be matched with data from the domain data as to what action to be taken at various states of learning.

The analysis of the development of the cognitive processes is gained from the history recorded in the Learner models and the development of the documents on which the group is working.

The communication between software agents resembles the Blackboard model and human agents also co-operate on the group interface which resembles a Blackboard model of the project outputs. In both cases the blackboard is being used to solve the problem which no single agent (human or software) is capable of solving alone.

Determine the cognitive processes

The states which are conducive to learning according to Akhras and Self [4] are: cumulative, constructive, self-regulated, reflective and attentional processes. Processes can be modelled in the collaborative software domain as

having the following properties:

Cumulative Repeated encounters with a concept or learning process which is analysed by an arrangement pattern.

Constructive Applying knowledge to new situation by adapting or using previously developed materials which is analysed by a transformation pattern.

Group-regulated Accessing course resources or linking to either group documents or examples from other projects which can be analysed by induction or deduction patterns.

Reflective Accessing learning model interface, logs of interaction or version tree which can be analysed by reflective pattern.

Attentional An abstraction of the discourse participants' focus of attention, records the objects, properties, and relations that are salient at a given point in the discourse.

These are developed through the different agent types. As well as analysing the processes specific to each state of the learning activity, these categories are the general properties of constructive learning that need to be supported. The interface can assist by providing the facilities for following such processes, and keep track of whether students or groups have used them.

The types of patterns of structural development analysed in learning are expressed in the formal language below. It should be noted that the agents are interested in looking for the occurrence of any such developments, rather than a development relating to specific keywords.

Cumulative Aspects

Students repeatedly return to certain parts of the project for further changes. Also, when saving versions of their work, students repeatedly refer to an aspect of the design which is requiring some thought to work out. The latter requires the implementation of a domain limited subject list for version descriptions to reduce the computational load of analysing the cumulative aspects.

When students are seen to access similar information or use similar concepts in their design or report writing, it may be assumed that they are

accumulating knowledge about that concept. The model can keep track of the concepts that are accumulated and their relation to the above aspects of learning.

In decision making the sort of concepts that may be considered are the use of disagreements or the providing of answers to improve common understanding and the resolution of conflict in terms of effective design changes (which are not changed in later sessions).

Arrangement :

$$\begin{aligned} &\exists keyword1.(keyword1 \in (Paragraph.text \vee Shape.text) \wedge \\ &\exists keyword2.(keyword2 \in (Paragraph.text \vee Shape.text) \wedge \\ &keyword1 \mapsto keyword2)) \end{aligned}$$

where " \mapsto " is a mapping in the tool structure, either by XML entries in the documents or links in a diagram.

Constructive Aspects

At present the cut and paste facilities, especially between versions, and using the linking structure within a document, is the only access available through the present analysis to this aspect of learning.

As students develop links both within their document and between different documents they may access or save. The concepts that are linked can be assumed to form a construction of knowledge. In decision making, this would include referring back to previous decisions, revisiting old versions and relating conflicting ideas to their resolution.

Transformation :

$$\begin{aligned} &\exists keyword1.(keyword1 \in (Paragraph.text \vee Shape.text) \wedge \\ &\exists keyword2.(keyword2 \in (Paragraph.text \vee Shape.text) \wedge \\ &keyword1.level < keyword2.level \wedge keyword1.deleted \\ &\Rightarrow keyword2.created)) \end{aligned}$$

where there may be different forms of transformation depending on the tools used.

Group-Regulated

Various resources relating to the domain knowledge can be provided to students through the groupware and accessing these links can be recorded, and information provided to the student on their access of these resources. These

resources may be provided by the course, the material the student has found, or previous versions of the document with editing.

The Group Model provides the resource for keeping track of group activities. These activities are the sum, difference or maximal value of the Individual models plus specific group data. The analysis of these links is done by deduction and induction patterns for which examples are expanded as:

- Deduction (Text): $\exists keyword1.(keyword1 \in (Paragraph.text \vee Shape.text) \wedge occurs(keyword1, t) \wedge (keyword1.level \Rightarrow keyword2.level)) \wedge occurs(keyword2, t)$
- Induction (Planner): $\exists task.(task \in Tasks, task.end < t1) \wedge t < t1 \wedge occurs(task.finish, t)$

Affordances

The first aspect of affordances is how the agents relate and develop from each other. Through the formal language this is expressed as refinements, decompositions and generic instantiations. These relations in the language

As a learner completes activities or types of group interactions, they effectively move from one level of control and abstraction to another, the language shifts, or incorporates features of the new level of abstraction. This process is said to raise the *cognitive floor*⁵ of their actions in the domain, reducing the cognitive load.⁶

The pattern structure to describe student activities followed this development or refinement (see summarise and extend functions in Section on Learning Objective). As students actions match higher level patterns, or all the sub-level patterns of a higher level pattern, then the concepts involved in the higher level pattern can be used for generating advice responses.

The structure in the learning and interaction analysis has the following features:

⁵The level at which a learner understands a domain is their cognitive floor, on which all other knowledge of the domain is built.

⁶The cognitive load is the effort required to keep distinct ideas in working memory. By combining a group of ideas into a concept, the group of ideas reduces to a single entity in memory.

Extend Rules can be an extension or refinement of a more general rule by making a deeper analysis of the occurrence of an approach or process.

Implement A more complex rule can use the analysis or part of the analysis of other simpler rules through decomposition in simple parts.

Generic Rules can be sub-pattern of a more generic rule that summarises their features in common. The generic rule provides a generic instantiation of the sub-rule.

These are expressed as:

1. `extend(RuleRefinement, Rule)` — same as *part(x,y)* relation as used in [8];
2. `implement(ComplexRule, SimpleRule)` — which includes analysis output from one agent in another; and
3. `generic(GenericRule, SpecialCaseRule)` — same as *kind(x,y)* relation as used in [8].

These relations between functions are used to provide combined feedback from agents. Where two rules are related by their role and enacted together in time, their feedback can be combined. The role of the Action is derived from the function and is an attribute of the Action.

Student interactions develop patterns of processes that are cumulative, constructive, group-regulating, reflective or attentive. After a sequence of learning actions several patterns of interaction hold, and hence a set of processes may be developing. These rules and processes characterise the state of the learning.

The possibilities for any future interaction to develop the rules or processes further are the affordances of the activity. Formalising these affordances allows the agent system to reason about the features of the activity that can be emphasised or de-emphasised to enhance deeper learning or more collaborative interaction.

Outcome

The aim in providing learning support is to change the learning environment to enable the next state of learning or interaction to be reached. There

are two approaches to changing the environment presented by Akhras and Self [5]. They are:

1. Analyse the patterns in the learning process that describe the opportunities provided for learning at that state (which may then be assumed in the learner model) and
2. Infer the learner's needs at a certain time which determine the environmental changes to be made to provide suitable learning opportunities.

In an open learning groupware system the environmental changes are mostly in the form of feedback, either of comments or alternate examples. Much of the scaffolding is based on case based reasoning in that it provides information about similar problem solutions in similar context. The aim is to enhance cognitive flexibility by providing multiple perspectives, themes or interpretations on the problem.

By providing agents that self-analyse the response to their action, we are able to assess the value of the agent scaffolding. If the opportunities are taken up by the learners or the learning problems are resolved, then the agent input may have been suitable. The agent can itself analyse whether the effect it was responding to is still present, and to what degree.

Alternatively the user is given the option of responding directly to the feedback in popup windows. For instance if the users chooses to 'Hide' the paperclip advisor, then the advise would seem to be redundant or incorrect. While the specific features of the group and individual model are retrievable, their significance is largely unintelligible to the user except in terms of the feedback they promote.

Conclusion

Analysis of groups working on-line has provided the list of structures which must be formalised in the learning domain and group work domain to enable the analysis of student contributions in group mode. The formal language also provides mechanism for linking and developing agent complexity through refinement and extension.

The difficulty in providing agent support for feedback is the variety of learning activities that students can undertake on-line and within various domains of learning. This paper classifies all such activities into a generic

language. The scaffolding can be developed from these, based on the patterns of students activities.

The significance of any pattern of learning which is used in setting the learning level within the environment, can be determined by modelling learning, and specified in formally written files or rules. The method of modelling learning can be simple (such as how often a concept is referred to) or complex (as in the analysis of Hidden Markov Models by Soller and Lesgold [31]).

Particularly in the more complex analyses, it is recommended that the user assessment of the model be indirectly through the user response to agent advise in terms of their subsequent learning and interactions.

References

- [1] Abrial J. R. The B-book: *'Assigning programs to meaning*, Cambridge University Press, 1996.
- [2] Abrial J.R. Event-Based sequential program development: Application to constructing a pointer program, in *FME 2003:Formal Methods*, September 2003, 51–74, Springer.
- [3] Abrial J.R. A formal (proved) approach to System Engineering: presentations and case studies. Seminar run NICTA Formal Methods Program Educational Activities. Retrieved September 2003 from <http://www.cse.unsw.edu.au/carrollm/Abrial>.
- [4] Akhras F.N, Self J. A process-oriented perspective on analysing learner-environment interactions in constructivist learning, *AAI/AI-ED Technical Report No.126, Proceedings of the 6th Brazilian Symposium on Computing in Education (SBIE'95)*, 1995, Florianopolis, Brazil, SBC.
- [5] Akhras F.N, Self J. (1996) A process-sensitive learning environment architecture, in *Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96)*, Montreal, 430–438.
- [6] Akhras F.N, Self J. (1997) Modelling learning as a process, *Proceedings of the 8th World Conference on Artificial Intelligence in Education (AIED'97)*, Kobe, Japan, 418–425.
- [7] Akhras F.N, Self J. (1997) Modelling the Process not the Product of Learning. Unpublished manuscript, Computer Based Learning Unit, University of Leeds.
- [8] Akhras F.N, Self J. (2000) System Intelligence in Constructivist Learning. *International Journal of Artificial Intelligence in Education*, **11** 4, 344–376.
- [9] Brown J.S, Collins A, Duguid P. (1989) Situated Cognition and the Culture of Learning. *Educational Researcher*,**18**, Jan–Feb, 32–42.
- [10] Bui, H. (2003) A general model for online probabilistic plan recognition. In *Proceedings of the International Conference on Artificial Intelligence (IJCAI)*.

- [11] Chaib_Draa B, Moulin B, Mandiau R, Millot P. (1992) Trends in Distributed Artificial Intelligence, *Artificial Intelligence Review*, **6**, 35–66.
- [12] Constantino-González M, Suthers D.D. (2000) A coached collaborative learning environment for Entity-Relationship Modelling, in G. Gauthier, C. Frasson and K. VanLehn, editors, *Intelligent Tutoring Systems, Proceedings of the 5th International Conference (ITS 2000)*, Berlin:Springer-Verlag, 325–333.
- [13] Constantino-González M, Suthers D.D. (2001) Coaching collaboration by comparing solutions and tracking participation, in P. Dillenbourg, A. Eurlings, K Hakkarainen, editors, *European Perspectives on Computer-Supported Collaborative Learning, Proceedings of the First European Conference on Computer-Supported Collaborative Learning*, Universiteit Maastricht, Maastricht, the Netherlands, March 22–24, 173–180.
- [14] Constantino-González M, Suthers D.D and Icaza J.I. (2001) Coaching web-based collaborative learning based on problem solution differences and participation, in J.D.Moore, C.L. Redfield and W.L. Johnson, editors, *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future (Proceedings AI&ED 2001)*, IOS Press, 176–187.
- [15] Constantino-González M, Suthers D, Santos J. (2002) Coaching Web-based Collaborative Learning based on Problem Solution Differences and Participation, *International Journal of Artificial Intelligence in Education*, **13**, Retrieved March 15 from <http://www.cogs.susx.ac.uk/ijaied/>.
- [16] Dillenbourg P, Self J.A. (1994). Designing human-computer collaborative learning, *AAI/AI-ED Technical Report No. 91*, in C. O'Malley, editor, *Computer-Supported Collaborative Learning*, Berlin, Springer-Verlag, 245–264.
- [17] Fjeld M, Lauche K, Bichsel M, Voorhorst F, Krueger H, Rauterberg M. (2000). Physical and virtual tools: activity theory applied to the design of groupware. B. A. Nardi and D. F. Redmiles, editors, *A Special Issue of Computer Supported Collaborative Work (CSCW): Activity Theory and the Practice of Design*.

- [18] Gibson, J. (1979). *The Ecological Approach to Visual Perception*.
- [19] Glaserfeld E. von. (1989). Cognition, construction of knowledge, and teaching, *Synthese*, **80**, 121–140.
- [20] Glaserfeld E. von. (1989). *Radical Constructivism: A Way of Knowing and Learning*, London: The Palmer Press.
- [21] Hoppe, H. U. (1993). Intelligent user support based on task models. In M. Schneider-Hufschmidt, T. Kuehme, & U. Malinkowski (editors), *Adaptive user interfaces*, pp. 167-181. Amsterdam: Elsevier Science
- [22] Kautz, H. (1990). A formal theory of plan recognition and its implementation. In J. A. Allen, H. A. Kautz, R. N. Pelavin, & J. D. Tenenber (editors), *Reasoning about plans*, pp. 69-125. San Mateo, CA: Morgan Kaufman.
- [23] Kutay C. (2003). Intertac Software Architecture, Technical Report No. 0318, Computer Science and Engineering, University of New South Wales, <ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/0318.pdf>.
- [24] Nardi B.A. (1996). Studying Context: A comparison of activity theory, situated action models, and distributed cognition in Bonnie A. Nardi, editor, *Contexts and Consciousness: Activity Theory and Human-Computer Interaction*, MIT Press, Cambridge, Mass., 69–102.
- [25] Mühlenbrock, M. Action Based Collaboration Analysis for Group Learning Volume 244 Dissertations in Artificial Intelligence, IOS PRes.
- [26] Riley, P and Viloso, M. (2000). On behavior classification in adversarial environments. In Lynne E. Parker, Georgy Bekey and J. Barhen (editors), *Distributed Autonomous Robotic Systems 4*, pp 371–380, Springer-Verlag.
- [27] Soller A, Goodman B, Linton F, Gaimari R. (1998). Promoting effective peer interactions in an intelligent collaborative learning system, *Proceedings of the 4th International Conference on Intelligent Tutoring Systems (ITS98)*, San Antonio, Texas, 186–95.

- [28] Soller A, Lesgold A. (1999). Analyzing peer dialogue from a active learning perspective, *Workshop at AI-Ed '99 9th International Conference on Artificial Intelligence in Education — Analysing Educational Dialogue Interaction: Towards Models that Support Learning*, Le Mans, France, 18th-19th July.
- [29] Soller A, Lesgold A, Linton F, Goodman B. (1999). What makes Peer Interaction Effective? Modelling Effective Communication in an Intelligent CSCL. Proceedings of the 1999 AAAI Fall Symposium: Psychological Models of Communication in Collaborative Systems, Cape Cod, MA.
- [30] Soller A. (2001). Supporting social interaction in an intelligent collaborative learning system, *International Journal of Artificial Intelligence in Education*, **12**, 40–62.
- [31] Soller A, Lesgold A. (2003). A computational approach to analyzing online knowledge sharing interaction. In U. Hoppe, F. Verdejo and J. Kay, editors, *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies (AIED'03)*, 253–260.
- [32] Sukthankar, G and Sycara, K. (2005) Automatics Recognition of Human Team Behaviour, *Proceedings of Modeling Others from Observations (MOO), Workshop at the International Joint Conference on Artificial Intelligence (IJCAI)*, July.
- [33] Vygotsky, LS. (1989). *Mind in Society: Development of Higher Psychological Processes*. Editors: Cole, M, John-Steiner, V, Scribner, S and Souberman, E., Cambridge, Mass: Harvard University Press, 1989.