Explorations in fine-grained learning analytics

CTIT
Centre for Telematics and Information Technology

IBR
RESEARCH INSTITUTE FOR
SOCIAL SCIENCES AND TECHNOLOGY

Typeset by the author using LaTeX

ISBN: 978-94-610-8307-4

EXPLORATIONS IN FINE-GRAINED LEARNING ANALYTICS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. H. Brinksma
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op woensdag 6 juni 2012 om 16:45 uur

door

Anjo Allert Anjewierden
geboren op 20 juli 1957
te Usquert

Dit proefschrift is goedgekeurd door de promotoren:

Prof. dr. A.J.M. de Jong
Prof. dr. R. de Hoog

Voor mijn moeder

Preface

It is a little weird that, after finishing the thesis, there is still some writing to be done. First of all, I would like to thank Ton and Robert. Ton started the whole process and persistently ensured it kept on moving in the right direction. Robert excelled at getting the product right. Our conversations were on topic most of the time, although we did discuss soccer and the coffee quality from time to time. Thank you very much gentlemen.

A necessary ingredient to make progress in science is collaboration, and in the past 30+ years there has been a lot of that. Jan Wielemaker has documented our endeavours at the University of Amsterdam in his thesis in an entertaining way. Looking back, it is amazing that the software we initially developed in the 80s and 90s of the previous century is still of great value today. Bob Wielinga has inspired many to reach a higher level. Fortunately, I was one of them.

I did get distracted by several colleagues who had a similar problem, and it was not difficult to decide what had the highest priority. Petra, Bas, Wout, and Mieke thanks for having the opportunity to work on your data. My heroes are Nadira and Yvonne who collected and coded data analysed in the pages that follow, and Hannie and Robert who helped with additional coding. Another distraction was work on the SCY project. Isabelle, Rachel, Margus and Costas helped with specifying the concept mapping agents. Stephan, Jan, Lars and Jörg provided the necessary infrastructure. Thanks guys!

I thank Alieke, Danish, Frank, Judith, Marjolein, Mieke, Nico, Wout and Yvonne for the honorary membership of ProIST with limited duties: drinking tea, eating cookies, wining and dining, writing Ph.D. songs while enjoying a bokje (with Sylvia and Wout), and organising cycling trips (with Hannie, Sylvia and Irina).

Finally, I want to thank Daphne for the secretarial assistance, and Johan, Jakob, and Henrik Rinse and his mama for help with the cover.

Acknowledgement

Part of this thesis was conducted in the context of Science Created by You (SCY), which was funded by the European Community under the Information and Communication Technologies (ICT) theme of the 7th Framework Programme for R&D (Grant agreement 212814). This document does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of its content.

Contents

Chapter 1

Introduction

Data about students and their learning process is recorded by electronic learning environments. Learning management systems (LMSs), such as Blackboard and Moodle, register grades on assignments, examination results and, for many courses, the resources learners have visited. Interactive learning environments for specific domains, for example cognitive tutors or inquiry-based simulation and modelling tools, record all actions learners perform and the products they create. In this way learning generates large amounts of data that can form the basis for adapting learning environments to individual learners. Such adaptation is what is sought, especially in open inquiry learning environments, as can be read in the following quote:

> "The promise offered by inquiry learning [in which students actively discover information] is tempered by the problems students typically experience when using this approach. [...] A challenge lies in adapting the learning environment to respond not only to differences between learners but also to the developing knowledge and skills of the individual learner. [...] Automating this would need an adequate cognitive diagnosis of both a student's learning process and developing knowledge and might be based on the log files of the student's interaction with the system." de Jong (2006, p. 532–533)

Inquiry learning environments encourage students to discover underlying phenomena through scientific inquiry processes like defining a hypothesis, performing an experiment, interpreting the results and drawing conclusions. Students find inquiry learning difficult. The environments offer a lot of freedom and students have to think about both the domain of learning as well as following the inquiry process: "data gathering, analysis, interpretation, and communication are all challenging tasks that are made more difficult by the need for content-area knowledge" (Edelson, Gordin, & Pea, 1999, p. 399). Unassisted inquiry learning is not effective (Mayer, 2004) and inquiry learning environments therefore provide guidance and scaffolds to increase the effectiveness of learning. A recent study (Eysink, de Jong, Berthold, Kollöffel, Opferman, & Wouters, 2009) and a meta-study (Alfieri, Brooks, Aldrich, & Tenenbaum, 2011) in which inquiry learning with additional support is compared to other educational approaches found that inquiry learning is more effective.

Given these findings, we expect that the effectiveness of inquiry learning environments can be further improved when adaptation is decided on dynamically, based on an analysis of learner activity. The general idea is to develop analytics software, called *pedagogical agents*, that continuously monitor and analyse the activity of learners with the objective to adapt the learning environment to the learner when this is appropriate. Results of the analysis can be presented to the learner in different ways: the activation of scaffolds or prompts, or by visualizing certain aspects of the learning process. In addition to tracing the activities of learners, pedagogical agents can evaluate products of the learning process (e.g., models students created), and compare these products to products of peers or normative reference objects.

Actions students perform in inquiry learning environments are stored in log files and the analysis of this data can help to understand how students use a particular inquiry learning environment and what kind of adaptation might be appropriate. Log file analysis is therefore a prerequisite for the development of pedagogical agents. More broadly, the analysis of educational data has gained considerable attention in recent years and two, closely related, research communities have emerged: *Educational Data Mining*[1] (EDM) and *Learning Analytics*[2] (LA). Educational data mining is described as "the area of scientific inquiry centered around the development of methods for making discoveries within the unique kinds of data that come from educational settings, and using those methods to better understand students and the settings which they learn in" (Baker, 2010, p. 548). Learning analytics is described as "the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs" (Siemens, 2011, online). These definitions indicate that both disciplines aim to understand what learners do based on the traces they leave behind. In EDM this understanding is generally achieved by applying data mining techniques (clustering, classification, prediction, association rules, sequential pattern mining, text mining), and discovering relationships that describe some aspect of learning behaviour. The survey of Romero & Ventura (2007) and the Handbook of Educational Data Mining (Romero, Ventura, Pechenizkiy, & Baker, 2011) provide an overview of the application of data mining techniques to educational data. Eductional data mining starts with the educational data and tries to reveal the patterns

[1]`www.educationaldatamining.org`
[2]`www.learninganalytics.net`

it may contain. Learning analytics is broader in scope, it is "a holistic approach that combines principles of different computing areas (data and text mining, visual analytics and data visualization) with those of social sciences, pedagogy and psychology" (Ali, Hatala, Gašević, & Jovanić, 2012, p. 470). Learning analytics is more result-centered than EDM. Learning analytics starts with a motivation of what to search for and for what purposes the outcome could be used. A natural outcome of learning analytics are abstracted overviews and visualisations of the findings.

Data resulting from learning can be thought of as being "coarse" or "fine grained". An example of *coarse-grained* data is a grade for a course. This type of data is coarse grained because all the intermediate steps the learner took to obtain the grade are unknown, and can therefore not be analysed. Of course, coarse-grained learner data itself can be analysed. For example, by applying data mining techniques such as relationship or association mining, to investigate whether students who obtained high grades in one course also obtain high grades in other courses (e.g., Romero, Ventura, Espejo, & Hervas, 2008). In learning analytics, coarse-grained learner data is frequently related to other indicators that are available, for instance grades on courses related to activity in social media or prior education. *Fine-grained* data, on the other hand, consists of a "complete" trace of the activities the learner performed. Depending on the learning environment, types of actions may include selecting variable values in a simulation tool, chat messages in a collaborative environment or answers in tutoring systems.

A survey of the proceedings of the latest educational data mining (EDM 2011; Pechenizkiy, Calders, Conati, Ventura, Romero, & Stamper (2011)) and learning analytics conferences (LAK 2011; Long, Siemens, Conole, & Gašević (2011)) illustrates the differences between the two communities. All twenty full papers at EDM 2011 used data mining methods, and most papers motivated the research to improve on previous applications of data mining. Ten papers used visualisation to present the results to stakeholders (individual learners, instructors or learning institutions) and three papers mentioned "recommendation" (of learning objects or peers to cooperate with). All twenty full papers presented at EDM 2011 used fine-grained analysis, sixteen related to student modelling in tutoring systems. Of the 27 papers at LAK 2011, seven used fine-grained analysis (log files, chat analysis), nine coarse-grained analysis (data from learning management systems or learning object repositories), two used both fine- and coarse-grained analysis, and the remaining nine papers were either theoretical or provided a framework without data analysis. In conclusion, the educational data mining and the learning analytics communities are primarily concerned with data analysis and visualisation to understand learner behaviour. Some learning analytics research tries to change the behaviour of learners through visualisation of learner activities. Based on the survey, there is no evidence of active research into dynamically changing the learning environment to fit the needs of learners.

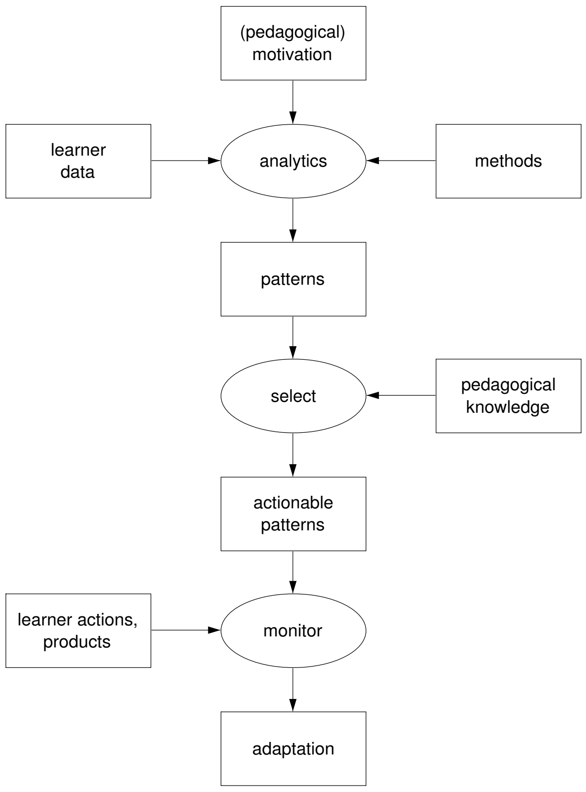


Figure 1.1: Steps to realise pedagogical agents. *Analytics* step finds patterns based on (historical) learner data. Pedagogical knowledge *selects* patterns that are actionable. Agents *monitor* learner activity and base adaptation on the actionable patterns.

In this thesis we investigate how the analytics of fine-grained data resulting from inquiry learning environments can be used to initiate adaptation of the learning environment. Figure 1.1 gives an overview of the steps required to realise this. Methods are used to find patterns in the data sets resulting from learning. To make these patterns actionable, pedagogical knowledge is applied to select relevant patterns and link these patterns to the appropriate adaptation for learners. Finally, the actionable patterns are implemented in a pedagogical agent and the agent monitors learner activity for the occurrence of the actionable patterns and initiates adaptation of the learning environment. The realisation of pedagogical agents therefore depends on several aspects: discovering patterns in data sets related to learning environments, selecting the patterns suitable to base adaptation on, and detecting the occurrence of the selected patterns.

The approach sketched in Figure 1.1 readily applies to much of current learning analytics research. A coarse-grained example is the Signals system developed and used at Purdue University (Campbell, 2007). Signals collects data from the learning management system on course materials used, sessions attended, participation in discussions, and so forth. This data is then related to students' test scores and historical data of previous students resulting in a prediction of how well a student will perform. The patterns are assessed by teachers, and are depicted as a "traffic light" (green, yellow, red) visualisation to students.

Section 1.1 presents an overview of the types of data generated and manipulated in learning environments. Learners perform actions, produce objects during the learning process and collaborate with their peers. The data that results from these activities provides a baseline to which data mining and analytics can be applied. Section 1.2 describes how the results of the analysis of learner data can be used for adaptation. Section 1.3 reviews methods which contribute to the realisation of pedagogical agents given the types of data available. These include general techniques such as data mining and text analysis, as well as log file analysis. Finally, in Section 1.4 we summarize this chapter and give an outline of the remainder of the thesis.

1.1 Data

In this section we provide an overview of the data related to learning and learning environments. Baker (2010) uses a distinction based on the context in which the data was generated and distinguishes keystroke, answer, session, student, classroom, and school level data. We make a distinction between the processes underlying the generation of the data and the types of data. This distinction is motivated by the analysis envisaged: tracking the learner interacting with the learning environment, possibly based on information about the learner (the process), and evaluating the results of learning (the products).

1.1.1 Where the data comes from

The two main sources of data related to learning are data about learners, and data that results from the interaction of a learner with a learning environment. Most learning environments store a detailed record of learner actions in log files (Hulshof, 2004) and tools inside learning environments keep track of both actions and the learning objects created as a result of the learning process. In collaborative environments, chat logs track the interactions between learners. Sometimes observational data (video, audio, eye tracking) is collected to supplement the analysis of log files (Dyke, Lund, & Girardot, 2009). Log files represent what the learner has done in the learning environment and given that they capture the "behaviour" are a primary source for analysis.

Information about the learner in general (age, gender, etc.), assessments (e.g., test scores, skills), learning histories (e.g., courses taken, classes), and presence and activeness in social media can be used to ground the process data in the log files. A common method in the learning sciences is to measure the "quality" of the behaviour in the learning environment by calculating the differences on tests before and after interaction with the learning environment. Analysis techniques can use these "quality" measures to find patterns in the log files that explain the difference in learning outcomes.

A special case is when the learning environment and assessment are intertwined. This happens when the learning environment consists of quizzes or other drill-oriented tasks (e.g., spelling a spoken word, solving a small problem, web-based courses). In these cases, the process data primarily consists of the time taken for an assignment and (the correctness of) the answer. These types of learning environments are popular in the EDM community, the process data has a relatively simple structure, comes in large volumes and can be analysed with a variety of data mining techniques (e.g., Baker, Barnes, & Beck, 2008). The Pittsburgh Science of Learning Center's DataShop[3] hosts a publicly accessible repository of such data sets (Koedinger, Cunningham, Skogsholm, & Leber, 2008).

Log files and other stored data are sufficient to support the offline analysis step in Figure 1.1. For the monitor step it is necessary that learning environments make learner actions and products available to agents in real time. More and more learning environments provide a communication infrastructure that makes this possible, see for example the blackboard software architecture described in Section 3.4.3.

[3]`http://learnlab.org/datashop`

1.1.2 What the data looks like

We distinguish four types of data related to learning that can be analysed: the learning process (activities), the objects or products learners create, communication and collaboration between learners, and data about learners (age, courses taken, etc.). The first three are the most relevant for our research and are described in further detail below.

Activities

Log files keep track of all learner-initiated interaction. They thus contribute to the "keystroke level" analysis (Baker, 2010) of educational data. Although there have been attempts to standardize the format of log files, e.g., Analog (Christoph, Anjewierden, Sandberg, & Wielinga, 2003) and Common Format (Martinez, Harrer, & Barros, 2005), the diversity of learning environments and the types of actions possible varies so widely that the lowest common denominator is to use a standardised machine-readable representation such as XML. Tatiana (Dyke et al., 2009), a tool for the analysis of computer supported collaborative learning (CSCL), also defines a proprietary format but includes filters that researchers can program to import their data.

Formally, we can view a log file as a chronologically ordered set of items where each item represents a learner action. For each item at least the following information is usually available.

Action type. The type of learner-initiated action. Common types are *answer* (to a question or an assignment), *add*, *delete*, *insert* (edit operations), *chat*, *run simulation*, *request hint*, etc. In quiz environments, the number of different action types is relatively limited (provide answer, request hint). Inquiry learning and simulation environments allow many different action types. The simulation environment SimQuest (van Joolingen & de Jong, 2003) generates more than sixty different action types, for example *start session*, *run assignment*, *change variable* and *open answer*.

Timestamp. The point in time the action occurred. Usually a precision of one millisecond is used to allow synchronization with observational data, e.g., video or EEG.

Learner. Identifier for the learner or group of learners.

Context. Most learning environments have a notion of context. The context can be related to the learning material, an assignment for example, the learning environment, a particular phase or sub-tool, or a combination of these. Context information can contribute to transition analysis, how learners navigate through the learning environment (Hulshof, 2004).

Attributes. Additional information that represents the necessary detail of an action. Obviously, there is a strong dependency on the action type. For example, a *change variable* action has the name of the variable and the new value as attributes.

The above information on learner actions potentially supports all standard types of static content analysis (frequency, coding), as well as analysis over time (sequences, transitions between contexts).

In practice, log files contain all actions the designer of the learning environment deems relevant to record. Mostow (2004) suggests to log actions at different levels of granularity to support different types of analysis. Given that it is difficult to even anticipate the types of analysis, it appears more appropriate to log at least all actions such that replay becomes possible. If the objective of the analysis is to determine learner behaviour at a more abstract level, actions not relevant to such analysis can simply be ignored. Another type of problem are actions that cause a state change in the environment. Suppose a learner wants to run an experiment with $k = 5$ and $n = 3$ (k and n are input variables). Setting these two variables may be represented as two unrelated actions in the log file, and pressing the *run experiment* button as a third. From the analysis point of view, the activity the learner wants to pursue is *run experiment*($k = 5, n = 3$). In the log file we might see *change variable*(k, 5), other actions, *change variable*(n, 3), other actions, *run experiment*, and the intended *run experiment*($k = 5, n = 3$) needs to be inferred from the action sequence.

In this section we have touched on several issues related to log files of learning environments. Technically, the representation of a log file is relevant. For computer-based analysis an XML-based representation appears most appropriate.

Products

In many learning environments students are given the task to produce something. These objects can be products of the learning process (e.g., essays, runnable models), or serve as an externalization or structuring mechanism of learner knowledge (e.g., concept maps, drawings).

Free text. Products represented as text are, for obvious reasons, very common. They can play the role of summary, report, essay, argumentation, and so forth.

Structured text. Forms or templates which the learner has to fill in are an example of structured text. Simple types are sentence openers, hypotheses and open answers, which are templates with a single field. Forms provide some guidance to learners, through the labels associated with the fields, and it may also be easier to analyse and compare learners based on forms rather than on free text.

Drawings, diagrams. Freehand drawings and diagrams can be used by learners to externalize their knowledge and help with self-explanation (Ainsworth & Iacovides, 2005). They can also be used in collaborative environments to exchange ideas with others or to obtain a common understanding (Gijlers, van Dijk, & Weinberger, 2011). Applications that support freehand drawings, for example FreeStyler (Hoppe & Gassner, 2002), are being integrated in learning environments.

Concept maps are a popular restricted type of diagrams or graph in which learners can structure their knowledge about a domain by defining relevant concepts and the relations between these concepts.

Models. Models are formal notations that are runnable. In learning environments the ability to run models is attractive because the learner immediately obtains feedback about the functioning of the model (van Joolingen, de Jong, Lazonder, Savelsbergh, & Manlove, 2005). A distinction can be made between environments in which the learner interacts with a predefined model, often referred to as simulation environments (e.g., KM Quest (Leemkuil, de Jong, de Hoog, & Christoph, 2003)), and modelling environments in which the learner has to construct a model for a given task. In simulation environments the learner's task is to understand how the underlying model works. The products are the sets of input variables the learner has manipulated. In modelling environments, the model created by the learner is the product. An example of a modelling environment is Co-Lab (van Joolingen et al., 2005) which is based on system dynamics and supports both simulation of predefined models and model construction by learners.

Data sets. A final type of product is a data set resulting from experimentation. Data sets can be an output of one tool and an input in another.

When several learners work on the same or similar tasks, object repositories result. These repositories can be used to track the progress of a single learner over time and also to compare the products of learners. In collaborative environments, the repositories can reflect progress of a group of learners and provide an opportunity for learners working together based on an analysis of their products.

Most types of products resulting from learning as listed above also occur in non-learning situations. This implies that for the analysis and evaluation methods may already exist elsewhere.

Communication and collaboration

Communication and collaboration facilities in learning environments provide a rich source of data and various opportunities for analysis. One popular method of analysis is social network analysis (SNA) (Scott, 1991) in which nodes reflect the actors (students) and edges represent the ties or social connections between actors. An example of SNA in relation to learning is "who replies to whom" on a discussion forum, the edges then represent the number of replies between students. SNA can contribute to the discovery of clusters or communities of students that share a relationship. SNA is a popular approach in learning analytics, seven of the 27 papers at LAK 2011 are about applying SNA and visualisation of social networks.

Communication and collaboration can also form the basis of content or semantic analysis. Following on from the forum example, one can try to determine whether there is a relation between the content (or topic) of forum messages and whether a given student replies. Often, collaborative environments provide a text-based chat facility and the messages students exchange can thus be analysed. Sometimes this analysis is simply counting the number of words, sometimes text analysis techniques are applied to understand what students communicate about.

Summary and discussion

In the previous sections we have presented an overview of the learner data that is available for analysis. The presentation has largely been logical: the different types of data sources and the role of these sources. The exact physical representation can be slightly different due to design choices and practical considerations. The simulation environment SimQuest, for example, represents an action type called *chat* which has the text of the message and the receiving peer as attributes. Similarly, during the development of a product, edit operations are generally sufficient to reconstruct the intermediate products. In some cases edit operations may have side effects, for example deleting a concept in a concept map causes the relations of the concept to disappear as well (without the student explicitly deleting these relations and as a consequence no *delete relation* actions).

1.2 Adaptation

This section addresses the question of how the results of the analysis of learner data (Section 1.1.2), can result in a change of the learning environment.

Adaptation based on analysed learner data can influence the learning environment and the learner in several ways. This can be implicit (changing the difficulty of the learning environment), directive (specific instructions to students), or informative (showing interaction patterns).

Implicit feedback. Based on the analysis of the data sources, the learning environment can be adapted without letting the student know. If a student makes many errors, the assignments can be made easier, or when students are working with a simulation environment, the number of variables can be reduced. Of course, the learning environment can also be made more challenging for students.

Directive feedback. Directive feedback is when students are given specific instructions. An example, based on product analysis, is suggesting missing elements in a model, or the suggestion to collaborate with a specific peer. In some intelligent tutoring systems, the student can ask for directive help by pressing a hint button.

Informative feedback. Analysis can also be used to provide students with a perspective on their own learning process. Informative feedback is usually visualised in a "dashboard". The indicators in the dashboard change dynamically depending on learner activity. Students are themselves responsible for changing their behaviour.

In the next two sections we describe interventions that occur during implicit or directive feedback and, visualisation as the primary method to provide informative feedback to students.

1.2.1 Interventions

In the learning sciences the term "scaffolding" is commonly used to refer to the adaptation of learning environments to match the skill and knowledge level of an individual learner. Scaffolds are add-ons in the learning environment that are not strictly necessary, but can help learners to focus on the learning process, e.g., a hypothesis scratchpad (Gijlers & de Jong, 2009). As mentioned earlier, some form of scaffolding is nearly always required in inquiry learning environments (Alfieri et al., 2011). Examples of how scaffolds are presented to the learner are prompts, a simplified user interface, or sequencing the order in which assignments or questions are presented. In current practice, scaffolds are permanent during a session with a learning environment. The challenge is to fade in and fade out scaffolds on the basis of activity patterns detected.

Interventions can be based on log data but also on the evaluation of products, sometimes in combination with activity analysis, particularly how long the student has been active. As mentioned in Section 1.1.2 products can range from short texts, such as open answers or hypotheses, to complex models. Evaluation can take place at the level of the structure of a product or analysing the semantics represented by the product compared to the domain of learning. An example of structural analysis is determining whether a hypothesis object contains terms that indicate it is a hypothesis (e.g., a conditional statement involving “if”, “then”). If it does not contain ”hypothesis like words” then the student could be prompted to think of another hypothesis, or the learner could receive a scaffold that contains a typical syntactical pattern, “if ... then ...” to complete.

1.2.2 Information visualisation

There are many “consumers” for visualisations resulting from learner data: the individual learner, (small) groups of learners, teachers, researchers, and even learning institutions. For individual learners an important purpose of visualisation is as an awareness indicator, for instance by visualising a state or how much progress is being made.

Visualisations for groups of learners contribute to awareness with respect to relations in the group. These indicators are often visualisations of some aspect related to the learning process. For example, Janssen (2008) has identified several problems in collaborative environments: lack of awareness (of other group members), communication problems (mainly caused by using a computer to communicate), coordination problems (focusing, engagement, agreeing, etc.) and lack of quality in the discussions. He proposes to use visualisations to partly address this, for example by a participation tool (see Figure 1.2) which aims to “affect participation through motivational and feedback processes” (Janssen, 2008, p. 37–38).

Learning analytics almost exclusively relies on visualisation to communicate information to learners. “For learners [..] it can be extremely useful to have a visual overview of their activities and how they relate to those of their peers or other actors in the learning experience” (Duval, 2011, p. 12). The role of visualisations in learning environments is to help learners better understand what they are doing. These visualisations can contain either a representation of the activity of learners, for instance the number of chats, or an interpretation of the results of the activity of a learner, for instance about the content of the chats.

A simple example of such an indicator, inspired by smileys, is an avatar representation of two learners in a collaborative environment (Anjewierden, Kollöffel, & Hulshof, 2007). The shape of the two learner avatars, see Figure 1.3, changes when they exchange messages in a simulation environment. Automatic chat analysis is applied

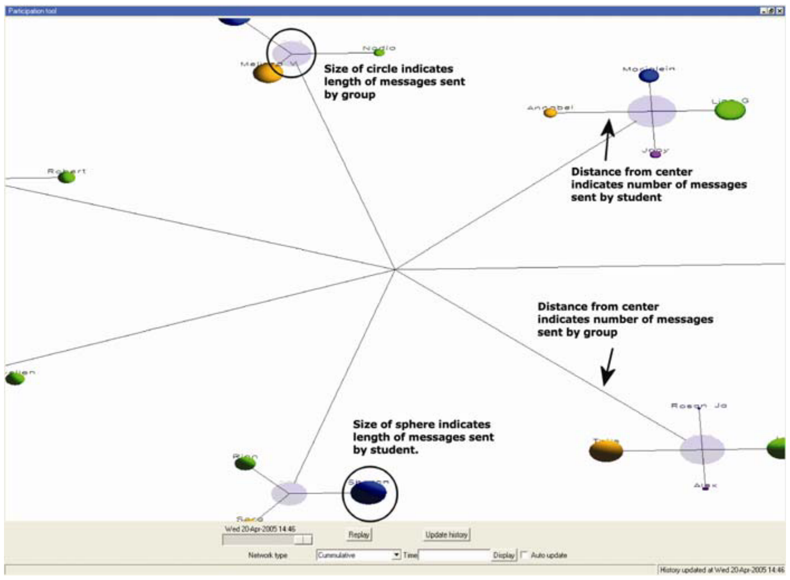

Participation tool
Size of circle indicates length of messages sent by group
Distance from center indicates number of messages sent by student
Distance from center indicates number of messages sent by group
Size of sphere indicates length of messages sent by student.
Replay
Update history
Network type
Cummulative
Display
Auto update

Figure 1.2: Participation tool showing a visualisation of the level of communication in a collaborative environment (Janssen, 2008, p. 45, Figure 2.2).

to classify the messages as one of domain (head), regulative (body), social (arms) and technical (legs). If a domain message is typed the head becomes larger. Pedagogically, the idea is that the learners reflect on the shape of the avatar, if the head is very small and the body is very large, the suggestion is to discuss the domain of learning more.

Visualisations that represent indicators of learner activity are called dashboards. Figure 1.4 contains an example in which traditional information graphics (Harris, 1999), such as bar charts and line graphs is used. One of the most appealing visualisations of log file data is the Wattle Tree (Kay, Masionneuve, Yacef, & Reimann,

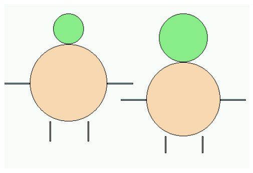


Figure 1.3: Avatars representing content of chat communication between two learners. See text for details.

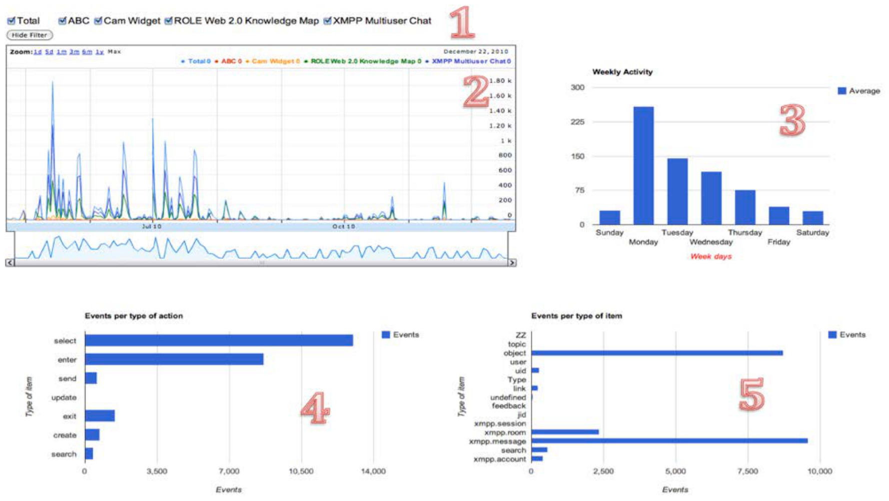


Figure 1.4: Example of a dashboard (Duval, 2011, p. 13, Figure 5).

2006a), see Figure 1.5. Here the activities of learners in a group are displayed in a time line, one for each learner running from bottom to top. Activities of the learners are visualised as yellow and orange “flowers” and green leafs. Larger flowers represent more activity, larger leafs represent that it took the learner longer to respond to a request from another learner. Kay, Masionneuve, Yacef, & Reimann (2006b, p. 7) note “It appears that group members would gain far more from all the displays than the lecturer can. In particular, each individual would have a real understanding of what their own Wattle Tree meant.” This type of visualisation can be used for reflection by learners and as an overview for teachers.

In conclusion, visualisation is a powerful technique to present information about the learning process to all stakeholders involved. This is especially true when the visualisation changes dynamically. Dashboards and the indicator of participation (Janssen, 2008) are examples of dynamic visualisations that can help learners monitor their own activity. Both static and dynamic visualisations can aid researchers and teachers to understand learner behaviour.

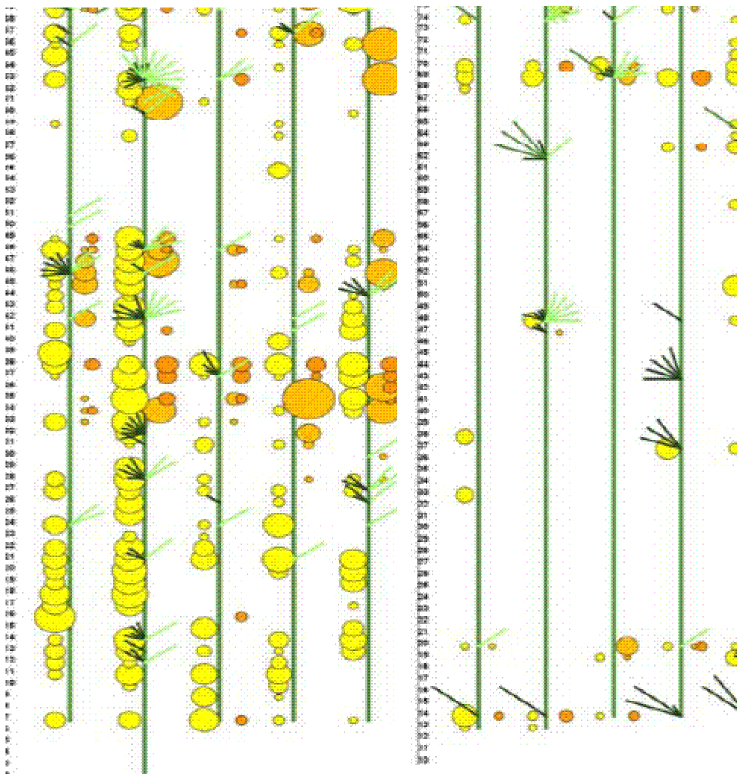


Figure 1.5: Wattle trees to represent group activity, good group (left; Kay et al. (2006b, p. 201, Figure 4)) and dysfunctional group (right).

1.3 Methods

In the following sections we describe methods to analyse learner data and, where appropriate, the relation between the methods and pedagogical knowledge.

1.3.1 Mining methods

Data mining is concerned with the automatic discovery of patterns in a set of data. Data mining is applied to large, often homogeneous, data sets to find patterns that are well-supported (frequent). Brief descriptions of commonly used data mining methods are given below.

Classification. Classification refers to the assignment of parts of the data set to predefined categories. Most classifiers are "supervised", i.e., they use an example data set from which they "learn" the parameters that determine the classification. Classification is similar to categorical coding (see Section 1.3.4).

Clustering. Clustering separates the data into subsets based on the similarity of features found in the data.

Relationship mining, association rule mining. This method is applied when an "item" has several features and associations between the features are expected. The traditional example for an "item" is the shopping basket, and the associations that can then be discovered are of the form "if A buys X and Y he is to buy Z with probability P". In education, an item can be replaced by a student, X and Y by student activity (following courses, reading learning material), and Z with succeeding on a course.

(Predictive) modelling. The objective of predictive modelling on learner data is to define or select a model that best predicts the next step or action of a student. Predictive student modelling is the dominant method in the analysis of intelligent tutoring systems.

In the first issue of the *Journal of Educational Data Mining* Baker & Yacef (2009) give an overview of the discipline, partly based on an earlier review of EDM literature published in the period 1995–2005 by Romero & Ventura (2007). Both publications provide statistics on which data mining methods are used and changes in the trends of the usage of these methods. The major shift Baker & Yacef (2009) identify is that relationship mining has declined from 43% in the early days of EDM to less than 10% based on the proceedings of the annual EDM conferences (2008–2009). Methods from psychometrics, especially model discovery, have gained in prominence from 0% in the early days to 28% recently. The explanation provided is that this increase "is likely a reflection of the integration of researchers from the psychometrics and student modelling communities into the EDM community" (Baker & Yacef, 2009, p. 8). The emphasis of EDM has seemingly shifted from "pure" data mining approaches, such as relationship mining, to student modelling approaches. The popularity of student modelling can be partly explained by the extensive use of cognitive tutors, especially in the United States. This results in large, homogeneous, publicly available data sets that can readily be analysed by modelling techniques.[4] Inquiry learning environments generate both heterogeneous data and are used on a smaller scale. This makes it more difficult to apply data mining techniques and get meaningful outcomes.

[4]See for example the KDD Cup 2010: `http://www.kdd.org/kdd2010/kddcup.shtml`

Although the relative prominence of relationship mining (e.g., using association rules) has declined in the EDM community, it is still one of the most important traditional data mining techniques used on educational data. For example, the Signals system (Campbell, 2007) depends on patterns from relationship mining.

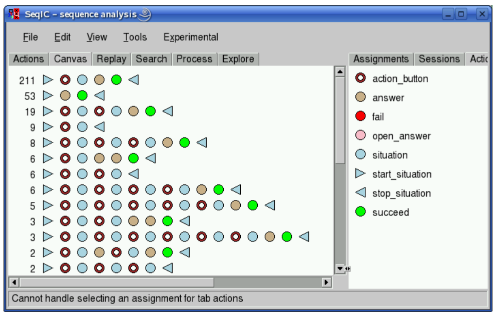
1.3.2 Frequency analysis

A standard method in the learning sciences to understand data from learning is to apply *frequency analysis*. Count the number of actions of a particular type in the data and use this count as an indicator for a certain type of behaviour. For example, in a simulation environment the number of simulations tried can be seen as a proxy for a learner's experimental or theoretical approach to learning. In a collaborative environment, the number of chats can be seen as representative for the intensity of communication with the group.

It is widely acknowledged (e.g., Rosé, Cui, Arguello, Weinberger, & Stegmann (2008); Erkens & Janssen (2008)) that frequency analysis should be used with care, as it often ignores too much relevant detail about the behaviour of a learner. For simulations it is interesting to know which values for the input variables the learner has tried. In inquiry learning trying extreme values is a good tactic, and knowing whether the learner has tried such values can be valuable input for pedagogical interventions. In a collaborative environment, the number of chats says little about the quality of the contribution, but can be used to obtain data about who talks with whom.

1.3.3 Sequence analysis

Sequence analysis takes the order in which learner actions are performed into account. Finding frequent sub-sequences and relating these sub-sequences to other information about learners is an established approach. To find interesting sequences a representation is needed that is expressive enough to capture relevant details of the learning process, but not too complex as it would reduce the likelihood of finding frequent patterns (Kay, Masionneuve, Yacef, & Zaïane, 2006). Such representations will dependent on the particular learning environment and which patterns are of interest. For example, Perera, Kay, Koprinska, Yacef, & Zaïane (2009) have used the notation (iRj) to capture a sequence in which j learners used resource R a total of i times in succession. In this abstraction the particular learners who used the resource and the order in which they did this is ignored, for example $(3R2)$ could be AAB, BBA, ABA, BAB (where A and B are learners). The abstraction (iRj) is an example of a pedagogically motivated selection of the underlying patterns from the raw data.



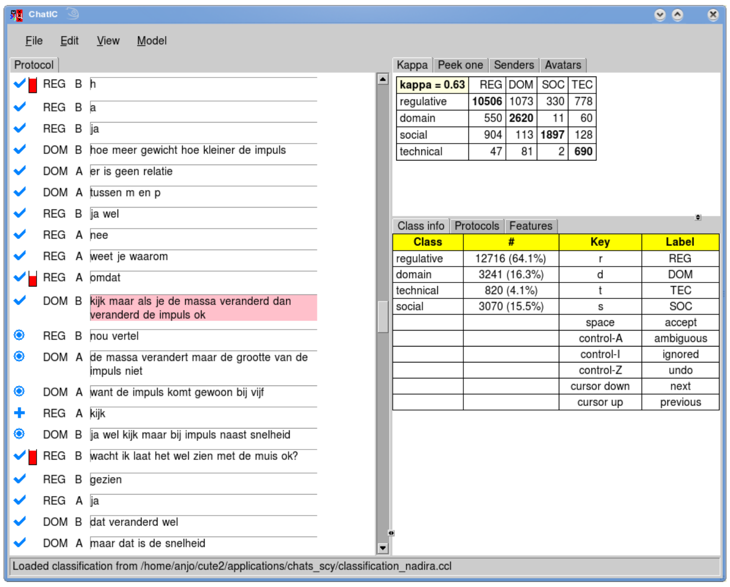
Cannot handle selecting an assignment for tab actions

Figure 1.6: Most common action sequences for an assignment. 211 learners pressed the action button to start a simulation, followed by the correct answer.

Another application of sequence analysis is to look at the order in which the learner performs actions. An illustration is provided in Figure 1.6. Right is a legend for the action types and at left sequences found in the log files from Kollöffel (2008). The most frequent sequence (211 learners) was to run a simulation (type: "action button") and next give the correct answer (type: *answer* followed by *succeed*). 53 learners did not run a simulation, but immediately gave the correct answer. This form of sequence analysis has several applications. One can develop a mathematical model of the probability that one action is followed by some other action (e.g., using Markov chains). These models can be used to predict the behaviour of future learners. A second type of application is to derive different kinds of strategies from the sequences, for example to determine whether learners run a simulation even when they know the answer.

1.3.4 Coding, abstracting and representing expert knowledge

A general method, often used in the behavioural sciences, is to assign a categorical code to learner actions. Categorical coding is necessary when the "raw data" is too heterogeneous for analysis. Applied to log files, the code is an interpretation of the action that is more abstract than the details of the action itself, but less abstract than the type of action alone. For example, all edit actions in a concept mapping tool could be coded as either improving or worsening the map. Coding schemes define the codes that are possible and give guidelines when a given code should be assigned by a human coder. A wide variety of coding schemes have been defined and applied (de Wever, Schellens, Valcke, & van Keer, 2006). After coding, frequency or sequence analysis can be applied on the codes rather than on the underlying data.



✓ 🟥	REG	B	h
✓	REG	B	a
✓	REG	B	ja
✓	DOM	B	hoe meer gewicht hoe kleiner de impuls
✓	DOM	A	er is geen relatie
✓	DOM	A	tussen m en p
✓	REG	B	ja wel
✓	REG	A	nee
✓	REG	A	weet je waarom
✓ 🟥	REG	A	omdat
✓	DOM	B	kijk maar als je de massa veranderd dan veranderd de impuls ok
◉	REG	B	nou vertel
◉	DOM	A	de massa verandert maar de grootte van de impuls niet
◉	DOM	A	want de impuls komt gewoon bij vijf
+	REG	A	kijk
◉	DOM	B	ja wel kijk maar bij impuls naast snelheid
✓ 🟥	REG	B	wacht ik laat het wel zien met de muis ok?
✓	REG	B	gezien
✓	REG	A	ja
✓	DOM	B	dat veranderd wel
✓	DOM	A	maar dat is de snelheid

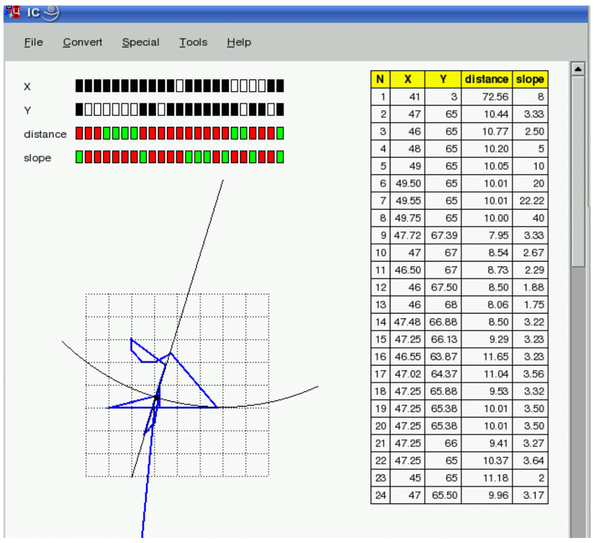
kappa = 0.63	REG	DOM	SOC	TEC
regulative	**10506**	1073	330	778
domain	550	**2620**	11	60
social	904	113	**1897**	128
technical	47	81	2	**690**

Class	#	Key	Label
regulative	12716 (64.1%)	r	REG
domain	3241 (16.3%)	d	DOM
technical	820 (4.1%)	t	TEC
social	3070 (15.5%)	s	SOC
		space	accept
		control-A	ambiguous
		control-I	ignored
		control-Z	undo
		cursor down	next
		cursor up	previous

Loaded classification from /home/anjo/cute2/applications/chats_scy/classification_nadira.ccl

Figure 1.7: ChatIC, an interactive tool researchers can use to train an algorithm to code chats from a collaborative learning environment.

For the analysis of complex structures, such as concept maps, the usual method is to define expert solutions. An agent can then compare the expert solution to the student solution. For the classification of (short) text messages, machine learning techniques are used to train the agent. Several tools that support this training and categorical coding have been developed. The most prominent is TagHelper (Rosé et al., 2008) which uses a suite of machine learning algorithms and can learn to classify chats and other short texts after human training. An alternative to TagHelper is ChatIC (Anjewierden & Gijlers, 2008). ChatIC, see Figure 1.7, has an intuitive interactive interface for training. At left are the messages prefixed with the coding (REG, ...). Each time the expert enters a code for a chat message, ChatIC updates the underlying model and then recomputes the coding for all chats. If there is a discrepancy between the code applied by the human and the algorithm a "red" indicator appears before a message. At lower left is the coding scheme, and at upper right ChatIC displays a confusion matrix of the agreement between the coder and the algorithm, including kappa. Practical results from TagHelper and ChatIC suggest that automatic chat coding, after training, results in acceptable accuracy. The avatar of Figure 1.3 is updated by automatic analysis of messages trained with ChatIC.



N	X	Y	distance	slope
1	41	3	72.56	8
2	47	65	10.44	3.33
3	46	65	10.77	2.50
4	48	65	10.20	5
5	49	65	10.05	10
6	49.50	65	10.01	20
7	49.55	65	10.01	22.22
8	49.75	65	10.00	40
9	47.72	67.39	7.95	3.33
10	47	67	8.54	2.67
11	46.50	67	8.73	2.29
12	46	67.50	8.50	1.88
13	46	68	8.06	1.75
14	47.48	66.88	8.50	3.22
15	47.25	66.13	9.29	3.23
16	46.55	63.87	11.65	3.23
17	47.02	64.37	11.04	3.56
18	47.25	65.88	9.53	3.32
19	47.25	65.38	10.01	3.50
20	47.25	65.38	10.01	3.50
21	47.25	66	9.41	3.27
22	47.25	65	10.37	3.64
23	45	65	11.18	2
24	47	65.50	9.96	3.17

Figure 1.8: Example of abstractions and replay based on raw log data (Hendrikse, 2008). See text for a description.

An example of abstraction is detecting VOTAT (vary one thing at a time). Only changing one variable and observing the effect on other variables can aid understanding the relation between the variables. Automatic data mining methods are unlikely to find patterns that correspond to VOTAT. It is therefore necessary to abstract from the raw data and recode the learner actions before VOTAT can be detected. In an assignment from Hendrikse (2008), learners had to find the x- and y-coordinates of a point at a certain distance from a start point while the line between these two points had to have a given slope. After each try, learners obtained feedback whether the distance and/or slope was correct. Figure 1.8 shows a tool to analyse the raw data from this assignment. The black arc, at lower left, represents the correct distance and the black line the correct slope, the point learners had to find is where the arc and line cross each other. Right are the (x, y) values tried by a learner, and the computed distance (correct is 10.0) and slope (correct is 3.26). The blue lines represent the "path" of the values tried, this path can be animated. At upper left are two abstractions of the data. The black and white abstraction is like VOTAT. If the x or y is unchanged compared to the previous try, the rectangle is white, otherwise it is black. Whether the distance and slope were correct is represented by the red (wrong) and green (correct) abstraction. The researcher has used this abstracted visualisation of the raw log data to obtain an understanding of the strategies learners use, which can in turn be used for adaptation.

1.4 Discussion and outline

Using the analysis of what learners do as a source for improving the learning experience has been a goal for a long time. However, examples in which the results of data mining are used in learning environments are difficult to find. Hübscher & Puntambekar write:

> "Educational data are mined with the goal to discover knowledge about the learners, educational software and other classroom interventions. Thus, the designers need to be explicit about how that knowledge is being used to redesign educational software. Yet, many of us working in the general area of educational technology too often talk about software or more general interventions at the implementation level. Staying at that level leaves the use of the data mining knowledge and its integration with pedagogical knowledge implicit." Hübscher & Puntambekar (2008, p. 97)

In other words, they suggest that EDM research should be less concerned about data mining technology and focus on addressing how the outcomes of data mining can be integrated into learning environments such that learners might benefit. Based on a survey of the full papers at EDM 2011 (Pechenizkiy et al., 2011) given at the start of this chapter there is still a focus on technology.

Learning analytics may be a more promising path. We repeat its definition: "the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs" (Siemens, 2011, online). Learning analytics has a purpose "understanding and **optimising** learning and the environments in which it occurs". The analysis of educational data can result in an understanding of what learners are doing (which is what EDM predominantly aims at), changing the behaviour of the learner (through for example visualisation and other forms of informative feedback as exemplified by learning analytics) and dynamic adaptation of the learning environment.

In this thesis we will contribute to the latter of these challenges: the dynamic adaptation of inquiry learning environments on the basis of learner behaviour. Previous sections have described related work supportive of our research. In Section 1.2 approaches to achieve adaptation are given. These approaches have a sound foundation in the learning sciences and related disciplines, and they can be used. A major challenge is to discover actionable patterns (Figure 1.1) in the kinds of data

(Section 1.1) with the available methods (Section 1.3). Past research indicates that actionable patterns do not simply emerge from the data by applying data mining techniques. The search for actionable patterns has to be more focussed and use techniques beyond data mining. The approach we follow in the thesis is to look at each of the three types of data available (process, products, communication and collaboration) and select methods specific to find actionable patterns in these types of data.

Process. The way in which students use a learning environment can traditionally be found in log files. In modern environments, see for example de Jong et al. (2010), user actions can be made available immediately for analysis. The analysis of the actions students perform can result in patterns of unsystematic behaviour (for instance violating the VOTAT rule during experimentation), not using all resources available or discovering a student is stuck. Feedback based on process analysis is often in the form of hints or through dashboard like indicators. Chapter 2 addresses process analysis and describes techniques to find pedagogically interesting sequential patterns. These patterns can be linked to feedback to students.

Products. These are the complex structures students create in learning environments. Examples of products are models, concept maps or other graph-like structures, essays, and experimental designs. The analysis of these products is often both domain specific and dependent on the type of product. Patterns normally result from a comparison between the student product and an "expert solution" or solutions that contain misconceptions students might have. Feedback can be in the form of indicators, the number of correct concepts in a concept map for example, or by pointing out specific errors. Chapter 3 describes agent-based support for the analysis of graph-like structures occurring in instructional contexts: concept maps and (system dynamics) models.

Communication and collaboration. In many modern learning environments communication and collaboration plays an important role. Students work together by exchanging short (chat) messages, by commenting on each other's work, or creating a collaborative product. The analysis of the content of messages can be used to understand what learners discuss. Chapter 4 looks at the content of chat communication between dyads of learners who collaboratively solve assignments in an inquiry learning environment.

Chapter 5 contains a summary and discussion.

Chapter 2

Detecting patterns in log files

Another Brick in the Wall (Part II)

We don't need no education
We don't need no thought control
No dark sarcasm in the classroom
Teachers leave them kids alone
Hey! Teachers! Leave them kids alone!
All in all it's just another brick in the wall
All in all you're just another brick in the wall

Roger Waters, Pink Floyd (1979)

2.1 Introduction

Adaptation of a learning environment can be based on the analysis of the process, products (Chapter 3) and communication and collaboration (Chapter 4). In this chapter we look at the analysis of the process: the activities learners perform in inquiry learning environments. The observable behaviour of learners, on which adaptation has to be based, are the actions logged by the learning environment. The challenge is to find pedagogically interesting patterns in this ordered stream of actions, and link these patterns to adaptation.

Inquiry learning environments support students in actively discovering knowledge through processes like hypothesizing, running experiments and interpreting results. An important aspect of the study of learner activity in inquiry learning is that students have a large degree of freedom. Inquiry learning encourages exploration, and

This chapter is a much expanded version of A. Anjewierden, H. Gijlers, N. Saab & R. de Hoog (2011). Brick: Mining pedagogically interesting sequential patterns. *EDM 2011: 4th International Conference on Educational Data Mining*, pp. 341–342, Eindhoven, The Netherlands.

students can themselves decide which activities to perform and in which order these activities are performed. This makes it practically impossible to define, in advance, which action sequences are "good" or "bad".

Patterns of activity, therefore, have to be derived from actual learner activity as recorded in the log files of a learning environment, and pedagogical knowledge then has to judge whether the patterns found are suitable for adaptation and what type of adaptation has to be provided. Finding patterns may be challenging: "[in inquiry learning environments] many fine-grained interface actions [...] can be done in any order, which makes looking for recurring sequences in user actions computationally expensive without much added value." (Kardan & Conati, 2011, p. 161). The authors argue that sequence analysis on the "raw" log data, is neither easy nor useful. There are too many different types of actions and the order in which students perform these actions might not reveal patterns of pedagogical interest.

Is inquiry learning sequence analysis a *contradictio in terminis*? SimQuest simulation environments may contain more than sixty types of actions. Provided an action can follow every other action, 3,600 different sequences of length two (bigrams) can occur, 216,000 sequences of length three (trigrams), and so forth. This is clearly unmanageable, and to find interesting sequences a representation is needed that is expressive enough to capture relevant details of the learning process, but not too complex as it would reduce the likelihood of finding frequent patterns (Kay et al., 2006). We illustrate how complexity might be reduced and expressiveness increased with detecting VOTAT (Chen & Klahr, 1999) (the strategy to modify one variable at a time to find the relation between the changed variable and its effects on the other variables in a simulation). If there are three variables that can be changed, a *variable changed* action can be replaced by the variable it changes (say with the codes X, Y, and Z, for the three variables), other actions are represented by lowercase characters, and running a simulation by a space. An action sequence is then coded as, for instance, "saX efYa ZaXfwY abY". Sequences can be further simplified by removing the "other" actions, resulting in "X Y ZXY Y" and now VOTAT, or lack thereof, can be detected. Shanabrook, Cooper, Woolf, & Arroyo (2010) provide another example where pre-processing and reformulation of log data makes it suitable for the analysis of the behaviour of learners. In a tutoring system they re-coded each action, taking into account the immediate context and how long it took the learner to perform an action. For example, there are different codes for giving an answer within five seconds of starting a problem, and longer than 30 seconds. Analysis of these coded sequences revealed patterns pointing to typical student behaviour that can be explained: "Too difficult" is when students take time to read the problem, but still use hints to find the answer (Shanabrook et al., 2010, p. 198).

In the remainder of this chapter we assume raw log data has been pre-processed and coded as described in the previous paragraph. The data available for analysis can thus be characterized as sequences with a limited number of different relatively high-level actions. Next the sequences need to be analysed to find interesting patterns that can be linked to feedback. An approach is to use sequence analysis methods to find "statistically interesting" patterns automatically, and then use pedagogical knowledge to decide whether the patterns found are suitable for adaptation. Sequence analysis methods of potential relevance are described in Section 2.2. We developed a tool, called *Brick*, that supports both manual and semi-automatic discovery of patterns. The user interface and basic functions of Brick are given in Section 2.3. In the following two sections we apply Brick and the sequence analysis methods to find patterns for adaptation in short sequences (Section 2.4) and longer sequences (Section 2.5). The discussion is found in Section 2.6.

2.2 Sequence analysis methods

In this section we give a brief overview of sequence analysis methods and tasks potentially relevant for the analysis of learner generated sequences. The formal definition of a (discrete) sequence is an ordered list of objects, events, observations, or, in our case, coded learner actions.

Data mining. In traditional data mining the most common sequence analysis task is frequent sequence discovery in large databases. Methods are generally based on the apriori principle (e.g., Srikant & Agrawal, 1996), a sequence of length n can only be frequent if it starts with a sequence of length $n - 1$ that is frequent. Frequent, relatively short, sequences are of interest to consider as a source on which feedback can be based.

Markov chains. Sequences can be characterized by a Markov chain if the probability distribution of the next action only depends on the current action, and not on any prior actions. Processes that can be modelled by a Markov chain are therefore called memoryless. The memoryless assumption is somewhat difficult to defend, in general, when the sequence generating process is a student using an inquiry learning environment. Markov chains have many applications, and are also used to analyse learner generated data, particularly when the number of possible actions is small. In a tutoring system there may be three possible student actions "get hint", "correct answer" and "incorrect answer". Based on these actions, Köck & Paramythis (2010) use Markov chains to describe and classify student types (e.g., trial and error, hint abuse, etc.). An attractive property of Markov chains is that for any given set of sequences the transition matrix, which contains the transition probabilities between actions, can be easily generated. This transition matrix can then be used to determine whether actual sequences differ from those predicted by the Markov chain.

Hidden Markov models (HMMs). Hidden Markov models (Rabiner, 1989) are used when the process that generates the sequences can be in several unobservable (hidden) states. Whereas in a Markov chain the next action is only determined by the current action, in an HMM the next action depends on the current state and the current action. Each state therefore has its own transition matrix (as in a Markov chain), as well as probability distributions to enter the next state. Hidden Markov models can be automatically derived from the available data. They are potentially attractive for understanding student behaviour if the states of the model can be associated with types of student behaviour.

Entropy (information theory). Entropy in information theory (Shannon & Weaver, 1949), measures the predictability of a message as it arrives from a source. Suppose the source is a student familiar with VOTAT and one of his coded action sequences is "X X X Y Y X X Z Y ". Each time we see an X, Y, or Z (a variable changed) it is immediately followed by a space (a simulation run). This sequence is somewhat predictable and has low entropy. A sequence of a non-VOTAT abiding student may be "XY YZ X XZ XYZ X ". In this sequence it is much more difficult to predict what follows after an X, and the sequence has higher entropy. Entropy can be thought of as a measure of information. When the VOTAT student has changed variable X, we know he will run a simulation next, no information is added by the simulation run. When the non-VOTAT student changes variable X, we have to wait until the next action arrives and that action thus provides new information. The idea of a student as the source of a message that conveys information is intuitively appealing. We will use entropy and relative entropy, which provides a measure for the difference between sets of sequences.

Regular expressions. Regular expressions (Kleene, 1956) are used for pattern matching. The most common application is in string matching. For example, the regular expression "A*.pdf" when applied to a list of file names, matches all PDF documents that start with an A. Regular expressions provide a powerful mechanism to specify queries about a sequence corpus.

The methods described above support the discovery of frequent sub-sequences, predicting the next action of a student (Markov chains, HMMs, conditional probabilities in general), characterisation of a sequence (entropy) and a notation to manually find sub-sequences of interest (regular expressions). Combined, these methods cover the basic approaches to sequence analysis used.

2.3 Exploring action sequences with Brick

Brick is an interactive tool to explore action sequences. One way to find interesting sequences is to let a user define a query and next visualise the matching subsequences. Another way is to apply (probabilistic) sequence analysis methods. The two approaches can complement each other, for example sequence analysis methods can be applied to the results of a query. Examples of queries are "which actions follow a simulation action", "which sequences of three consecutive actions occur", or the more complex "what happens between two simulation actions"? Queries about action sequences are very similar to matching patterns in strings (character sequences with a finite alphabet) with regular expressions (Kleene, 1956).

In Section 2.3.1 we describe a corpus of coded action sequences that will be used throughout the chapter. In Section 2.3.2 we give the regular expression notation used, and how it is applied in Brick.

2.3.1 Learning environment and coded actions

The coded action sequences used are derived from a simulation-based inquiry learning environment in which learners collaborated in dyads, using a chat channel for communication (Saab, 2005). The learning environment was created with SimQuest (van Joolingen & de Jong, 2003) and provided assignments in the physics domain of collisions. Each assignment consisted of a description of a situation in the domain, a question, a list of possible answers and an interactive simulation dyads could manipulate.

Two students worked on their own computer and used chat communication to discuss what simulations to perform, interpret results of simulations and decide on the answer to give. All actions in the learning environment and all chats were logged. The actions selected for analysis are:

ⓢ **start.** An assignment is started.

🟩 **simulation.** Running a simulation in the learning environment.

🟩 **correct.** Providing a correct answer to a multiple choice question.

🟥 **wrong.** Providing an incorrect answer to a multiple choice question.

ⓔ **end.** An assignment is closed.

The pictorial symbols in front of the actions are “bricks”. Bricks are used to build queries from as well as to visualise sequences. For instance, the sequence in which a dyad starts an assignment, runs a simulation, selects the correct answer, and then closes the assignment is ⓢ 🟩 🟩 ⓔ.

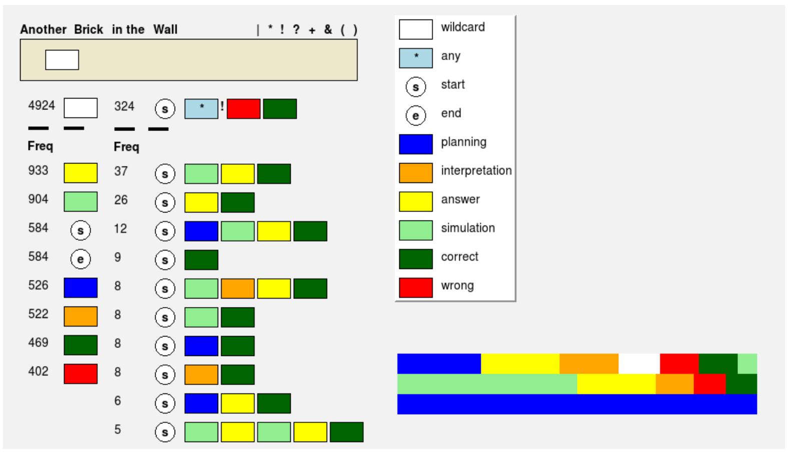
Chat communication within dyads has been manually coded with a coding scheme that captures the underlying inquiry process (Kuhn, Black, Keselman, & Kaplan, 2000):

🟦 **planning.** Domain-related chat about planning a simulation experiment: “I think we have to increase mass”.

🟧 **interpretation.** Domain-related chat about interpreting the results of an experiment: “Speed is halved”.

🟨 **answer.** Discussing what answer to give: “Answer 4 appears to be correct”.

The chat messages were assigned one of the above by two coders (inter-rater reliability $\kappa = 0.72$ for the 7,384 individual chat messages). Off-topic and short intermittent messages, such as “hmm” or “continue”, are ignored and when adjacent chats obtained the same code these were collapsed into a single chat episode. A total of 2,081 such episodes remained for analysis.



Freq		Freq	
4924	⬜	324	ⓢ * ! 🟥 🟩
933	🟨	37	ⓢ 🟩 🟨 🟩
904	🟩	26	ⓢ 🟨 🟩
584	ⓢ	12	ⓢ 🟦 🟩 🟨 🟩
584	ⓔ	9	ⓢ 🟩
526	🟦	8	ⓢ 🟩 🟧 🟨 🟩
522	🟧	8	ⓢ 🟩 🟩
469	🟩	8	ⓢ 🟦 🟩
402	🟥	8	ⓢ 🟧 🟩
		6	ⓢ 🟦 🟨 🟩
		5	ⓢ 🟩 🟨 🟩 🟨 🟩

Figure 2.1: Brick user interface. See Section 2.3.3 for a description.

2.3.2 Specifying queries with regular expressions

Regular expressions can be created using the following standard operators and meta-characters:

□ **wildcard.** Matches precisely a single action in the corpus. For example, □ □ matches a sequence of two actions.

$\boxed{*}$ **any sequence.** Matches any sequence of actions, including no actions at all. All actions within an assignment match Ⓢ $\boxed{*}$ ⓔ.

| **alternation.** Infix operator which matches the expression before or after the |. For example 🟩 | 🟥 matches either a correct or a wrong answer.

? + * quantification. Postfix operators which match zero or one (?), one or more (+), and zero or more (*, Kleene star) of the preceding regular expression.

( ) grouping. Parentheses are used for scoping and to override the default precedence of operators. They don't match anything.

In addition two non-standard constructs are provided. Negation is often required to filter unwanted matches. For example, finding all sequences in which the first answer provided by the dyad is correct, requires that there is not an intermediate wrong answer: Ⓢ ($\boxed{*}$! 🟥) 🟩 (result is shown in Figure 2.2). Here ! is the negation operator. Similarly, sequences in which the dyad does not provide any answer are matched by Ⓢ $\boxed{*}$! (🟩 | 🟥) ⓔ. The second extension are repeat loops, which occur frequently in learner action sequences. An example of such a loop is 🟩 🟧 🟩 🟧. The notation $\boxed{X}$ □ $\boxed{X}$ matches all trigrams in which the first and third action are the same. Once $\boxed{X}$ gets bound, any subsequent $\boxed{X}$ has to be bound to the same action. $\boxed{X}$ $\boxed{Y}$ ($\boxed{X}$ $\boxed{Y}$)+ matches all sequences of alternating actions. For convenience, this may be abbreviated by the & postfix operator which matches two or more: ($\boxed{X}$ $\boxed{Y}$)&.

Regular expression matching in Brick is implemented by the algorithm described in Thompson (1968). The Thompson algorithm does not include negation. A simple extension to the algorithm makes negation possible without increasing computational complexity.[1]

[1]We are not aware of an existing implementation of a regular expression algorithm that supports "full" negation.

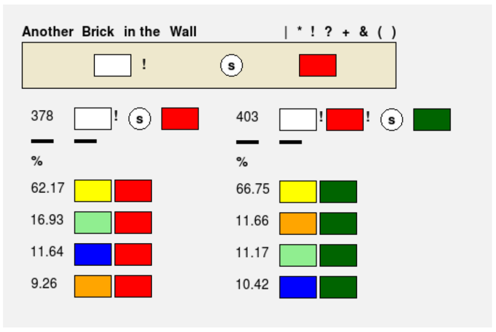


Figure 2.2: Which action precedes a wrong (left) or a correct (right) answer?

2.3.3 Finding patterns manually

Figure 2.1 shows the Brick user interface. At the right is a palette with bricks. The user creates a query by dragging bricks from the palette to the search box (top left). The basic functions are available through four buttons above the search box (Another / Brick / in the / Wall). Once a query has been completed, pressing the **Brick** button causes the sub-sequences and their frequencies to be displayed. At the very left of the figure the matches of ⬜ are shown (these are the frequencies of the actions in the corpus). The **in the** button clears the results, and the **Another** button clears the search box. At bottom right is an example of pressing the **Wall** button, with the query 🟦⬜⬜. A wall provides a visual summary of the actions based on the colours of the bricks. In the wall shown, we can see that about half of the actions that follow planning are simulation actions.

The basic user interface and the ability to use regular expressions to specify patterns is already useful. For example, in the center of Figure 2.1 the most frequent sequences leading to a correct answer are shown. Visual inspection suggests that 🟨 often immediately precedes a correct answer. Perhaps this is a pattern to base feedback on. If a dyad often gives a wrong answer and does not discuss what answer to give, then directive feedback to discuss which answer to give might be useful. Figure 2.2 shows the results for both correct and wrong answers. There is often a 🟨 before a 🟩, but this is also the case for 🟨 and 🟥. The results also suggest that a simulation is more often followed by an incorrect than a correct answer. Verifying this with the pattern 🟩 🟩 | 🟥 results in 41% for a correct answer and 59% for an incorrect answer after a simulation. This can perhaps be used for feedback.

In this section we have described how regular expressions can be used to specify patterns that are matched against a corpus of learner action sequences. Brick shows the matching sequences as well as how frequent these sequences are. Regular expressions can mimick Markov chains with □ □ which gives the transition matrix as a list. Similarly, the standard sequential data mining algorithm of finding long and frequent sequences can be emulated by adding more □ pattern elements. In the next section we look at probabilistic methods to find "interesting" short sequences automatically.

2.4 Interesting short sequences

Intuitively, it can be argued that short sequences are more interesting to consider for adaptation than longer sequences. Shorter sequences occur more frequently, they may be easier to interpret, and it may be easier to associate feedback with short sequences. In this section we apply statistical measures on short sequences. The matching sequences are ranked according to the value computed by the measures. The motivation is to identify measures that select "interesting" sequences automatically, rather than on frequency alone as in the previous section.

2.4.1 Measures

Collocation

In statistical natural language processing sequences of two or more words that correspond to some conventional way of expression are called collocations (Manning & Schütze, 1999, p. 163–166). Collocation detection methods can also be applied to sequences in general. One method based on the T-test is as follows:

$$t = \frac{\overline{x} - \mu}{s} \sqrt{N} \tag{2.1}$$

N is the total number of words in the corpus, μ is the probability a collocation occurs based on independence of the occurrence of words. For example, the probability of the occurrence of "piece of cake" is $Pr(piece)Pr(of)Pr(cake)$. $\overline{x}$ is the frequency of "piece of cake" in the corpus. If we assign 1 to each occurrence of "piece of cake" and 0 otherwise, the outcome is a Bernoulli trial. The variance for a Bernoulli trial is $p(1 - p)$, where p is the probability of "piece of cake", s is then the standard deviation. In collocation analysis, the value of t is used for ranking the results to which it is applied. When $t \gg 0$ a sequence is interesting because it happens (much) more often than chance, whereas $t \ll 0$ points to patterns that occur much less than chance.

D'Mello's likelihood metric

D'Mello and colleagues (D'Mello, Taylor, & Graesser, 2007; D'Mello, Olney, & Person, 2010) propose a so called *likelihood metric*[2] for the study of behavioural sequences. Equation 2.2 takes into account that the sequence $a_t a_{t+1}$ (two subsequent actions) also depends on the frequency of a_{t+1}:

$$Pr(a_{t+1}|a_t) = \frac{Pr(a_{t+1} \cap a_t)}{Pr(a_t)} \tag{2.2}$$

$$L(a_t \rightarrow a_{t+1}) = \frac{Pr(a_{t+1}|a_t) - Pr(a_{t+1})}{1 - Pr(a_{t+1})} \tag{2.3}$$

The likelihood metric in Equation 2.3 is similar in structure to Cohen's kappa (Cohen, 1960). The range is between $-\infty$ and 1.0. A value around 0 (zero) can be interpreted as at chance level, a positive value is above chance (and therefore potentially interesting) and a negative value is below chance (and perhaps also interesting).

Equation 2.3 only applies to bigrams. For longer sequences D'Mello et al. (2010) propose Equation 2.4.

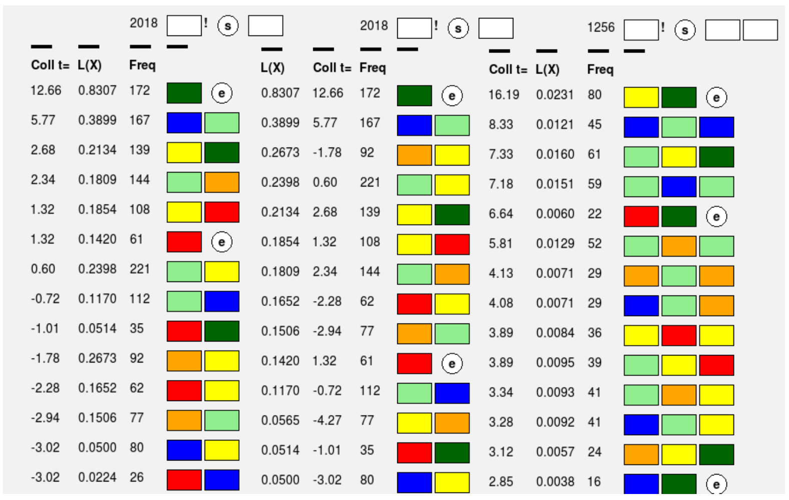
$$L(X) = \frac{Pr[X_O] - Pr[X_S]}{1 - Pr[X_S]} \tag{2.4}$$

Here, $Pr[X_O]$ is the observed probability of seeing sequence X and $Pr[X_S]$ is the probability of X occurring randomly. The interpretation of $L(X)$ is the same as for the likelihood metric above.

2.4.2 Examples

The measures described above have been integrated in the results display of Brick. Figure 2.3 provides an illustration for sequences with two or three actions, ignoring the start action. Both measures assign a high value to action sequences that occur together more often than might have been predicted by the actions occurring independently. The highest ranked sequences, the bigram 🟩 ⓔ and the trigram 🟨

[2]The name is somewhat unfortunate as it is a measure, not a metric.



		2018	▢! ⓢ ▢
Coll t=	L(X)	Freq	
12.66	0.8307	172	🟩 ⓔ
5.77	0.3899	167	🟦 🟢
2.68	0.2134	139	🟨 🟩
2.34	0.1809	144	🟢 🟧
1.32	0.1854	108	🟨 🟥
1.32	0.1420	61	🟥 ⓔ
0.60	0.2398	221	🟢 🟨
-0.72	0.1170	112	🟢 🟦
-1.01	0.0514	35	🟥 🟩
-1.78	0.2673	92	🟧 🟨
-2.28	0.1652	62	🟥 🟨
-2.94	0.1506	77	🟧 🟢
-3.02	0.0500	80	🟦 🟨
-3.02	0.0224	26	🟥 🟦

		2018	▢! ⓢ ▢
L(X)	Coll t=	Freq	
0.8307	12.66	172	🟩 ⓔ
0.3899	5.77	167	🟦 🟢
0.2673	-1.78	92	🟧 🟨
0.2398	0.60	221	🟢 🟨
0.2134	2.68	139	🟨 🟩
0.1854	1.32	108	🟨 🟥
0.1809	2.34	144	🟢 🟧
0.1652	-2.28	62	🟥 🟨
0.1506	-2.94	77	🟧 🟢
0.1420	1.32	61	🟥 ⓔ
0.1170	-0.72	112	🟢 🟦
0.0565	-4.27	77	🟨 🟧
0.0514	-1.01	35	🟥 🟩
0.0500	-3.02	80	🟦 🟨

		1256	▢! ⓢ ▢ ▢
Coll t=	L(X)	Freq	
16.19	0.0231	80	🟨 🟩 ⓔ
8.33	0.0121	45	🟦 🟢 🟦
7.33	0.0160	61	🟢 🟨 🟩
7.18	0.0151	59	🟢 🟦 🟢
6.64	0.0060	22	🟥 🟩 ⓔ
5.81	0.0129	52	🟢 🟧 🟢
4.13	0.0071	29	🟧 🟢 🟧
4.08	0.0071	29	🟦 🟢 🟧
3.89	0.0084	36	🟨 🟥 🟨
3.89	0.0095	39	🟢 🟨 🟥
3.34	0.0093	41	🟢 🟧 🟨
3.28	0.0092	41	🟦 🟢 🟨
3.12	0.0057	24	🟧 🟨 🟩
2.85	0.0038	16	🟦 🟩 ⓔ

Figure 2.3: Top-ranking sequences for the likelihood L(X) and collocation (Coll t=) measures. Left are bigrams ranked by decreasing collocation, center are bigrams by decreasing L(X), and right are trigrams by decreasing collocation.

🟩 ⓔ are logical given the learning environment, and the same applies to most of the other higher ranked sequences. The high-scoring sequences are interesting because they occur more often than based on chance and thus help explain how dyads order their actions in the learning environment. The low scoring sequences (for bigrams these are 🟢 ⓔ and 🟩 🟢) may also be interesting as they point to deviating behaviour. The high-ranking sequences can be interpreted as resulting from non-random behaviour, the dyads purposely performed the actions in particular order. A study of the pedagogical desirability of high-ranking sequences can result in indicator of how "systematic" dyads are. This systematic indicator can then be used for informative feedback.

2.5 Analysis of longer sequences

The study of longer sequences is of interest when we want to know whether the behaviour of learners changes over time. One approach to study longer sequences is to derive a model from the data and generate sequences based on the model. If the model provides a good fit of the real data, it can be used to interpret or predict what learners will do. Another reason to study longer sequences is comparing corpora from different groups of learners. We investigate modelling approaches and comparison of groups of learners in the next sections.

2.5.1 Hidden Markov models

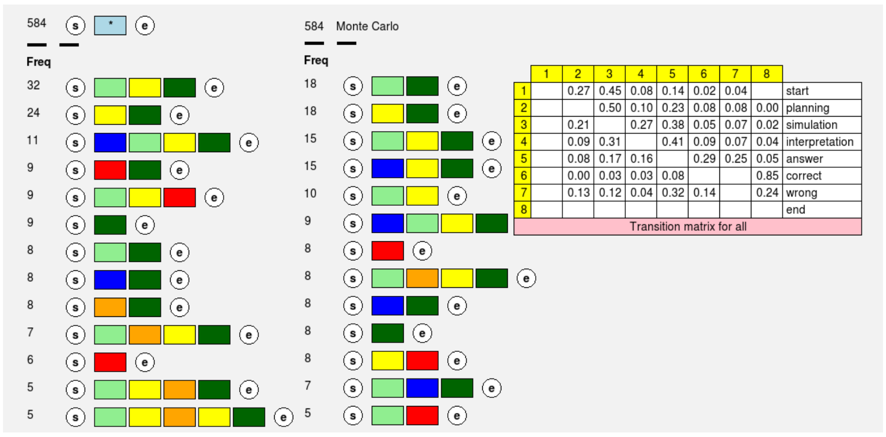
Hidden Markov models (HMMs) are a useful approach when we expect there to be several hidden states that influence which action will follow next. For our data set we used the UMDHMM package (Kanungo, 1999) to generate a HMM with between 2 and 25 states. In the HMMs generated, 85% or more of the actions were assigned to a single state and we found it impossible to meaningfully interpret the generated models. D'Mello et al. (2010) report similar difficulties applying HMMs to behavioural data similar to ours. We do not consider HMMs further.

2.5.2 Markov chains

A commonly used method to model discrete sequences is a Markov chain. For a Markov chain the next action only depends on the current action with a fixed probability distribution. For example, in our corpus, the probability of a simulation following a planning action is 0.50. A Markov chain derived from the corpus thus predicts that a simulation action will follow a planning action half of the time. A Markov chain is completely described by the initial probability distribution (in our corpus this 1.0 for Ⓢ and 0.0 for the other actions), and the transition matrix. The transition matrix for the corpus is given on the right of Figure 2.4. The rows represent the current action and the columns the next action. For example, cell R1, C2[3] contains the probability of going from start to planning, it is 0.27. The 0.50 probability of going from planning to simulation is found in cell R2, C3.

At the very left of the figure are the most frequent complete sequences in the corpus. In the center are the complete sequences predicted by the Markov chain based on a Monte Carlo simulation with the given transition matrix. For relatively short, and frequent, complete sequences the corpus and the Monte Carlo simulation produce somewhat similar results. For longer sequences, visual inspection suggests a deviation between the content of the corpus and the sequences randomly generated based on the Markov chain. In the next section we use entropy to find an explanation for the apparent differences.

[3]The cell of row 1 and column 2.



	1	2	3	4	5	6	7	8	
1		0.27	0.45	0.08	0.14	0.02	0.04		start
2			0.50	0.10	0.23	0.08	0.08	0.00	planning
3		0.21		0.27	0.38	0.05	0.07	0.02	simulation
4		0.09	0.31		0.41	0.09	0.07	0.04	interpretation
5		0.08	0.17	0.16		0.29	0.25	0.05	answer
6		0.00	0.03	0.03	0.08			0.85	correct
7		0.13	0.12	0.04	0.32	0.14		0.24	wrong
8									end

Transition matrix for all

Figure 2.4: Left are the most frequent complete sequences in the corpus. In the center the most frequent complete sequences generated by a Monte Carlo simulation based on the transition matrix (right) are shown.

2.5.3 Entropy

Entropy in information theory (Shannon & Weaver, 1949), is a measure of uncertainty. In the previous sections we have given examples of which actions follow each other. Entropy provides a measure for *how well* we can predict the next action, given actions just seen. The idea of entropy can be illustrated by comparing the rows for correct and wrong in the transition matrix of Figure 2.4. The probability of 🟩 ⓔ is 0.85 (cell R6, C8), and the probabilities of the other actions after a correct answer are low. In other words, we would be surprised if a dyad does not follow 🟩 by ⓔ. The next action after a wrong answer is much more difficult to predict, the highest probability is 0.32 (cell R7, C5) for 🟥 🟨 and the probabilities for other actions are close to each other.

Entropy provides a measure for the predictability of the next action, given the seen action(s). Entropy, denoted by H, is given in Equation 2.5.

$$H(p) = -\sum_{x \in X} p(x) log_2 p(x) \tag{2.5}$$

Here, p is a probability distribution. The probability distribution for the actions that follow a correct answer can be found in row 6 of the transition matrix. Applying the probabilities in row 6 to Equation 2.5 results in 0.84. For a wrong answer, row 7, the entropy is much higher: 2.37. Entropy is measured in bits, on average 0.84 bits are required to encode the action that follows a correct answer.

In the corpus there are seven possible actions (ignoring the initial start action), and if these actions would occur randomly, it takes $log_2(7) = 2.807$ bits per action to represent an action sequence. As the transition matrix illustrates, the dyads do not generate random action sequences. We can compute the entropy for the entire corpus as follows. Suppose, we have seen the previous action, how well can we predict the next action? This can be computed by taking the weighted average of the entropy of the individual actions. Knowing the previous action, the number of bits to encode the next action, the entropy, is 2.022. This value is also the entropy of the transition matrix, as the transition matrix gives the probability distribution for the next action only based on the previous action (and no actions before that).

In the previous section visual inspection of the sequences generated randomly based on the transition matrix suggested that these sequences differed from the actual sequences for the corpus. We are now ready to verify this based on entropy. The entropy of knowing the previous action was 2.022, if we know the previous *two* actions the entropy becomes 1.804. In other words, knowing more about actions that occurred earlier lowers the entropy and therefore increases predictability. If we know the previous *three* actions, the entropy lowers further to 1.673.

The lower entropy for longer known sequences provides an explanation why the Markov chain does not accurately predict the actual sequences. The dyads are not memoryless, their future behaviour is influenced by actions in the past. One hypothesis is that there are tipping points that cause the behaviour of the dyads to change. Potential tipping points are the actions themselves, in particular planning, simulation, interpretation, answer or wrong actions. The occurrence of giving a correct answer is, as described above, also a tipping point, it causes the dyad to close the assignment. In the next section relative entropy, or Kullback-Leibler divergence, is used to determine whether there are tipping points.

2.5.4 Kullback-Leibler divergence

Kullback-Leibler divergence (Kullback & Leibler, 1951), or relative entropy, provides a measure for the difference between two probability distributions. Kullback-Leibler is given by Equation 2.6.

$$D(p\|q) = \sum_{x \in X} p(x) log_2 \frac{p(x)}{q(x)} \tag{2.6}$$

Here, $p(x)$ and $q(x)$ are the two probability distributions and, by convention $0\,log\frac{0}{q} = 0$, and otherwise $p\,log\frac{p}{0} = \infty$. The Kullback-Leibler divergence is 0.0 when the two probability distributions are identical, otherwise the measure gives the average number of extra bits necessary to encode p when the distribution is based on q.

We can now determine whether there are tipping points as follows. For sub-sequences of increasing length, the probability distribution is computed for the entire corpus. This is the probability distribution for q. For each potential tipping point, the probability distribution of the sub-sequences that occur after the tipping point is computed. This is p. Table 2.1 lists the results for sub-sequences of length one through three.

$D(p\|\|q)$		*Sub-sequence length*		
p	q	1	2	3
planning	corpus	0.002	0.019	0.055
simulation	corpus	0.016	0.027	0.050
interpretation	corpus	0.035	0.068	0.119
answer	corpus	0.089	0.131	0.157
wrong	corpus	0.130	0.223	0.368

Table 2.1: Kullback-Leibler divergence after one of the given actions compared to the entire corpus.

The results suggest that the actions after a planning or simulation action, and to a lesser extent after an interpretation action, are more or less the same as before, based on the low values. For the actions after discussing an answer, and especially after a wrong answer, the values are much higher. For example, three actions after a planning action the divergence is 0.055 whereas the divergence one action after discussing an answer is higher (0.089) and after a wrong answer much higher (0.130). The tipping points seem to occur when the dyads start discussing what answer to give and giving a wrong answer. Apparently, dyads are in two modes: using the learning environment to determine what is going on; or, deciding on the answer. Once a dyad starts discussing a possible answer, and certainly after giving a wrong answer, the likelihood of returning to inquiry mode is small.

The change of behaviour after starting to discuss what answer to give can be seen in the walls of Figure 2.5. Both walls result from the same query: 🟨⬜⬜⬜. Walls are read from the bottom upwards to identify the sequences they represent. The top wall shows the sequences precisely as extracted from the corpus, for example the sequence at the very left of the top wall is 🟨🟩🟨🟩, and there are several of

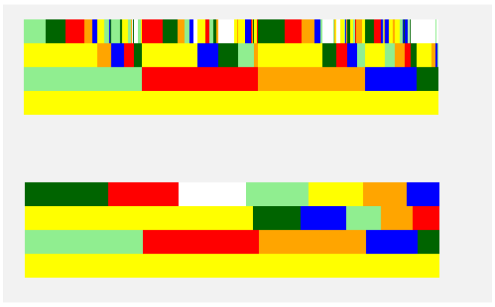


Figure 2.5: Walls for the next three actions after discussing the answer. See text for an explanation.

these sequences. In the bottom wall, the distribution of the actions is re-ordered for each row and this makes it possible to see the distribution of actions over time. For example, in the bottom visualisation it can be seen that 🟨⬜🟨 matches about 50% of the sequences. An explanation of the behaviour visible in the wall may be that when dyads start discussing what answer to give, and then either give an answer immediately, or decide to perform another simulation to make sure, followed by again discussing what answer to give.

If the "entry into answer mode" pattern is detected repeatedly for a dyad, it might be appropriate to give directive feedback suggesting to run or plan another simulation to obtain more information from the learning environment, rather than "guess" the answer.

2.6 Discussion

In this chapter we looked at the analysis of process data resulting from learners using inquiry learning environments. Process data can be represented as coded sequences of the actions learners perform. Following the steps in Figure 1.1 (page 16), we used analysis methods to find (statistically) "interesting" patterns in these sequences, and then applied pedagogical knowledge to select patterns to base adaptation on. We discuss whether these steps can also be applied to other corpora, and how patterns found can be used by pedagogical agents for online adaptation (the monitoring step in Figure 1.1).

The sequence analysis methods implemented in the Brick tool and applied to the corpus appear to be valuable, they can be used to find the type of patterns that are interesting to base feedback on. A caveat, if we want to apply the patterns found in the corpus is, that some actions are based on manually coded chat messages. The question whether these chats can be automatically coded by a software agent has a simple answer: this is not possible with current language technology. For example, determining whether a chat should be coded as planning or interpretation is too difficult (κ is around 0.3, see also Anjewierden & Gijlers (2008)). The implication is that the results obtained in this chapter cannot be used in a software agent in a straightforward manner.

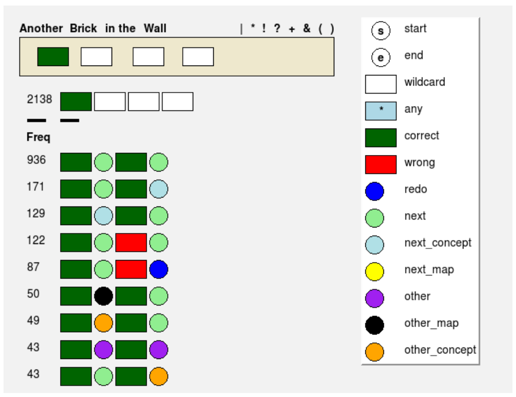


Figure 2.6: Brick on a corpus in which log data has been coded automatically.

As has been argued in the introduction, supported by Kardan & Conati (2011) and Kay et al. (2006), re-coding of the raw log data is almost always necessary for meaningful analysis. Chats are difficult to code automatically, especially if a distinction has to be made between planning and interpretation. However, in many cases the analysis can be based on a corpus that is derived from the log files in a procedural way. If the actions of interest are relatively high-level, these can be selected and all low-level interface actions can be ignored. This is the case for the data used in Section 3.4. Here actions represent modelling activities, for example adding or deleting elements and relations, and actions that represent running a model or a simulation. For this corpus we are particularly interested in detecting “floundering” with the sequence analysis methods described. Another approach to obtain a relatively small number of different types of actions is illustrated by Figure 2.6 based on Hagemans, van der Meij, & de Jong (2012). The log data in this case consisted of a series of assignments, the main actions being 🟩 and 🟥 for answering these assignments correctly or wrongly, respectively. In the learning environment, learners could decide on the ordering in which they performed the assignments. Whether they followed a "prescribed" order, or deviated from this order is coded by the circle bricks. The particular type of coding was decided on by the researcher. The coding itself is deterministic based on the log files. Patterns that emerge from analysis of this data can be implemented in a pedagogical agent and be used for adaptation.

A question that might be asked is whether the "corpus" of an individual learner is large enough to base adaptation on. A distinction has to be made between the analysis of a corpus to determine which patterns are interesting using sequence analysis, and applying the patterns found for an individual learner. To find interesting patterns, a substantial amount of data is required. Once the patterns have been found and implemented in an agent, this is no longer the case. If the pattern matches, adaptation can be applied. In the spirit of the exploratory nature of inquiry learning some constraint may be necessary. For example, to wait with adaptation until the pattern matches multiple times. The details of when and how a matching pattern should result in adaptation is a pedagogical and not a technological issue.

Chapter 3

Agent-based support for the analysis of graph-like structures created by students

3.1 Introduction

Graph-like structures, such as concept maps and models, play an important role in many instructional contexts. In this chapter we address the automatic analysis of graphs created by students. Traditionally, the (manual) analysis of graphs has mainly been used for evaluation and assessment after the graphs are completed. Our primary objective is to make it possible to create learning environments in which the graphs students create are analysed online. Such analysis can be used for the adaptation of the learning environment and for informative feedback based on an evaluation of the graphs. There are two main challenges to make automatic online analysis possible. The first, somewhat technical, challenge is the automatic analysis itself. The second challenge is to find an approach that makes it possible for researchers to specify how the analysis is performed, and when it should result in adaptation or informative feedback.

Graph-like structures are graphical depictions that can be formally represented as mathematical graphs with nodes and edges. The most common graphical representation used in education is a concept map (Novak & Gowin, 1984). Other graph-like representations created by students, are models. A model also consists of nodes connected by edges, the types of nodes and what the edges represent depends on the particular modelling notation. An example is the system dynamics notation for modelling dynamical processes (Forrester, 1961). Concept maps have only informal semantics, they are static and not executable. In system dynamics, relations have semantics and students can manipulate the formulas associated with the variables (nodes) and observe changes over time by running the model. For both concept mapping and system dynamics modelling there are equivalent non-graphical representations. The knowledge contained in a concept map can be represented as text, and models can be represented as a set of differential equations. Part of the attractiveness of the graph-like representation for instruction is the visualisation of the relatedness between concepts or variables.

In concept mapping students draw nodes that represent concepts and connect these concept nodes by (labelled) lines representing the relation between the connected concepts. The primary motivation for concept mapping in an instructional context is making students' knowledge about a certain (science) domain explicit. Maps students draw provide an insight into their cognitive structure which can be used for assessment (Acton, Johnson, & Goldsmith, 1994). Computer-based tools that provide support for concept mapping are widely available. Surprisingly, none of these tools provide support for the analysis of the graphs created (Marshall, Chen, & Madhusudan, 2006). This suggests that the analysis has to be done manually, which is tedious (Kinchin & Hay, 2000). Ideas about how to analyse concept maps have existed for a long time. Novak & Gowin (1984) introduced concept mapping for educational purposes and also proposed a scoring method which is still widely used. Approximately ten years later extensive review studies about whether and how to assess concept maps were published (Shavelson, Lang, & Lewin, 1994; Ruiz-Primo, Shavelson, & Schultz, 1997). However, the first papers about the automatic analysis of graph-like structures we have been able to find date from 2006 (Bravo, van Joolingen, & de Jong, 2006; Marshall et al., 2006).

The main reason to provide automatic analysis of graph-like structures is related to the general idea of adaptation in learning environments. Students generally find system dynamics modelling difficult (van Borkulo, 2009). If the models can be analysed while they are created, several scenario's that can benefit students become possible. One is to provide students with feedback about the quality of the graph created, similar to what a teacher would do. Another type of scenario centers around detecting how a student progresses. For example, in a complex domain a concept map might be broken down into several partial maps. Based on analysis it could be suggested to the student that one of the partial maps is more or less complete. In system dynamics there is a distinction between qualitative and quantitative models. In qualitative models students only have to specify the type of relation between variables, whereas in quantitative models the relation has to be specified exactly with a formula. Qualitative models are easier for students and a scenario for adaptation is to use information about the quality of a qualitative model to control when students can progress to the more difficult quantitative modelling. Finally, in collaborative learning environments, the automatic analysis of graphs can be used to suggest peers to collaborate with, based on similarity or dissimilarity of the graphs.

The remainder of this chapter is organised as follows. In Section 3.2 we review related work, considering manual scoring and evaluation of graph-like structures, as well as automatic analysis support. In Section 3.3, based on the outcome of the review, we present a generic approach to support automatic analysis. With this generic approach researchers can specify both the domain of learning and how analysis and evaluation of graphs takes place. Section 3.4 provides an example of the application of the approach in system dynamics modelling. This section also describes how software agents that implement the analysis communicate with a learning environment. Section 3.5 illustrates the use of the approach based on several different concept mapping tasks. Finally, in Section 3.6 we reflect on what has been achieved and give an outline of further application of the approach.

3.2 Related work

Concept maps are thought of as informal representations of a students' knowledge and many scoring, evaluation and assessment approaches and scoring systems were proposed (Shavelson et al., 1994). Ruiz-Primo & Shavelson (1996) identify three approaches to scoring concept maps: 1) scoring the structural content of a map; 2) comparing a student's map with an expert's map; and 3) a combination of these two approaches. The scoring system developed by Novak & Gowin (1984) is an example of evaluating the structural content of a concept map. Their scoring system awards points for meaningful propositions, the number of hierarchical levels, cross links between concepts in different branches of the hierarchy, and examples (instances).

3.2.1 Computer-based analysis of graph-like structures

In this section we look at computer-based implementations for the analysis of graph-like structures. For the following three systems graph analysis was the prime motivation. Marshall's tool (Marshall et al., 2006) supports open-ended concept mapping, and the main objective is developing an automatic scoring algorithm that can replace human scoring. SDM Technology (Ifenthaler, 2010; Ifenthaler, Masduki, & Seel, 2011) supports the analysis of the development of concept maps over time. The focus is on computing graph-theoretic measures of student maps and study how these measures change over time. The third system, Cline's concept mapping tool (Cline, Brewster, & Fell, 2010) implements a scoring method and students can request the score for their map as well as an explanation of what (propositions) contributed to the score.

In two other implementations, graph analysis is part of a broader instructional context. Bravo's advice system (Bravo et al., 2006; Bravo, van Joolingen, & de Jong, 2009) aims at helping students to create a system dynamics model with Co-Lab (van Joolingen et al., 2005). Students can ask for advice on their model and the advice system next compares the student model to a reference model and gives advice based on the differences between the two models. In Betty's Brain (Biswas, Schwartz, Leelawong, & Vye, 2005; Leelawong & Biswas, 2008) students teach Betty (a software agent) about ecology-related domains by drawing a concept map. For links, students can include a causal qualification, for example "animals *produce (increase)* carbon dioxide". Betty interprets these causal links with a qualitative reasoning engine and, given the example, can conclude that when the number of animals increases, so does the amount of carbon dioxide. Graph analysis is provided by a second agent, called Mr. Davis. This agent generates quizzes for Betty, "What happens to carbon dioxide when the number of animals decreases?", and scores the answers Betty gives by using an expert causal map of the domain.

The Pathfinder system (Schvaneveldt, 1990) is sometimes associated with concept map analysis. In Pathfinder students are given a list of concepts and they specify for each pair of concepts how strongly these are related. Pathfinder then constructs a graphical representation, called a knowledge or concept map, from the ratings given by students. Although the final outcome of Pathfinder is a concept map, this map is not drawn or edited by students but by an algorithm based on the ratings. We therefore do not consider Pathfinder as an implementation of automated concept map analysis.

The descriptions of how graph analysis is implemented by the five systems are organised around three sub-tasks. *Terminology normalisation* refers to the analysis of the labels of nodes and edges entered by students. It is not normally explicitly mentioned when humans evaluate graph-like structures, for computer-based analysis it is of course mandatory when the labels are not given in advance. *Structural analysis* refers to scoring the structural content without taking the domain into account. The third sub-task is *use of reference models* to compare the student graph with an expert reference graph. The latter two tasks are the principal approaches of concept map evaluation identified by Ruiz-Primo & Shavelson (1996).

3.2.1.1 Terminology normalisation

The labels of the nodes and edges in a graph can, depending on the instructional context, be given in advance or have to be specified by students. An example of the former is a concept mapping task in which students are given a list of concept and link labels and have to create the correct structure. In open-ended concept mapping tasks, for example creating a concept map from scratch, terminology normalisation is a prerequisite for the sub-task of comparing a student graph to a reference model.

Terminology normalisation has to consider lexical- and syntactic variations and semantic similarity. The two most common methods to deal with lexical variations are stemming (Porter, 1980) and edit distance (Damerau, 1964; Levenshtein, 1966). Stemming is generally inappropriate for the analysis of node and edge labels in graph-like structures as it can result in the same stem for different concepts (usually nouns) and links (usually verbs). For example, "computers" and "compute" both have the stem "comput". In computer science, correcting for misspellings and other lexical variations is an example of an approximate string matching task with applications in many areas (e.g., DNA sequences). Edit distance is the method most commonly used for approximate string matching (Navarro, 2001). Edit distance algorithms compute the shortest number of edit operations to transform one string into another. In most natural languages, a low edit distance indicates small lexical differences, for instance plural and singular forms. Misspellings also tend to have a low value for edit distance. The systems of both Bravo et al. (2006) and Marshall et al. (2006) use edit distance. Bravo's system takes small syntactic variations into account by allowing that words may appear in arbitrary order. For example, when the normalised concept is "power source", "source of power" is also accepted. Semantic similarity involves taking into account that students may use alternative terms for a concept. Bravo's systems defines for each variable in the model a list of terms students can use. This list typically includes synonyms and abbreviations. An accuracy of 74% for the matching algorithm, which includes compensating for word order and using edit distance, is reported in one of the experiments (Bravo et al., 2006). If no match is found the student is prompted to manually link the variable to one of the variable names proposed by the system.

An accuracy of 74% may seem relatively high but a single concept can be part of several propositions and if the concept label is not recognised, neither are the propositions involved. A holistic approach to terminology normalisation is to also consider the propositions a concept is part of. Suppose a student has created the concept map shown in the Figure 3.1. In this example, three of the four concepts match the expert map (75% accuracy), but none of the propositions match. Marshall et al. (2006) use a *similarity flooding* algorithm (Melnik, Garcia-Molina, & Rahm, 2002) to establish whether, in the above example, "pet" and "animal" are terminological variations. The similarity flooding algorithm first assigns a similarity value between all student and expert concepts based on edit distance and iteratively takes into account the propositions. The similarity value between "pet" and "animal" increases given

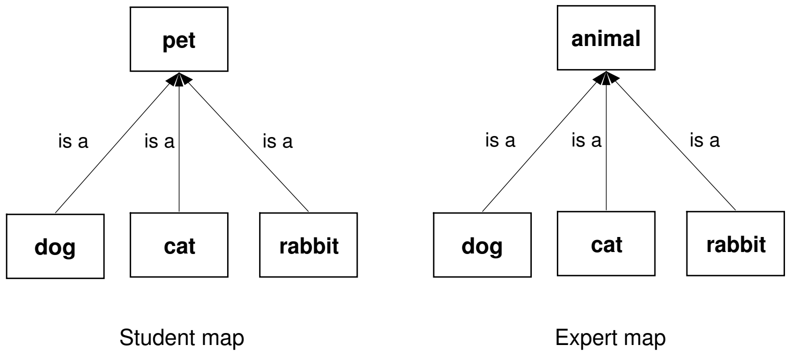


Figure 3.1: Example of structural similarity.

that in each of the three propositions "pet" can be replaced by "animal" to obtain a perfect match with the expert map. In an experiment in which computer science students summarized course material as concept maps, similarity flooding produces better matching of nodes between student and expert maps than edit distance alone: 94% for similarity flooding and 88% for edit distance (Marshall et al., 2006).

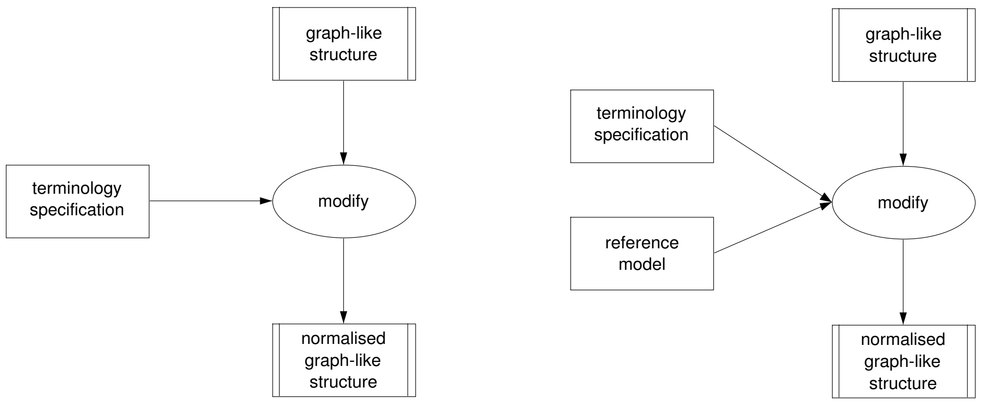

graph-like structure
terminology specification
modify
normalised graph-like structure
graph-like structure
terminology specification
reference model
modify
normalised graph-like structure

Figure 3.2: Inference structures for terminology normalisation. Left uses only a specification of terminology, and right also uses structure of the graph as given in a reference model.

The similarity flooding algorithm has some drawbacks. Marshall et al. mention that the algorithm sometimes corrects misconceptions, which is undesirable in an instructional context. The complexity of the algorithm is $O(N_s^2 \times N_r^2)$ (N_s and N_r are the number of nodes in the student and reference graphs) for each iteration (Melnik, Garcia-Molina, & Rahm, 2001). If the graph is large and the algorithm does not converge quickly, feedback in a few seconds is not guaranteed. The largest drawback is that the algorithm does not allow lexical variations to be specified. Section 3.4.5 gives data about the frequency of lexical variations, and illustrates that synonyms are often necessary to achieve acceptable results.

Figure 3.2 summarizes the approaches using the inference structure notation (Schreiber, Akkermans, Anjewierden, de Hoog, Shadbolt, Vandevelde, & Wielinga, 1999). In an inference structure ellipses represent the type of inferencing to be performed. Rectangles represent input and output roles. Input roles make explicit the kind of data required to be able to perform the inferencing. The inference structure on the left reflects Bravo's system as it modifies the student graph on the basis of a specification of terms alone. The inference structure on the right reflects Marshall's similarity flooding approach in which both a terminology specification and the structure of the graph in the reference model are input roles to normalise the labels. The output is, in both cases, a modified graph-like structured in which the labels have been normalised.

3.2.1.2 Structural analysis

Graph-like structures can be analysed based on their structural properties alone. Ifenthaler and colleagues have developed a software tool called *SMD Technology* (surface, matching, deep structure) which converts concept maps drawn on a computer into the underlying graph representation with nodes, edges, and propositions (Ifenthaler, 2010; Ifenthaler et al., 2011). Graph-theoretic measures are computed to characterise the structure of the concept maps. Examples of measures SMD computes are node count, surface structure (number of propositions in the graph), and matching structure (the minimally longest path between any two nodes). Ifenthaler et al. (2011) provide an operationalisation of the measures in terms of cognitive structures. The surface structure is, for example, operationalised as "The complexity of a cognitive structure indicates how broad the understanding of the underlying subject matter is" (Ifenthaler et al., 2011, Table 1).

Graph-theoretic measures do not consider the labels or semantics in the graph, just like linguistic measures such as word count disregard the semantics of an essay. These measures can therefore not be used to determine whether a concept map reflects a correct understanding of the domain being mapped. It is therefore also difficult to see how instructional feedback can be based on these measures alone. The measures can, however, provide insight into the structural complexity of a map. Hoppe, Engler, & Weinbrenner (2012) found a negative correlation between

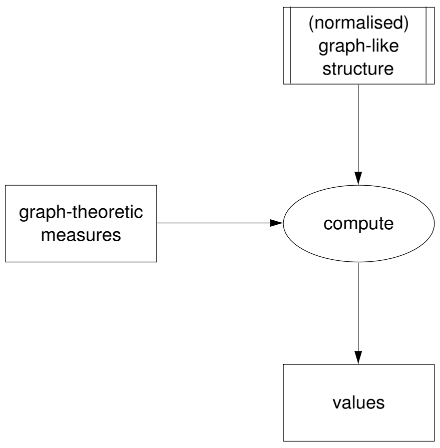


Figure 3.3: Inference structure for structural analysis.

the graph-theoretic measure of density[1] and the quality of concept maps scored manually. Ifenthaler et al. (2011) use graph-theoretic measures to study how the understanding of a topic develops over time. The inference structure for structural analysis is shown in Figure 3.3. Because structural analysis does not consider the labels, the input role can be either the student graph or a normalised graph.

3.2.1.3 Use of reference models

A content-oriented approach for evaluating graph-like structures is to determine whether the nodes and propositions are meaningful in the domain. For example, the scoring method of Novak & Gowin (1984) gives one point for each correct proposition in a student map. Judging whether a proposition is valid requires sufficient domain knowledge by the evaluator, and this is difficult to achieve in a computer program. Rather, to support the automatic assessment of graphical representations a reference representation created by experts is used (Ifenthaler et al., 2011; Marshall et al., 2006; Cline et al., 2010; Biswas et al., 2005). A slightly different approach is taken in the advice system for system dynamics modelling (Bravo et al., 2006), where the reference model accommodates modelling choices using several of partial reference models.

The inference structure in Figure 3.4 is an abstracted view of the use of reference models in implemented systems, in particular the description of Bravo's system and

[1]Defined as $D = \frac{2E}{N(N-1)}$, where E and N are the number of edges and nodes respectively.

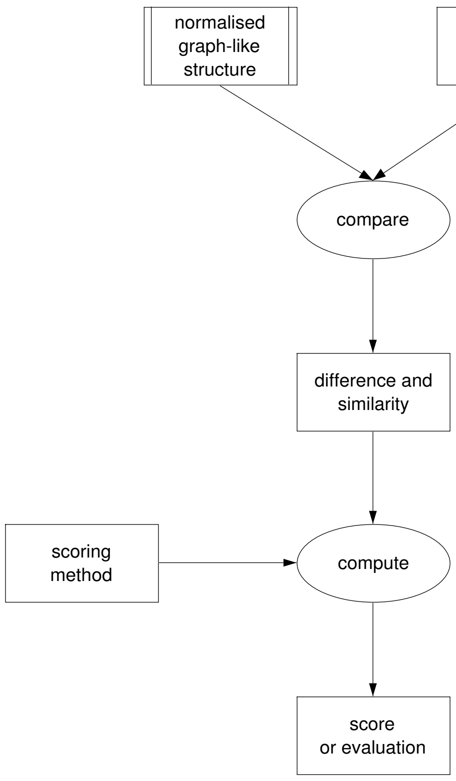


Figure 3.4: Inference structure for comparison with a reference model and scoring.

the detailed technical architecture of Betty's Brain (Leelawong, 2005). The *compare* step takes as input a normalised graph-like structure and a reference model. The output is the difference and similarity between the student and reference graph. The *compute* step uses the difference and a scoring method to produce a numerical score or some other evaluation that reflects the similarity between the student and reference graphs.

In structural analysis, domain-independent graph-theoretic measures, are computed. Results of structural analysis can therefore only be used for informative feedback, for example related to the connectedness or density of a graph. A reference model usually reflects an expert view of the domain. It is of course also possible to define a reference model that reflects a typical misconception of the domain. Use of reference models can result in all types of adaptive feedback: implicit, directive, and informative (see Section 1.2).

3.2.2 Summary

Three sub-tasks related to the evaluation of graph-like structures have been identified in the implementations of automatic analysis described above: terminology normalisation, structural analysis, and use of reference models.

Table 3.1: Overview of sub-tasks for the implemented systems

	Betty's Brain	Bravo	Cline	Marshall	Ifenthaler
Graph type	causal map	system dynamics	concept map	concept map	concept map
Terminology normalisation	no	yes	no	yes	no
Structural analysis	no	no	no	no	yes
Reference model use	yes	yes	yes	yes	yes

Table 3.1 provides an overview of how the systems address the evaluation sub-tasks. Terminology normalisation is performed by two systems, Cline et al. (2010) provides the concepts and links, whereas Ifenthaler et al. (2011) use manual normalisation. In the papers related to Betty's Brain (e.g., Biswas et al. (2005); Leelawong & Biswas (2008)) it is stated that students *enter* the name of a concept in a dialogue box. The papers do not mention whether terminology normalisation, for instance to compensate for typing errors, takes place. The thesis by one of the designers of Betty's Brain, however, states:

> ConceptMapComparator [a software module] compares two concept maps to find what concepts and links they shared by having **exact labels** and being synonyms and what each map owns exclusively (Leelawong, 2005, p. 269).

The example of a synonym given is that students can enter "bug" rather than "macrovertebrates". The reason that it is perhaps not necessary to perform terminology normalisation in Betty's Brain, is that students are encouraged to read the support material (online descriptions of the domain) before working on the concept map. In these online descriptions names of concepts are emphasized and students may cut-and-paste the names into the concept entry dialogue box. In a more recent version of Betty's Brain[2] the concept entry dialogue has been replaced by a list from which

[2]Version 3.01 (January 2012), see `www.teachableagents.org`

students select concept labels. In general, whether or not terminology normalisation is necessary depends on the task given to students. For instance, in some concept mapping tasks the emphasis is on defining the structure for a given set of concepts (Betty's Brain is an example). In open-ended concept mapping students draw a concept map from scratch, and terminology normalisation is necessary before the map can be evaluated.

Structural analysis is only extensively supported by Ifenthaler et al. (2011). This is understandable as the main focus of their work is to study the development of concept maps and the underlying cognitive structures over time. In all implementations described, a clear distinction is made between the graph notation in general and a specific domain. For the systems that implement terminology normalisation, the domain terminology can be specified by researchers. And in all systems the structure of the domain can be given as a reference model created by experts. In other words, all systems can be configured for the domain at hand without programming.

The systems can also be characterized on how evaluation takes place and what the outcome of the evaluation is. In manual evaluation a scoring method is often defined (e.g., Novak & Gowin (1984); Ruiz-Primo & Shavelson (1996)). Evaluating a student graph results in a (numerical) value which reflects applying the scoring system to the student map. Ifenthaler et al. (2011) compute a variety of single value measures, which can be viewed as "mini" scoring methods. They compute, for instance, the number of concepts or the link-concept ratio to characterise concept maps created by students. Another use of evaluation is giving feedback to students while they are creating their graphs. Cline et al. (2010) have integrated concept map evaluation in a concept mapping tool and provide two types of feedback. First, they compute an aggregated single value by comparing the student map to an expert map using a scoring method proposed in Ruiz-Primo et al. (1997). Second, they provide students with more detailed information on how the aggregated value is computed by listing the propositions in the student map and the contribution of these propositions to the aggregated value. Bravo et al. (2006), in the context of system dynamics modelling, also compare the student model to an expert model. Their feedback is in the form of advice messages that are initially very general ("you are missing a constant") and become more specific later ("you don't have a temperature variable in your model"). In Betty's Brain two different methods to provide feedback are used. First, Betty interprets the causal maps and the outcomes are scored by Mr. Davis based on the reference model. In this case the student obtains indirect feedback: whether the causal relations they have learned to Betty are correct by comparison to the reference model. Second, Mr. Davis also uses comparison of student and reference maps to give hints to students to improve their map.

The implemented systems can be easily adapted to a new domain, but it is much more difficult to customize them for a different evaluation method or type of feedback. Our primary objective is to make it possible to create learning environments in which students obtain online adaptive feedback about the graphs they create. Betty's Brain, Bravo's advice system and Cline's concept mapping tool achieve this objective to a great extent. These systems, however, implement both a very specific type of evaluation and specific types of feedback, which limits application to a few instructional scenario's.

3.3 Generic approach

In this section we describe a generic approach that gives researchers more control over how graphs are evaluated and how and when feedback is given to students. Section 3.3.1 describes the approach, and gives examples of how researchers can configure agents for different scenarios. Section 3.3.2 addresses the implementation.

3.3.1 Description

In this section we describe how evaluation of graphs can be used for online adaptive feedback. In Section 3.2.1 inference structures represent abstract descriptions of the evaluation sub-tasks and the input and output roles for these sub-tasks. Figure 3.5 contains an outline of the generic approach in which the (partial) inference structures are integrated.

Three input roles allow researchers to specify both the domain and how the evaluation takes place. The domain is specified similarly to the implementations described in Section 3.2.1. Domain terminology is represented in a terminology specification called a *term set*, and the structure of the domain is given as a *reference model*. A *rule set* defines how to evaluate a graph. As the name suggests, a rule set may contain multiple rules, and for each of these rules it is implicit whether it is a rule for structural analysis or a rule for comparison to a reference model. For example, one rule might count the number nodes in a graph (structural rule), whereas another rule counts the number of nodes present in both the student graph and the reference model (comparison rule). Below we describe the three specifications and give examples of how they can be defined by researchers.

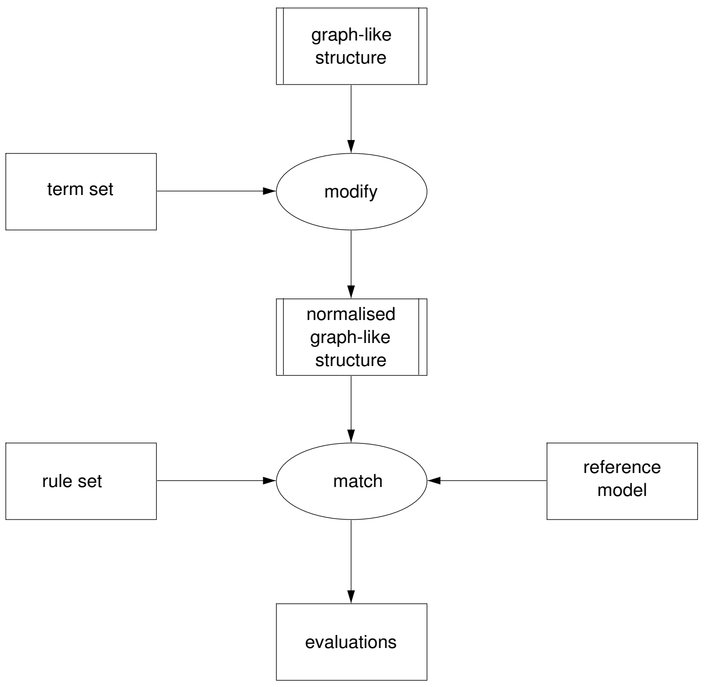


Figure 3.5: Inference structure with input/output roles for a generic approach to agent-based evaluation of graph-like structures.

3.3.1.1 Term sets for terminology specification

In evaluation by humans there is some scope to decide on an ad-hoc basis whether a label used by a student matches a particular node in a reference concept map or model. For real-time evaluation by a software agent all permissable labels must be specified in advance. Given that subsequent analysis of a graph-like structure depends on the interpretation of these labels, this is a crucial step in the overall process.

In order to reduce ambiguity the following terminology is used. A *label* is the character sequence a student uses. A *term* is the unique identifying character sequence associated with a node or edge. Terms are determined by the researcher. A *string* is a character sequence associated with a term. In the ideal case a student uses a label that is identical to a string and the label then represents the term corresponding to the string. For example, in a concept map the term "water" identifies the concept of water. The researcher can allow the chemical name "H2O" as a label students may also use.

The specification of terms for a particular domain is given in a *term set*. In a term set there is an entry for each domain term as defined by the researcher. For example, the definition for water might be:

```
<term name="water">                % Term
  <string>water</string>           % Base string
  <string>H2O</string>             % Synonym string
</term>
```

Instances when students use the same label as one of the strings are not necessarily very common. Based on an analysis of the labels used by students in three concept mapping and modelling tasks, the following categories of discrepancies occur most frequently: misspellings, inflections, synonyms, word order and descriptive terms. These categories are addressed in turn.

Misspellings. The necessity to correct for misspellings has been widely acknowledged for natural language utterances in learning environments (e.g., Nesbit & Nakayama (1990)).

Inflections. Inflected forms, often plurals, also occur frequently. For example, "cat is an animal" versus "cats are animals".

Synonyms, abbreviations. Synonyms and abbreviations are so common in scientific domains, that it is more or less necessary to provide a mechanism to be able to specify them explicitly. Whether a string is a synonym or not also depends on the learning task, thereby excluding general purpose semantic networks (e.g., WordNet or a domain ontology) to derive similar strings automatically. For example, in a concept mapping task about energy "power source" and "battery" are related and should probably be treated as separate concepts. Whereas, in a modelling task about electrical circuits, a "battery" might play the role of the "power source" for the circuit and both would be valid labels for the model element.

Word order and compounding. Multi-word labels also occur frequently in science domains. For example the already mentioned "power source" and "source of power", or similarly "CO2 reduction" and "reduction of CO2". In some languages, Dutch and German for example, such strings are compounded. The Dutch word "condensatorspanning" (capacitor voltage) is a compound of the nouns "condensator" and "spanning". Despite the compound, the words can still appear separated as in "spanning van de condensator" (voltage of the capacitor).

Descriptions. Students sometimes use informal or descriptive qualifications in the labels. For example "very many factories" or "factories in cities". Interpreting such labels requires linguistic knowledge. In a term set, it is possible to specify that only a sub-string of the label should be considered:

```
<term name="factory">                           % Term
  <sub_string>factory</sub_string>              % Sub-string
</term>
```

Table 3.4 in Section 3.4.5 lists how often the above types of terminology normalisation were found in the data of the system dynamics studies of Section 3.4.

3.3.1.2 Reference model for domain specification

A reference model represents an expert solution for a domain. The specification of a reference model follows the graph formalism of nodes and edges. The following example is the reference model for a concept map containing "cat is a animal".

```
<node term="cat"/>
<node term="animal"/>

<link term="is a"/>

<edge tail="cat" head="animal" link="is a"/>
```

In concept maps nodes refer to concepts, links to relation labels and edges to propositions. A reference model for the circumference of a circle in system dynamics is specified in a similar manner:

```
<node term="pi" type="constant"/>
<node term="diameter" type="auxiliary"/>
<node term="circumference" type="auxiliary"/>

<edge tail="pi" head="circumference" qualitative="linear"/>
<edge tail="diameter" head="circumference" qualitative="linear"/>
```

In system dynamics modelling elements (nodes) have a type, a constant or auxiliary variable in the example above (see also Section 3.4.2 for more information about system dynamics). Relations between nodes in a model are not labelled, but they can be given a qualitative relation type.

The two examples illustrate the basic definition of a reference model for concept maps and system dynamics models respectively. Both notations share the graph-like structure although there are differences in the details of the specification.

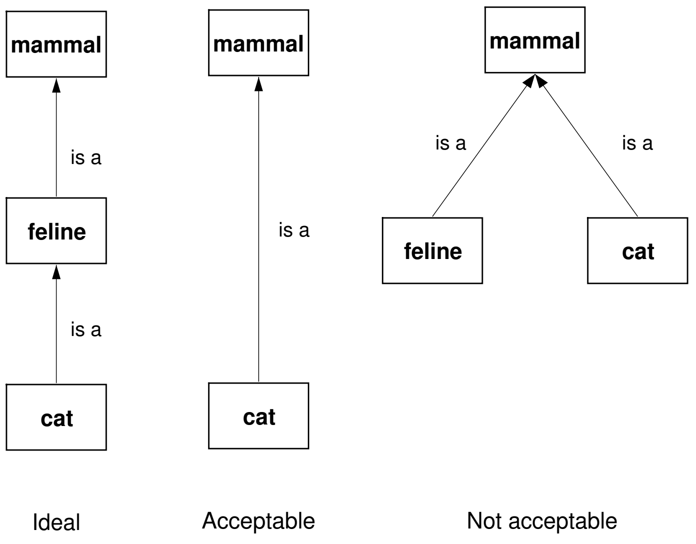


Figure 3.6: Illustration of transparent edges, see text for details.

Bravo et al. (2006) have identified the need to accommodate for modelling choices and cater for this by using a family of partial reference models rather than a single model. An extension to their approach is illustrated in Figure 3.6. The concept map on the left is the ideal solution. The map in the centre is an acceptable solution under the assumption that the "is a" relation is transitive: the student has forgotten to include "feline". The map on the right is not an acceptable solution despite containing two correct propositions, because the student missed the "is a" relation between "cat" and "feline". In a reference model this is represented with a *transparent edge* as follows:

```
<t_edge tail="cat" head="mammal" link="is a"
        excluded="feline"/>
```

This accepts "cat is a mammal" only if "feline" is not in the concept map at all. Transparent edges turn out to be a practical approach to allow for modelling choices. Further examples are given in Section 3.4.4.2.

3.3.1.3 Rule sets for evaluation specification

A rule set allows a researcher to specify how graphs are evaluated. Rules can answer structural analysis queries about the content of the student graph. Rules can also compare the content of the student and reference graphs. Three types of measures can result from the application of a rule: numerical indicators, lists or enumerations, or a logical value (true, false). For example, an indicator for the number of nodes could be 5, the list "cat, animal" could enumerate all present nodes, and a logical value could determine whether it is true that "animal" is connected to at least three nodes. A single rule consists of one or more patterns. Patterns either directly refer to the content of the graphs or contain logical constructs. In this section examples are given to illustrate the notation. Section 3.4 contains additional examples for the electrical circuits domain. For the examples below, we start with the feedback to be provided to the student and next describe the corresponding patterns and rules.

Does the graph contain a specific node? The following pattern matches when the student graph contains a node called "cat":

```
  <node graph="student" term="cat"/>
```

The equivalent for this pattern is the query: "which nodes with term `cat` are in the `student` graph". When a pattern refers to the student graph, rather than the `reference` graph, the graph attribute can be omitted. The pattern can be part of a simple rule that gives the student feedback on how many "cat" concepts are in his concept map:

```
<rule name="number of cats" result_type="numerical">
  <node term="cat"/>
</rule>
```

The rule is called "number of cats" and the numerical type means that the frequency, presumably zero or one, is returned.

Which nodes are in both the student and the reference graph? This is a query that requires a comparison of the student and reference graph. The pattern that matches all nodes that are in both graphs is:

```
  <node status="present"/>
```

For making comparisons each node or edge is assigned a status. This status is one of `present` (node or edge is in both student and reference graph), `missing` (in reference, not in student), `redundant` (in student, not in reference), and the special case `anonymous` (node is in the graph, but has not been given a label), Researchers can use these notions directly to specify a pattern to base feedback on. For example, the list of present nodes can be returned as follows:

```
<rule name="list of present nodes" result_type="list">
  <node status="present"/>
</rule>
```

Are there at least five animals in the student graph? Such a query may be used to prompt the student to think of more animals when the number is less than five. Assuming the propositions for animals are of the form "X is a animal" the pattern is:

```
  <edge head="animal" link="is a"/>
```

The next step is to count and compare the number of times this pattern matches:

```
<rule name="enough animals" result_type="logical">
  <count at_least="5">
    <edge head="animal" link="is a"/>
  </count>
</rule>
```

3.3.2 Implementation

The specifications described in the previous section represent the input roles of the inference structure in Figure 3.5. In this section we briefly describe how terminology normalisation and rule matching are implemented.

3.3.2.1 Terminology normalisation

We use the Damerau-Levenshtein algorithm (Damerau, 1964; Levenshtein, 1966) which allows for the edit operations: insertion, deletion, replacement and transposition. For instance, the edit distance between "cat" and "chat" is one (insert a "h"). The similarity between a student label (l) and a string in the term set (s) not only depends on the edit distance but also on the length of the two character sequences. The similarity measure is computed as follows:

$$sim(l, s) = \frac{max(|l|, |s|) - ed(l, s)}{max(|l|, |s|)} \tag{3.1}$$

Here $|l|$ is the number of characters in l and ed the edit distance. This results in a similarity value between 0.0 (no similarity) and 1.0 (the character sequences are identical).

Compounds are also compared using edit distance. All word permutations are generated for a string used in the term set and the spaces are removed. For example, the string "power source" results in either "powersource" or "sourcepower" as possible matches. In the label, all spaces are also removed and edit distance is applied. For example, the label "source of power" becomes "sourceofpower" and the similarity with the best match "sourcepower" is $(13 - 2)/13 = 0.84$.

The algorithm first computes a list of the similarity between all labels and all strings above a certain threshold. If the terminology of the domain is expected to include many descriptive terms, the threshold should be set relatively low (slightly above 0.5), and if the terminology consists mainly of single or two word labels a value around 0.75 is appropriate. In system dynamics modelling some students have the habit to "park" variables on the screen without connecting them to other variables. These singleton variables are ignored by the modelling engine, and therefore do not affect the output of running the model. For terminology normalisation that can affect the results when the graph includes two similar variables, for example a variable labelled "battery1" connected to other variables in the graph, as well as a "battery2" variable that is a singleton. The similarity measure computes the same value for these two variables, although the connected variable is to be preferred. Therefore, the list of potential matches is sorted on descending similarity whereby singleton nodes are placed after connected nodes. Finally, the matching pairs of labels and strings are selected starting at the beginning of the list and multiple assignments to a single term are ignored.

3.3.2.2 Rule matching

The agent is written in SWI-Prolog (Wielemaker, 2003). The nodes and edges of both student graphs and the reference model are represented in the Prolog database by clauses of the form `gls_node(Graph, Node)` and `gls_edge(Graph, Edge)`, where *Graph* is a unique identifier for a graph or reference model and *Node* and *Edge* are identifiers for the nodes and edges respectively.[3] Attributes of nodes and edges

[3]All Prolog predicates related to the graph-like structures are prefixed with `gls`.

are represented by additional clauses, for example the graph "cat is a animal" results in the following facts in the Prolog database:

```
gls_graph(g1).                    % g1 is a graph

gls_node(g1, n1).                 % n1 is a node in g1
gls_node_term(g1, n1, cat).       % n1 has term "cat"
gls_node(g1, n2).                 % n2 is a node in g1
gls_node_term(g1, n2, animal).    % b2 has term "animal"

gls_edge(g1, e1).                 % e1 is an edge in g1
gls_edge_term(g1, e1, 'is a').    % e1 has term "is a"
gls_edge_tail(g1, e1, n1).        % e1 has tail n1 ("cat")
gls_edge_head(g1, e1, n2).        % e1 has head n2 ("animal")
```

Although this representation is more verbose and less elegant than the standard representation of propositions in Prolog: `'is a'(cat, animal)`, it makes it straightforward to match the patterns of a rule. For example, determining whether a node is present is implemented as follows:

```
gls_node_status(present, Graph, Ref, Node) :-
    gls_node_term(Graph, Node, Term),  % a Node with Term in Graph
    gls_node_term(Ref, _, Term)).      % a node with Term in Ref
```

The inputs *Graph* and *Ref* are the identifiers of the student and reference graph respectively. The predicate succeeds for each *Node* that is present in *Graph*.

A rule returns the matches of the patterns it contains. If a rule has the result type numerical, it counts the matches. The following example shows a rule and the corresponding Prolog code that is executed to run it.

```
<rule name="number of present nodes" result_type="numerical">
  <node status="present"/>
</rule>

number_of_present_nodes(Graph, Ref, Result) :-
    findall(Node, gls_node_status(present,Graph,Ref,Node), Nodes),
    length(Nodes, Result).
```

The last line in the program counts the number of elements in the list *Nodes*. If the result type is an enumeration this line can be omitted and the list of matching nodes is returned directly.

3.3.3 Discussion

In this section we have presented a generic approach for the analysis of graph-like structures that can be used to provide online feedback to students. The approach is based on three specifications researchers define to customize a software agent for the domain and the kind of feedback. Domain independence is achieved by a *term set*, which contains an explicit specification of the domain terminology, and a *reference model* which specifies the structure of an expert solution. In a *rule set* researchers define the type of analysis the agent has to perform and what type of feedback is given to students.

Based on previous work, see Section 3.2.1, researchers already had control over the domain specification. With rule sets researchers also get control over most aspects of the analysis of graphs and the types of feedback. The approach described above has been used by five researchers, and one additional researcher extensively used term sets to automatically classify an existing set of system dynamics models. The overall experience is that term sets are easy to define, the guidelines in Appendix A.1 generally suffice. An initial version of a reference model can be created by drawing the corresponding graph using a concept mapping or modelling tool, the implementation generates the XML automatically. Special cases, such as transparent edges, are added manually.

The specification of a term set or reference model is essentially a simple list of factual statements. In rule sets, it is specified how the evaluation takes place, and this is potentially procedural. The current notation is motivated by a desire to keep it as simple and intuitive as possible. For example, `<node status="present"/>` is easy to understand for researchers (compare the procedural implementation in Section sec:gls-approach-implementation). There is one potentially important class of patterns that cannot be naturally defined with the current notation. This class involves patterns in which the outcome of one pattern is the input of another. An example is a pattern for "all nodes that are animals and that are present". This pattern could result in feedback like "there are five more animals in your concept map that are not connected to the animal concept". In many cases feedback centers around nodes and edges that are present or missing. The approach in the current implementation is to use additional attributes in a pattern. For the above example:

```
<edge head="animals" link="is a"
      status="missing" tail_status="present"/>
```

Here the `status` attribute refers to a missing edge and the `tail_status` attribute to a present node. A more principled solution requires the introduction of programming constructs, variables and assignment in particular, and given that rule sets are intended for use by pedagogical researchers, the relatively simple notation is preferred.

Section 3.4 describes an application of the approach in a system dynamics environment. This section includes a description of how the modelling environment communicates a request to the analysis agent and how the agent submits the feedback to the modelling tool. Section 3.5 gives examples of applying the approach to several concept mapping tasks.

3.4 Evaluation of system dynamics models

System dynamics modelling is the computer modelling of dynamic phenomena, in which students can acquire an understanding of the domain at hand by building, using, and testing computer models (de Jong & van Joolingen, 2007; Löhner, 2005; Sabelli, 2005). In this section we provide an example of how agent-based evaluation of graph-like structures, in this case system dynamics models, is applied in a "real world" learning context.

3.4.1 Pedagogical motivation

The design of the studies is motivated by an earlier study, without agent support, in the same domain (Mulder, Lazonder, & de Jong, 2009). It was found that low-level novices exhibit expert-like behaviour during the modelling task, but produce very poor models. They conclude:

> "It might be fruitful to restrain domain novices' natural tendency to engage in quantitative modelling from scratch by first having them create models that are qualitatively specified, and then enabling them to transfer these qualitative relations into quantitative ones. This type of support is in line with the model-progression approach. We propose to support students on an inquiry learning task with model progression, where the model is progressed in specificity. In line with the coding of the model

> hypotheses, three increasingly specific stage of modelling can be identified: a stage in which relationships between elements are unspecified, a stage in which relationships between elements are specified qualitatively, and a stage in which these relationships are specified quantitatively." (Mulder et al., 2009, p. 2050–2051)

Of the three modelling phases proposed, students are supported in the latter two: in the quantitative and qualitative phases students can obtain feedback from the modelling environment itself by running the model and examining the results in a graph or table. Students do not get feedback in the initial "sketching" phase. In this phase, the structure of the model is represented, but the relations are unspecified and therefore the model is not executable. In order to provide students with feedback in the sketching phase too, a model evaluation agent was developed. The feedback this agent gives is intended to help students improve model content and structure before they move to the next phase.

In the first and second study with agent-based support, model progression was decided on by students themselves and only feedback about model quality was provided by an agent. For the third study a model progression agent was included. For this agent the researcher specified rules that evaluated whether the student model was sufficiently developed to progress to the next phase: from sketching to qualitative modelling, and from qualitative modelling to quantitative modelling respectively.

3.4.2 Modelling environment

The modelling environment used in the studies is based on Co-Lab (van Joolingen et al., 2005). The model editor tool (right of Figure 3.7) enables students to build and test an executable model that represents the domain, in this case a charging capacitor. On the structure level, the syntax of a system dynamics model makes use of "stocks" (rectangle), "auxiliaries" (circles), "constants" (diamonds), "flows" (clouds) and "relations" (connections between elements). A model consists of several components: basic elements (elements that represent the model input: constants and stocks), auxiliary elements (elements that specify the integration of elements) and connecting arrows. Connected to a stock are flows, indicating changes in the stock. These changes are specified by the constant and auxiliary elements. On the content level, the model is specified in the elements: the basic elements are assigned a (initial) value and in auxiliary elements mathematical formulas express the relation between the connected variables. Modifications to a model can be tested by

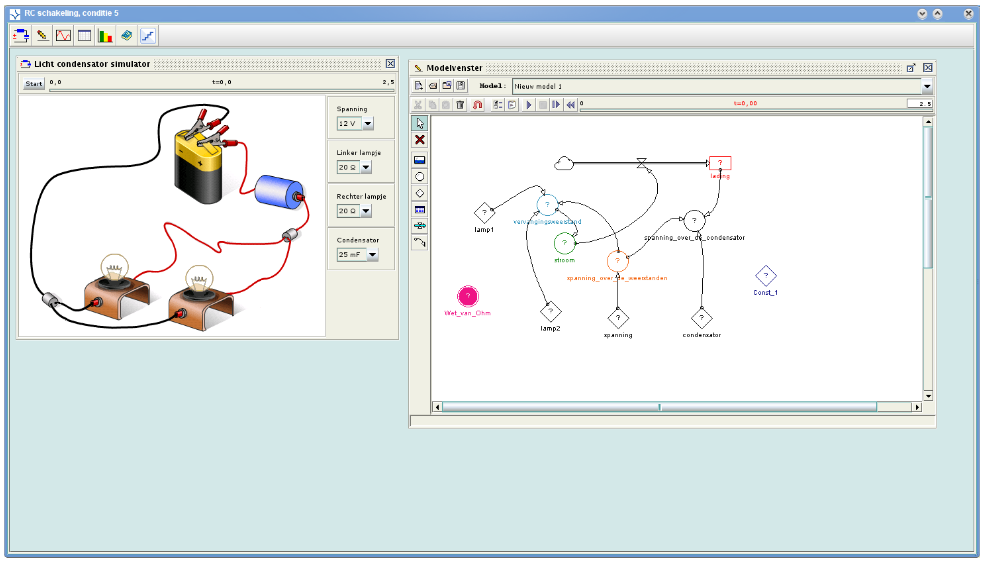


Figure 3.7: Modelling environment with a simulation (left) and a modelling tool (right).

running the model and analysing the results in the table and graph tool. These tools further allow students to compare model and simulation output in a single window.

3.4.3 Architecture

The software architecture consists of the Co-Lab system dynamics modelling tool, agents capable of evaluating models created by students and a communication server based on a blackboard system. Figure 3.8 provides an overview of the architecture.

The system dynamics modelling tool has been written in Java. The model tool was extended with logging and notification facilities, which directly connect to the communication server. This architecture allows for having multiple instances of a modelling tool, up to 60 students simultaneously in one of the studies, which are handled by a single communication server and a single agent. The logging module logs every user action by sending it in the form of an "action tuple" to the communication server. The notification module registers itself for messages from the agent and uses these messages togive direct feedback to the student (showing a bar chart that displays the quality of a model) or by adapting the modelling tool (progressing to the next modelling phase and making new modelling features available to the student).

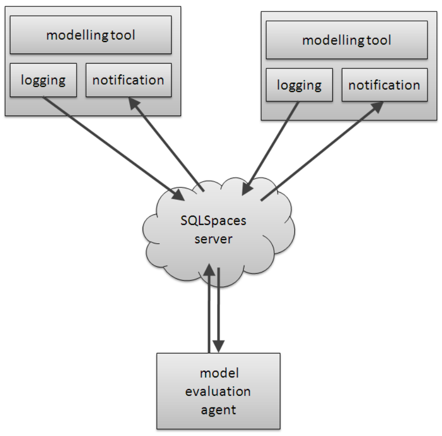


Figure 3.8: Software architecture used in the studies.

The communication server has been realised by an SQLSpaces server (Weinbrenner, Giemza, & Hoppe, 2007) as an implementation of the TupleSpaces approach (Gelernter, 1985). This blackboard architecture is well suited for agent-agent and agent-tool communication as it imposes few constraints and prerequisites on the communicating components. Furthermore, SQLSpaces clients are available in many programming languages. The Prolog client interface to SQLSpaces used by the modelling agents is described in (Weinbrenner & Anjewierden, 2011).

3.4.4 Agents

Two agents were developed. The *model evaluation agent* is a substitute for executing modelling in the first modelling phase. In this phase students sketch their model, but cannot execute it and therefore do not obtain feedback. This agent compares the student and reference model. Feedback consists of a count of how many elements in the model are correct, wrong or unspecified, how many relations are correct or wrong, and whether the direction of these relations is correct. The evaluation is provided on the student's request and displayed as a bar chart inside the modelling tool. The *model progression agent* decides whether the student can progress to the next modelling phase. The decision itself is based on comparing the student's model with a reference model using rules defined by the researcher.

The agents have been realised using the approach described in Section 3.3 and Figure 3.5. A *term set* was specified to capture the domain of a charging capacitor.

A *reference model* represents the expert solution and *rule set*'s which determine the feedback to students on evaluation. The model evaluation agent and the model progression agent use the same term set and reference model. The rule set for the model evaluation agent computes how many elements and relations are correct in the student model. The rules for the model progression agent assess whether the student model is sufficiently developed to progress to the next modelling phase.

3.4.4.1 Specification of domain terminology

In the modelling tool, students can themselves enter labels for the elements. Although some labels are obvious, either from the instructional material or the content of the simulation, there is still a large variation in the labels students use. For example, there are two lights (which act like resistors) in the simulation, some students call them "lampje 1" (light 1) and "lampje 2", others use "linker lampje" (left light) and "rechter lampje" (right light). A term set which captures the terminology was constructed based on trials and the content of the simulation and the help documents (see Appendix A.2.1).

To determine how the agent performs on the term normalisation task, a human rated all final models of the students in a pilot study ($n = 62$). For each node in the reference model it was determined whether it was present or missing. The confusion matrix is given in Table 3.2. Cohen's κ is 0.94, precision is 0.97 and these values were considered good enough to use the model evaluation agent in the first study.

Table 3.2: Inter-rater reliability of a human and term normalisation by the agent based on the final models in a pilot study ($n = 62$). Numbers refer to the nodes present and missing in the model.

κ = 0.94		**Human**	
		present	missing
Agent	present	289	12
	missing	5	252

3.4.4.2 Evaluating the model

The model evaluation agent provides feedback about the quality of the student model when compared to the reference model. Feedback is presented to the student in a bar chart (Figure 3.9).

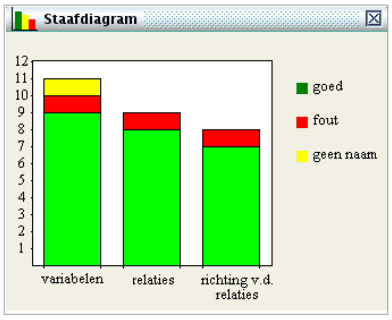


Figure 3.9: Bar chart showing informative feedback computed by the model evaluation agent.

The three bars represent the number of elements (constants, auxiliaries and stocks), the number of relations between these elements and the correctness of the directedness of the relation. Green means correct, red wrong and yellow are the anonymous model elements that have not been given a name by the student. The example in Figure 3.9 contains nine correct elements, one incorrect element and one unnamed element (left most bar), eight correct relations and one wrong relation (centre bar) and seven relations with the correct relation and one relation with an inverted direction (right most bar). The patterns in the rule set corresponding to the values in the bar chart are:

```
<node status="present"/>        % Left bar green
<node status="redundant"/>      % Left bar red
<node status="anonymous"/>      % Left bar yellow

<edge status="present"/>        % Centre bar green
<edge status="redundant"/>      % Centre bar red

<edge status="present" direction="correct"/>  % Right bar green
<edge status="present" direction="wrong"/>    % Right bar red
```

In the modelling task of the studies the notions of Ohm's law and parallel resistance are present. Ohm's law is $I = V/R$ (current is voltage divided by resistance) and parallel resistance is $R = 1/(1/R_1 + 1/R_2)$. If we substitute parallel resistance in Ohm's law we get: $I = V/1/(1/R_1 + 1/R_2)$. Whether to model Ohm's law and parallel resistance as two separate formula's or to combine them into a single formula is a modelling choice. Such modelling choices also have consequences for the surface structure of the model: there is no longer a need for R when the formula is substituted. This is illustrated in Figure 3.10. On the left is the structure without an explicit auxiliary for the combined parallel resistance R. The structure on the right, on the other hand, contains R and one can “see” Ohm’s law.

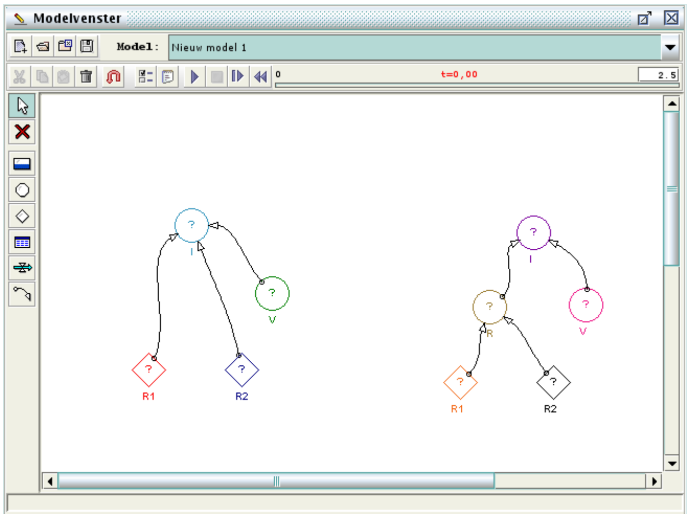


Figure 3.10: Modelling choices. The model on the right introduces an auxiliary to make the combined influence of R1 and R2 explicit. The model on the left hides the combined influence in the formula for I.

A consequence of these modelling choices is that a single reference model consisting of elements and relations is not sufficient. One approach would be to define multiple (partial) reference models (Bravo et al. (2006)) reflecting whether relations are explicitly modelled in the surface model or hidden in formulas. In the reference model in the studies, eleven different modelling choices have been identified and because many of these choices can be combined the number of reference models required becomes very large. For the development of the reference model transparent edges have been used. The researcher first defines the nodes corresponding to the elements:

```
<node term="I" type="auxiliary"/>
<node term="R1" type="constant"/>
<node term="R2" type="constant"/>
<node term="R" type="auxiliary"/>
<node term="V" type="auxiliary"/>
```

Next, the relations between nodes are specified as edges:

```
<edge tail="R1" head="R" qualitative="asymptotical"/>
<edge tail="R2" head="R" qualitative="asymptotical"/>
<edge tail="V" head="I" qualitative="unspecified"/>
<edge tail="R" head="I" qualitative="unspecified"/>
```

The evaluation of this reference model for the first modelling choice (left part of Figure 3.10) is four correct elements, one correct relation and two relations are wrong (R1 to I and R2 to I). For the second modelling choice (right part of Figure 3.10) all five elements are correct as are the four relations. Both modelling choices are accommodated by adding the following transparent edges to the reference model:

```
<t_edge tail="R1" head="I" excluded="R"/>
<t_edge tail="R2" head="I" excluded="R"/>
```

This reads as "accept an edge from R1 (or R2) to I in the student model provided that R does not appear in the model". With the transparent edges it becomes possible to define rules that accept the modelling choices as a correct structural representation.

3.4.4.3 Specifying rules

The model progression agent determines whether students are allowed to progress to the next modelling phase. In contrast to the model evaluation agent, which counts the number of matching nodes and edges in the student and reference model, the model progression agent requires a mechanism to express the rule(s) that determine whether progressing is allowed.

The examples of Ohm's law and parallel resistance (see the previous section) can be seen as sub-models within the overall model. If several of the sub-models are present in the student model, this likely reflects some understanding of the domain and model progression should be possible.

The pattern below states that the model must contain "light1", "light2" and relations between the lights and "total_resistance" with the correct direction.

```
<and>
  <node term="light1" status="present"/>
  <node term="light2" status="present"/>
  <edge tail="light1" head="total_resistance"
```

```
        direction="correct"/>
  <edge tail="light2" head="total_resistance"
        direction="correct"/>
</and>
```

For the model progression agent in the studies the rule that decides to move from the initial phase to the qualitative modelling phase consists of about 60 patterns. The rule from qualitative to quantitative modelling was similar, but also required the correct qualitative relation to be present in the model. See Appendix A.2.3 for the complete specifications. Both rules return either true (the student can progress to the next modelling phase) or false.

3.4.5 Results

The evaluation and model progression agents for the electrical circuits domain have been used in three studies. In the first and second study the evaluation agent made it possible for students to request an evaluation of the model during the first modelling phase (Mulder, Lazonder, & de Jong, 2011). In this phase, where students have to identify relationships between elements without specifying them, it was technically impossible for the model editor to execute these models (Mulder, 2012). For the third study, the model progression agent was made available to some students. The agent acted as a gatekeeper and only allowed students' to go to the next phase when their model was of sufficient quality. Students who obtained feedback from the model progression agent created better models than students who did not (Mulder, Lazonder, de Jong, Anjewierden, & Bollen, 2012).

For an analysis of the performance of the agents we take into account that a student potentially can request an evaluation of his model at any point in time. The rationale for this assumption is that the purpose of the agents is to give adaptive feedback when students want or need it, and it would therefore not be representative to only evaluate the agent on, for example, the final model created by the students. We also decided to use all the available data for the evaluation of the agents, including the models created by students who did not receive feedback. Such a full analysis is possible because the log files recorded all actions of the 238 students who participated in the three studies, as well as the 68,334 (intermediate) models they created.

Terminology normalisation is the most critical and difficult task for the agents. Computing the values for the bar chart in the case of the feedback agent, or deciding whether a student can progress to the next level for the progression agent, is deterministic once terminology normalisation has been performed. For the analysis of terminology normalisation quality a distinction is made between the accuracy of the normalisation itself, and what normalisation compensates for. The first question is addressed by manually performing the normalisation, and the second by considering the syntactic variations (e.g., synonyms, misspellings) students introduced in the labels.

For the analysis of the quality of terminology normalisation we extracted all labels students entered for the elements. After clean up (down casing, removing punctuation, etc.) 571 unique labels remained. The type of the model element (stock, constant, or auxiliary) is taken into account by the agent when matching labels to terms (as specified in the term set, see Appendix A.2.1). For example, if a student labels a stock as "condensator" (capacitor) it is assigned to "lading" (charge). And when "condensator" is a constant or auxiliary it is assigned to "condensator capaciteit" (capacity). Taking model element type into account resulted in 759 unique pairs of labels and types.

The researcher who conducted the studies scored the 759 pairs of labels and types as follows. The category *certainly correct* was assigned when the pair could only refer to a single model element: "batterij" (battery). Labels were scored as *certainly incorrect* when the pair could not possibly match any model element: "gloeidraad" (filament). When a label and type pair could not be scored as either correct or incorrect, it was scored as *ambiguous*. For these pairs it would be necessary to inspect the models in which they occurred to determine whether the pair could be assigned to a model element. Several of the ambiguous labels are of a descriptive nature, for instance "toename opladen condensator capaciteit" (increase charge of capacity). Other ambiguous labels are ambiguous, for example "ohm" or "volt".

Table 3.3: Confusion matrix containing agreement between researcher and agent on assigning model elements based on the labels students used. Certainly correct means that the label could unambiguously be assigned to a model element, and certainly incorrect that the label should not be assigned to a model element. In 18,239 cases (not in the table) the researcher could not assign with certainty whether a label matched a particular element.

κ = 0.908	**Agent**		
		Correct	Incorrect
Researcher	Certainly correct	259,714	111
	Certainly incorrect	623	3,721

Table 3.3 provides the confusion matrix. The numbers in the table reflect the total

number of occurrences of labels. For example, "batterij" (battery) occurred in 5,519 models as a constant and contributes 5,519 to the certainly correct agree cell in the table.

The agreement between the human rater and the agent for the certainly correct and incorrect labels is 0.908 (Cohen's κ) and the precision is 0.997. 6.49% (18,239) of the labels are scored as ambiguous by the human rater. In this case ambiguous means the rater needs to assess the structure and the other elements in the model to assign the appropriate term. Given that the agent's algorithm is syntactical in nature, it is very unlikely to perform well on the ambiguous labels. When all ambiguous labels would be normalised wrongly by the agent, precision drops to, a still respectable, 0.932.

The 259,714 labels for which the rater and the agent agree to be certainly correct were analyzed further to determine the types of normalisation the algorithm performed. The results are in Table 3.4. More than 75% of the labels were spelled exactly as one of the strings in the term set. Synonyms in the term set were necessary for a match in 31.28% of the cases. A different word order was found in nearly 19% of the labels, mainly for "spanningsbron" versus "bronspanning" (power source), and "linker lampje" (left light) versus "lampje links". Inflected words occurred in 6.3% of the labels, mainly plurals and a few diminutives. Spelling errors were found in slightly more than 5% of the labels. Redundant, extra words were found in 2.94% of the labels. The totals add up to more than 100% because a label can belong to more than one of the categories. For instance, a label can contain both a misspelling and be a synonym.

Table 3.4: Terminology normalisation by type

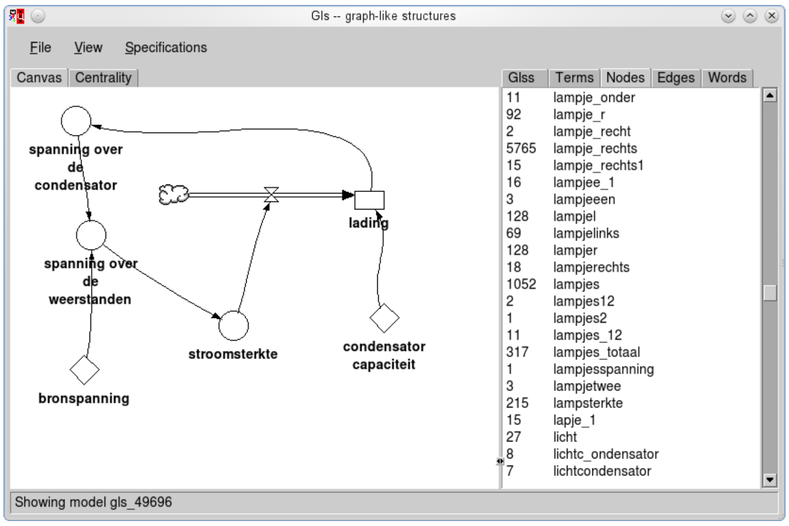
Normalisation	N	
All	259,714	100.00%
Exact match	197,134	75.90%
Synonym	81,215	31.28%
Word order different	48,991	18.86%
Inflected	16,367	6.30%
Misspelling	13,428	5.17%
Extra words	7,641	2.94%

The overall quality of terminology normalisation was acceptable. There may be several factors that contributed to this. The simulation in the learning environment and the help files contain relevant terms. Another important contributing factor may be that the term set, including the many synonyms, was carefully constructed based on a pilot study. Although students may have some idea which label to use, some labels are conceptually so close that selecting one or the other can be a matter of taste. An example is the aforementioned "spanningsbron" (power source; a role) versus "batterij" (a type of power source). One student used "stopcontact" (wall plug), which is a very special type of power source. A further example is "weerstand" (resistance; role) versus "lampje" (light; has resistance). Given the many synonyms used in this domain it is difficult to obtain acceptable matching without them. The similarity flooding algorithm used by Marshall et al. (2006) will therefore perform much worse in this domain than the term set approach. A surprise contribution to the quality of matching may have come from the evaluation agent itself. When a student enters an element into the model that is not recognised by the agent, the bar chart will contain a red bar. If the student subsequently corrects the element, the bar turns green. The outcome is then a symbiosis between student and agent. The student's model has improved because of the feedback, and the performance of the agent also improves because of the correction by the student. An indication this indeed happens is that for the students who obtained feedback from the evaluation agent 1.42% of the labels could not be matched, whereas this was 2.44% for students who did not get help from the evaluation agent.

An evaluation of the overall structure of the models and the application of the rule sets was supported by the tool shown in Figure 3.11. For the first and second study, the researcher assessed the students' final models and the output of the evaluation agent in detail. For the third study, when the progression agent was introduced, both the final models and the models at the time of a phase change were assessed. Based on the assessments, the researcher saw no reason to change either of the term set, reference model or rule sets. The raw data for the descriptive statistics in Mulder et al. (2011, 2012) were generated by the same software the agent is based on.

3.5 Evaluation of concept maps

In this section we give examples of agent-based support for the analysis of concept maps. As stated earlier concept maps and models share the graph-like structure, and the analysis of a concept map is, given the generic approach, the same as that for models (see Section 3.4). There are, however, differences in how models and concept maps are used in instruction. Modelling is often a self-contained activity, with the task of creating a model for a certain domain. For concept mapping there



11	lampje_onder
92	lampje_r
2	lampje_recht
5765	lampje_rechts
15	lampje_rechts1
16	lampjee_1
3	lampjeeen
128	lampjel
69	lampjelinks
128	lampjer
18	lampjerechts
1052	lampjes
2	lampjes12
1	lampjes2
11	lampjes_12
317	lampjes_totaal
1	lampjesspanning
3	lampjetwee
215	lampsterkte
15	lapje_1
27	licht
8	lichtc_ondensator
7	lichtcondensator

Figure 3.11: Tool supporting manual inspection of graph-like structures. Left is a model. The terms recognised by the agent are shown in bold, an alternate view (not shown) is to use the labels students used. Right are browsers listing summaries of model content. Shown are labels students used for the nodes.

are many variations in both the task students have to perform and the role concept mapping plays. Ruiz-Primo & Shavelson (1996) identify concept mapping tasks like: organize a given set of concepts, fill in a partial map, or construct a map from scratch.

3.5.1 Support for concept mapping tasks

We describe different concept mapping tasks based on a concept map evaluation agent developed for the SCY project[4]. The central pedagogical approach in SCY is organised around the principle that learning is seen as producing emerging learning objects (ELOs) (de Jong et al., 2010). The learning environment of SCY is called SCY-Lab. SCY-Lab consists of tools to manipulate and view ELOs (e.g., concept mapping tool, data analysis tool, etc.), and communication and collaboration facilities. The infrastructure of SCY-Lab makes student actions available for agents to analyse, almost identical to the architecture described in Section 3.4.3. For a domain of learning SCY-Lab is configured according to a selected instructional design, which results in

[4]`www.scy-net.eu`

a specific SCY learning environment based on a pedagogical scenario and populated with domain content (de Jong, Weinberger, Girault, Kluge, Lazonder, Pedaste, Ludvigsen, Ney, Wasson, Wichmann, Geraedts, Giemza, Hovardas, Julien, van Joolingen, Lejeune, Manoli, Matteman, Sarapuu, Verkade, Vold, & Zacharia, 2012). A specific SCY learning environment is called a SCY “mission”. In several missions one of the initial ELOs students create is a concept map to familiarize themselves with the domain. Students use the SCY concept mapping tool to manipulate their concept map and when they save the map, the concept map evaluation agent is activated.

3.5.1.1 Organize a given set of concepts

The pizza mission, on nutrition, use of energy and healthy food, contains a concept map called the energy fact sheet, see Figure 3.12. In this concept map all concepts and the structure of the map are given in advance: an “organize a given set of concepts” concept mapping task according to Ruiz-Primo & Shavelson (1996). Students complete the map by dragging-and-dropping concepts that are not connected in the initial map to the correct location. In the figure there are still four concepts (right) that need to be dragged to one of the empty slots.

Before evaluation can take place, the agent first replaces each empty concept with the concept the student dropped on top of it. Feedback is simply the number of concepts the student dropped on the correct location. To be able to do this the agent needs to know which concepts are droppable and which are not. This is specified in the reference model with the “drop” attribute:

```
<node term="calorie" drop="true"/>
<node term="thinking" drop="true"/>
<node term="running" drop="true"/>
...
```

Evaluation proper starts when the student has dropped all droppable concepts. Until that happens he gets the message: “There are still *X* concepts that need to be dragged to an empty location”. After having dropped all concepts, the message becomes: “You have *X* concepts correct and *Y* are wrong”. When a learner has completed the map he is congratulated with: “You have placed all concepts correctly. Congratulations!”.

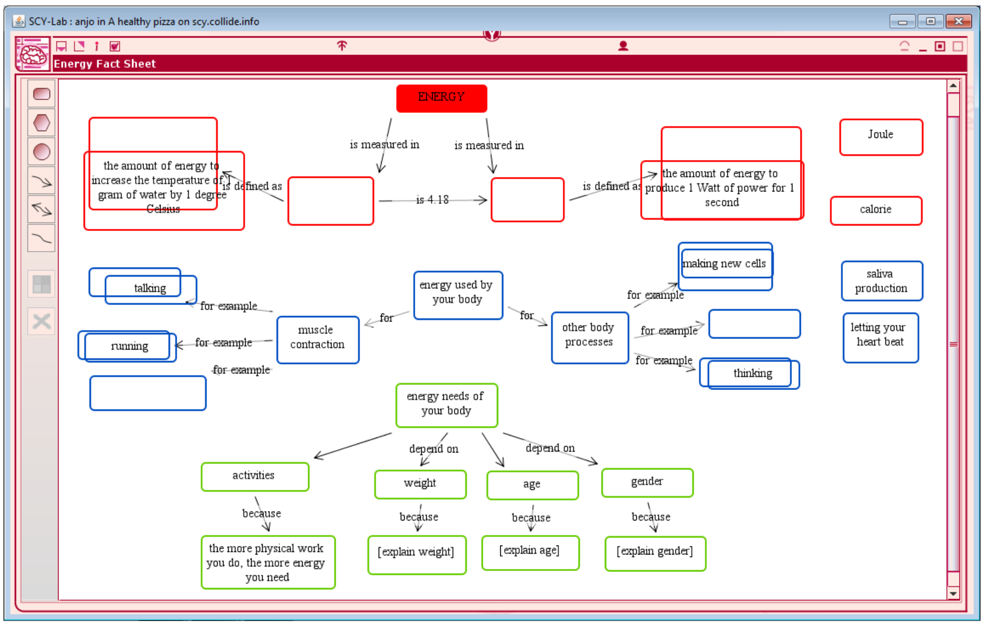


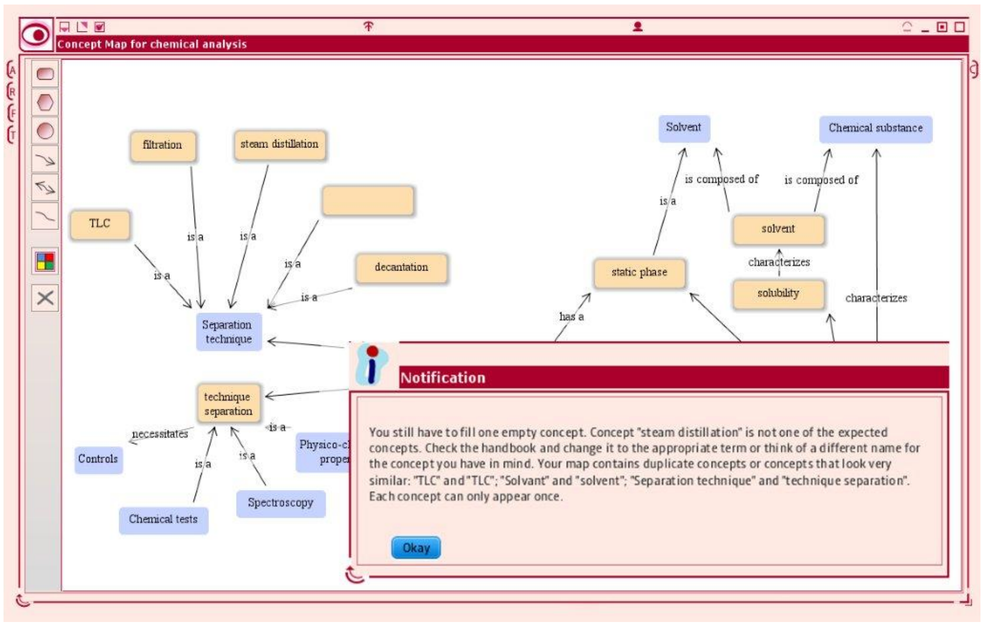
Figure 3.12: SCY concept mapping tool to organize a given of concepts about energy use and nutrition. Students drag-and-drop concepts to the correct location.

3.5.1.2 Fill in a partial concept map

The map in the forensic mission is about concepts related to chemical processes in forensic science. As with the energy fact sheet the complete structure is given, but some concept labels are blank, see Figure 3.13. The task of the student in this case is to fill in these blanks with the appropriate concept: a "fill in a partial map" concept mapping task according to Ruiz-Primo & Shavelson (1996).

Students get feedback about three aspects of the evaluation. Firstly, as with the energy fact sheet, the student is informed about the number of blanks that still need to be filled. Secondly, if the agent does not recognise a label (a redundant concept) the student is informed about this. Finally, the student is informed about concepts that are not in the correct location in the map. This can result in lengthy messages, like the following in French and English:

> Il vous manque un concept.
> Le concept "fumant" ne fait pas partie de la liste des concepts attendus.
> Afin de le remplacer par un autre concept, relisez le manuel de labora-



You still have to fill one empty concept. Concept "steam distillation" is not one of the expected concepts. Check the handbook and change it to the appropriate term or think of a different name for the concept you have in mind. Your map contains duplicate concepts or concepts that look very similar: "TLC" and "TLC"; "Solvant" and "solvent"; "Separation technique" and "technique separation". Each concept can only appear once.

Figure 3.13: Concept map on chemical processes in the SCY forensic science mission. Structure of the map and some concepts (blue) are given. Students have to fill in the blanks. The notification window at lower right shows an evaluation message.

> toire ou bien cherchez un synonyme.
> Les concepts “eluant” et “identification technique” ne sont pas bien placés. Relisez le manuel de laboratoire afin de leur trouver dáutres places.

> You still have to fill one empty concept.
> Concept “fumant” is not one of the expected concepts. Check the handbook and change it to the appropriate term or think of a different name for the concept you have in mind.
> The concepts “eluant” and “identification technique” are not in the right place. Check the handbook to find another place for these concepts.

The concept labels in the messages are those used by the students, for example “eluant” rather than the reference model node “mobile phase”. For this concept mapping task the agent needs to know which concepts are initially blank and have to be filled in by the student:

```
<node term="mobile phase" fill_in="true"/>
```

3.5.1.3 Construct a concept map from scratch

The concept mapping task in the ecology mission is open-ended. Students start with an empty map: a ″construct a map from scratch″ concept mapping task according to Ruiz-Primo & Shavelson (1996).

In the ecology mission is broken down into four topics in the ecology domain: primary production, photosynthesis, thropic levels and pH. Students first follow an inquiry cycle on the topic and update their map with the concepts they encountered in that cycle. The agent therefore has to know which inquiry cycle (topic) they come from and select the appropriate rule from the rule set to perform the evaluation. For example when a student saves a hypothesis related to the photosynthesis topic this is recorded and when the concept map is later saved it is evaluated with the photosynthesis rule.

```
<rule name="B2 - photosynthesis" type="logical">
  <true at_least="5">
    <node term="CO2" status="present"/>
    <node term="Water" status="present"/>
    <node term="O2" status="present"/>
    <node term="glucose" status="present"/>
    <node term="energy" status="present"/>
    <node term="light" status="present"/>
    <node term="photosynthesis" status="present"/>
  </true>
</rule>
```

Because the concept mapping task is open-ended, it is possible that some students cannot complete the map on their own. The agent helps by presenting these students with a better map of a class mate. The feedback initially is: “That is a good start but you still miss *X* concepts.” Next, either of “The concept map has improved. *X* concepts are still missing” or ″The previous concept map had more correct concepts.″ or ″Your concept map has not improved. If you want me to look for a good concept map from a class mate then click the save button again″. In the last case, the agent searches for a better concept map from a class mate and presents that map to the student. If no better concept map can be found the message is “I did not find any class mate with a better concept map.”. When the concept map for a topic is complete, the student is congratulated. In Estonian this reads: “Palju õnne!”.

3.6 Discussion

In this chapter we have described an approach, implementation and several applications of agent-based analysis of graph-like structures created by students. The main motivation was to make it possible to create learning environments in which students obtained online feedback about the graphs they created. It is sometimes thought that automatic analysis is a technological problem (e.g.,Marshall et al. (2006)), but we think the problem is selecting or creating the right technology. There are three stakeholders that determine whether automatic analysis is useful and successful: students, researchers, and developers of agents. We discuss the role of these stakeholders in what has been achieved.

Graphical representations are attractive for students to externalize their knowledge in a domain. Concept maps have an appealing spatial layout that is easier to grasp than a textual representation containing the same knowledge. For models something similar applies, a graphical representation of a model is easier to create and manipulate than sets of formulas. Graphs are thus attractive for education. and they are widely used, especially in science education where there is a close relation with modelling as a tool to uncover relations, and concept mapping for knowledge representation

Creating graphs still holds several challenges for students. First of all they have to become acquainted with the graphing tool and the underlying notation. In open-ended concept mapping students initially often have difficulty to make a distinction between concepts and relations. And, as a consequence, relations become concepts. A second challenge students face is to think of appropriate labels for the concepts and relations. Finally, the structure that represents the student's knowledge of a domain has to be created.

Researchers have to decide on the instructional context and how the learning environment and agents contribute to this context. For example, in the current version of Betty's Brain, students are relieved of defining the concepts. They select concepts from a list, and the task is to define the correct structure and define causal relations. This is also the case for some of the concept mapping tasks described in Section 3.5, where students are either give the structure and arrange the concepts, or are given the concepts and have to create the correct structure. The system dynamics studies described in Section 3.4 are an example in which students faced both challenges: think of the labels and create an appropriate structure. In these studies, students were supported by the learning environment through instructional material that contained some of the correct labels, and a simulation which contained part of the structure and the labels.

The contribution of the approach and the implementation of the agents is addressing the problems that matter. In the system dynamics studies an instructional context in which the labels are provided, for instance "resistance 1", "resistance 2" and "total resistance", would have changed students' task dramatically. Terminology normalisation, taking into account that students use different terminology and make spelling errors, is a crucial first step that must be addressed. In the approach we used the researcher can specify the terminology and a variation on the well-known edit distance algorithm appears well-suited for normalisation itself. The analysis of edges is trivial once the head and tail nodes of an edge are known (after normalisation). The other important problem addressed is that of catering for multiple scenario's. The use of reference models to specify the structure of the domain has been used by others. Rule sets, in which researchers can specify how the analysis of student graphs is performed and how feedback is given, is new. Rule sets provide researchers with a flexible mechanism, using intuitive terms (present, missing), to specify the analysis. The researcher who conducted the system dynamics studies summarized the experience, and to some extent this thesis, as follows:

> "For future research the software agent holds promise as it enables a more sophisticated and possibly effective approach to give adaptive support on the students' actions. A software agent can detect patterns in the students' inquiry and modelling activities, and use this information to give tailor-made assistance and feedback at times appropriate." (Mulder, 2012, p. 105–106).

Chapter 4

Examining the relation between domain-related communication and collaborative inquiry learning

4.1 Introduction

In a collaborative learning setting two or more students share and construct knowledge while they work towards the solution of a problem or assignment. Research has shown that collaboration between students may enhance learning (Lou, Abrami, & d'Apollonia, 2001; Slavin, 1994; van der Linden, Erkens, Schmidt, & Renshaw, 2000). To maintain a successful collaborative working relationship, ideas and theories must be externalized and explained in a mutually understandable way for the partners in the collaborative learning group (Teasley, 1995). Through externalization students express and explain ideas, ask for clarifications or arguments and might generate new ideas. The process of making ideas public through externalization and explanation, stimulates students to rethink their own ideas and might even make them aware of possible deficits in their reasoning (Cox, 1999; van Boxtel, van der Linden, & Kanselaar, 2000). Research indicates that the degree of participation in collaborative activities is related to group performance as well as students' individual learning (Fischer & Mandl, 2005).

Nowadays collaborative learning often takes place in computer supported collaborative learning (CSCL) environments that combine collaborative learning with ICT-based applications for learning. Within these CSCL environments student interaction usually is mediated by text-based communication tools like discussion boards or chat tools. Peer interaction is believed to facilitate learning in CSCL settings (Gijlers & de Jong, 2009). Analysis of interaction protocols may contribute to the under-

This chapter is based on A. Anjewierden, H. Gijlers, B. Kollöffel, N. Saab & R. de Hoog (2011). Examining the relation between domain-related communication and collaborative inquiry learning. *Computers & Education* 57, 1741–1748.

standing of the processes that contribute to meaningful and effective collaborative learning. These insights are not only relevant from a theoretical perspective but also from a practical point of view, for example the development of software tools that can analyze ongoing chat communication between students and provide these students with guidance (e.g., prompts, questions, hints) on the basis of what they are chatting about.

Protocols recording collaborative learning can be analyzed from several perspectives. They have been analyzed in terms of students' degree of participation, communicative activities such as arguments and elaborations (van Boxtel et al., 2000), different learning processes (Hmelo-Silver, 2003; Saab, 2005), or the exchange of domain-related information (van Drie, van Boxtel, Jaspers, & Kanselaar, 2005). In this study we focus on the collaborative construction of domain-related knowledge, by examining the communication protocols of dyads who together interacted with a collaborative inquiry learning environment. Within a dyad, students may not only differ in their overall degree of participation (Cohen, 1994), the characteristics of students' contributions may also differ. Being interested in domain-related knowledge construction, our focus is on students' externalizations of domain-related conceptions and their interpretation of information obtained from the learning environment or provided by their partner. More specifically, we are interested in the degree of domain-related information each partner contributes to the dialogue. In order to maintain a successful collaborative relationship, students need to reach a certain level of consensus about the task and their plans. The way students try to reach consensus may differ with respect to the level of domain knowledge students externalize. Students may accept their partner's contributions without fully understanding them or agreeing with them, because they are interested in completing the collaborative task. Weinberger & Fischer (2006) argue that these quick consensus building activities assist students in managing the collaborative learning process, but are not directly related to knowledge acquisition.

The domain related information externalized by students differs in terms of complexity. The mere externalization of simple domain related information without reference to other domain related concepts or prior knowledge can contribute to students definitional knowledge about the domain but is not likely to facilitate students' understanding of complex relations within the domain. However, the exchange of elaborated externalizations and explanation, including statements about relations in the domain, can enhance students' understanding of relations in the domain (Webb, 1989; Gijlers & de Jong, 2009).

Research question and hypothesis

Based on the considerations presented above, this study examines how students' domain-related contributions are associated to their individual learning outcomes. The simulation-based inquiry learning environment used in this study required students to focus on relations between variables in the domain. Based on the theories described above, it was expected that within a dyad, a student who externalizes a higher proportion of domain-related knowledge in that dyad than the partner, reaches a higher post-test score than students who externalize less domain-related knowledge during the learning session.

4.2 Method

4.2.1 Learning environment and task

Students worked in dyads with an inquiry learning environment that was based on a computer simulation of colliding objects. Inquiry learning environments are very suitable for collaborative learning. In a simulation based inquiry environment students learn through experimentation and scientific reasoning. The interface of the learning environment allows students to change input variables and observe the effects of their actions. Students learn about the principles and rules of scientific phenomena through processes like hypothesis generation, experimentation, and conclusion (e.g., de Jong & van Joolingen, 1998). During inquiry learning, students must make many decisions (e.g., which hypothesis to test, what variables to change) and in a collaborative setting, the presence of a partner stimulates students to make their plans and reasoning about these decisions explicit (Gijlers & de Jong, 2009).

In the current study the main task for the students was to discover the laws of physics underlying the simulation. Dyads worked collaboratively on two separate computers with a shared interface and communicated through a chat channel (based on Microsoft Netmeeting technology). All chat messages were automatically recorded. Learning material consisted of four progression levels accompanied with assignments that presented the learners with small research questions to guide their inquiry learning process. A total of 35 assignments were available in the learning environment. This learning environment provided sufficient variety in terms of knowledge to be mastered to be able to detect different effects of types of communication contributions on the domain knowledge. The learning environment was created in SimQuest (de Jong & van Joolingen, 1998).

4.2.1.1 Participants

Dyads were heterogeneous with respect to students' school achievement in the domain of physics (this information was provided by the participating schools). This grouping was based on the finding that heterogeneous grouping is beneficial for both high and low achieving students (Webb, Farivar, & Mastergeorge, 2002). Students were paired with a student from their own class. Participants attended two sessions. In the first session dyads were composed and students practiced in dyads with a SimQuest practice simulation that allowed them to explore the features of the interface and work with the chat tool. The second session started with the two pretests, followed by 90 minutes of interaction with the simulation environment and the chat facility. At the end of the session students completed the post-test versions of the knowledge tests.

4.2.1.2 Tests

As the nature of a student's contribution can vary between externalizing simple domain related information to explaining complex relations in the domain, the students' individual learning outcomes were assessed with two domain knowledge tests. A "definitional knowledge" and a "what-if" test for intuitive knowledge were administered. The definitional knowledge test (15 items) focused on domain relevant facts and formulae. Since experimentation within an inquiry learning environment is believed to foster intuitive knowledge about relations in the domain rather than definitional knowledge an intuitive knowledge test in the "what-if" format (13 items) was administered (Swaak & de Jong, 2001). Each question of the "what-if" test consists of three parts: condition, action and prediction. First a condition/situation before a collision is presented to the students. Subsequently, the action (for example, a collision against a fixed wall) is presented. Finally, three predicted states are presented to the students either in text or pictures. Students are asked to select the state that follows from the action in the given condition. The definitional knowledge test as well as the "what-if" test was computer administered and pre- and post-tests were parallel versions of the same test.

4.2.2 Determining learner contribution in dyads

The question we are addressing is whether there is a relation between the nature of the domain-oriented contribution of a learner and improvement on knowledge tests. In this section we first give some examples of the interaction between learners, and next motivate our choice of how to develop a method that allows answering the research question. An example of an excerpt of the interaction between two learners, called X and Y, is:

1	12:58:48	X:	if the mass becomes higher, the momentum decreases, I think
2	12:59:43	Y:	no, that is not true
3	13:00:02	Y:	p also is higher, have a look
4	13:00:03	X:	experiment
5	13:00:16		*Running a simulation*

Here X thinks there is a qualitative relation between mass and momentum: if the mass increases, the momentum decreases (line 1). Y, after about a minute, realizes that the relation is incorrect (line 2) and suggests that the momentum, the symbol p stands for momentum, increases when mass increases (line 3). X suggests to do an experiment to find out more (line 4) and runs the simulation. This brief example illustrates learners X and Y share their individual understanding of the domain and try to reach a joint understanding.

Excerpts like the one above are not very common. Below is an example, between learners P and Q, which follows a pattern that is much more frequent:

6	13:31:33	P:	speed is the same after the collision
7	13:31:37	Q:	yes
8	13:32:04	Q:	4
9	13:32:12		*Answer 4 is selected, it is incorrect*
10	13:33:11	P:	1
11	13:33:25	Q:	ok
12	13:33:29		*Answer 1 is selected, it is correct*

P communicates a domain-related observation (line 6). Q immediately agrees (line 7) and after about half a minute proposes 4 as the correct answer (line 8), the answer is wrong (line 9). Then P proposes 1 as the answer (line 10), Q agrees (line 11) and the answer turns out to be correct (line 12). The pattern is that one of the learners exchanges a domain-related finding (line 6) and the partner only acknowledges this without referring to domain-related terms. Often the entire discussion then switches to what answer to give. If this happens repeatedly, P may at some point decide to give up formulating domain-related messages all together as Q does not appear to do anything with them.

In order to determine what the level of domain-related contribution of each learner is, we introduce some abstractions. The level of domain-related contribution of a learner is denoted as $D(learner)$. Based on the example excerpts above, we have

$D(X) \approx D(Y)$ because X contributes one domain-related statement (line 1) and Y does the same (line 3) and $D(P) > D(Q)$ because P contributes one domain related statement (line 6) and Q does not contribute any. It would perhaps be tempting to conclude that $D(X) > D(Q)$ and $D(Y) > D(Q)$, however, we are not allowed to do this. The reason is that the value of $D(X)$ is dependent on the collaborative setting with Y, which we denote as $D(X|Y)$, whereas the value of $D(Q)$ depends on the collaborative setting with P, as is illustrated in the two example dialogues above. If we would like to estimate the "true" value of $D(X)$ in other collaborative settings a better approximation is:

$$D(X) = average(D(X|Y) + D(X|P) + D(X|Q))$$

Given that we have no values for $D(X|P)$ and $D(X|Q)$, $D(X)$ cannot be computed. The same goes for the other three learners in the example, and we can therefore not compute an estimate for any $D(learner)$ that can be meaningfully compared with D estimates for other learners.

An intuitive example, for comparison purposes, is the idea of a marathon run. Two runners P and Q decide to beat the world record. They agree that during the first 35 kilometers Q acts as a pacemaker and runs in front. P wins the race in a record 2.03:00, Q finishes in 2.04:00. A year later, under precisely the same circumstances, P′ and Q′ also want to beat the world record, but P′ and Q′ do not make any prior agreements, each runs his own race, P′ finishes in 2.03:30 and Q′ in 2.04:30. All other things being equal, it is not allowed to conclude that P is faster than P′ as their performance depends on having a runner as a pacemaker or not. Although the conclusion is drawn in practice, otherwise there could not be a world record for the marathon, it is hardly justified and proposals to ban pacemakers from marathons are based on these considerations.

To summarize: it is not possible to obtain a good estimate for the value of the contribution of a learner in a dyad (or larger group of learners) that can be compared to other learners in other dyads, even in the same experimental setting. This may be important for CSCL (Computer-Supported Collaborative Learning) in general, as it points to a major methodological issue, and might also explain why previous research has not related individual performance to individual learning outcomes in a collaborative learning setting and why researchers carefully consider the composition of the dyads or groups (Webb, Nemer, Chizhik, & Sugrue, 1998).

Does this prevent us from finding out whether the contribution measure, $D(learner)$, is related to learning outcome? We think this is still possible by using an indirect method. Within a dyad, the learner with the highest value for the level of contribution measure is added to a group of learners called A, the other learner in the same dyad is added to a group called B. This is like a knock-out competition, the A's would be the winners the B's the losers. If the average learning outcome of the A's is significantly different from that of the B's, the level of contribution measure is the probable cause, because this was the reason for partitioning the learners into A's and B's.

4.2.3 Classifying contributions

Partitioning learners into A's and B's, as proposed above, requires a measure of a learners' contribution to the dialogue, that allows us to do the assignment as being A or B. Inquiry learning environments, like the one used in this study, stimulate the acquisition of knowledge about relations in the domain. For example, in physics momentum is defined as $p = mv$, where p is momentum, m is mass, and v velocity. Learners can find out about these kind of relations by changing one of the variables and (graphically) inspecting the effect on the others. Learners can share their thinking on the relations with their partner through the chat tool. To determine the level of domain-oriented content of a message, we distinguish three types of domain-related contributions:

- *Domain terms*. The use of domain terms, such as *velocity*, *increases* and the abbreviation v, transfers at least a certain domain focus by the learner. The message *shall we look at mass* contains one domain term.
- *Qualitative statements*. These are phrases containing both a variable and a qualitative relation, for example *speed increases* or *momentum is lower*. We do not make a distinction between qualitative statements that result from an observation (*speed increases*) and qualitative statements that suggest future action (*shall we increase the mass*). All such statements demonstrate a clear domain focus. It should be noted that in the analysis the correctness of a statement is not taken into account.
- *Conditional sentences involving qualitative statements*. These are evidence of interpretations or hypotheses related to the domain. A conditional sentence is the grammatical construct which relates a condition to a consequence. In the collision domain, an example is *if the mass becomes higher, then momentum decreases*.

We also consider *Agreements* as relevant to determine domain-related contributions. Simply agreeing, using *ok* or *yes*, is often a syntactic indication for quick consensus building (Weinberger & Fischer, 2006) or leaving the thinking to their peer. The excerpt of the chats between P and Q in the previous section provide an illustration.

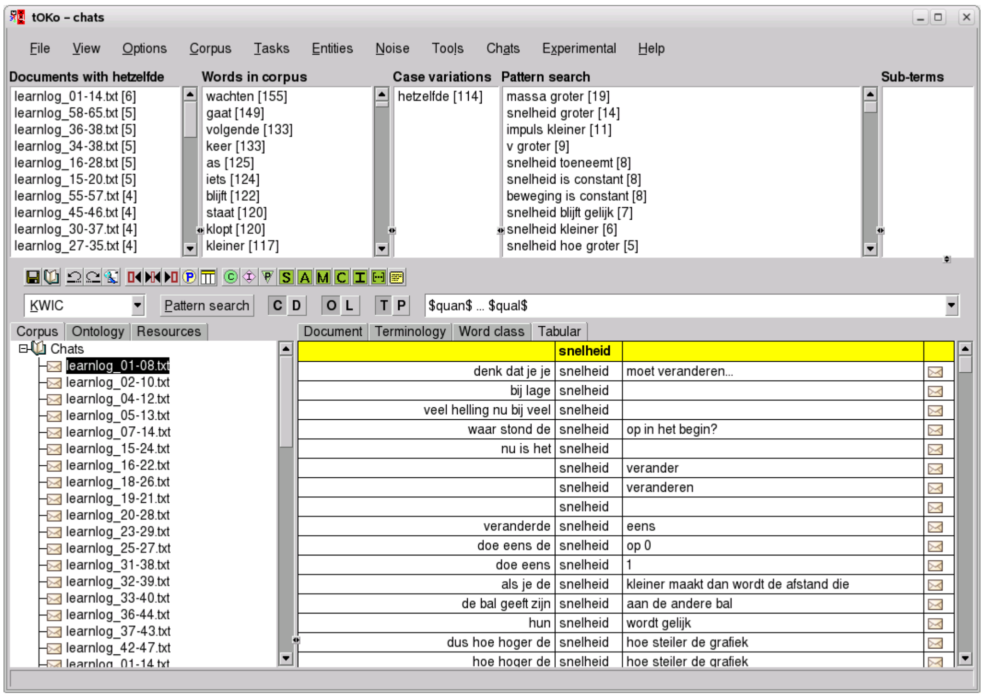
4.2.4 Analysis of the messages

Identifying the above three types of contributions in chat protocols, requires an analysis, or semantic interpretation, of the message. A human can, if the message is "velocity increases", reason that "velocity" is a quantity, and "increases" a qualitative relation. It therefore may be concluded that the message is a qualitative statement. Interpretations of this kind are very different from categorical coding, the usual method of analysis in the behavioural sciences in general and CSCL in particular. In (automatic) categorical coding, often specific marker words, phrases, or actions are used to determine the category or function of a message (see e.g., Erkens & Janssen, 2008; Rosé et al., 2008; Anjewierden & Gijlers, 2008). However, interpreting the semantics of messages like *velocity increases* requires a more flexible approach.

We have used a text analysis tool called tOKo (Anjewierden, 2006). This tool is being used by social scientists to study, for instance, online communities (e.g., de Moor & Anjewierden, 2007), and by semantic web researchers to create domain vocabularies and extract semantic relations (e.g., de Boer, van Someren, Wielinga, & Anjewierden, 2010). Automatic (semantic) text analysis is uncommon in CSCL, so we provide a global overview of how we applied tOKo to the analysis of the chat messages and omit technical detail where possible. The analysis starts with a corpus that contains all the chats and the objective is to define syntactic patterns in these chats that allow the extraction of the types of contributions we are interested in. For this we must identify the features detailed below.

Domain terms. The domain terms have been selected by sorting all words on frequency and manually selecting domain terms that occur at least five times. The most frequent word, ignoring stop words, is *ok* (1294), the most frequent domain term is velocity (*snelheid*, 440). Identifying the domain terms with tOKo took about two hours.

Qualitative statements. The extraction of qualitative statements consists of three steps. The first step is to enumerate all quantities and all qualitative relations. Next, for each quantity and each qualitative relation we need to find the terms learners use for them. For example, the quantity velocity can appear as *velocity*, *speed*, and v (the symbol). Similarly, the qualitative relation *increase* can also appear as *goes up*, *higher* and so forth. In our Dutch chats we found seven syntactic patterns for increase and eight for constant, including the negation *does not change*. Finding these variations is not difficult, the easiest method is to use a quantity as a key word and use a concordance index to inspect the surrounding text. The lower right pane of Figure 4.1 shows the results for *snelheid* (velocity). In the central column the concept snelheid is displayed, and to the left and right the surrounding text, for example, the first line "denk je dat je je snelheid moet veranderen" (do you think you must change your velocity).



Documents with hetzelfde	Words in corpus	Case variations	Pattern search	Sub-terms
learnlog_01-14.txt [6]	wachten [155]	hetzelfde [114]	massa groter [19]	
learnlog_58-65.txt [5]	gaat [149]		snelheid groter [14]	
learnlog_36-38.txt [5]	volgende [133]		impuls kleiner [11]	
learnlog_34-38.txt [5]	keer [133]		v groter [9]	
learnlog_16-28.txt [5]	as [125]		snelheid toeneemt [8]	
learnlog_15-20.txt [5]	iets [124]		snelheid is constant [8]	
learnlog_55-57.txt [4]	blijft [122]		beweging is constant [8]	
learnlog_45-46.txt [4]	staat [120]		snelheid blijft gelijk [7]	
learnlog_30-37.txt [4]	klopt [120]		snelheid kleiner [6]	
learnlog_27-35.txt [4]	kleiner [117]		snelheid hoe groter [5]	

$quan$ ... $qual$

Chats
- learnlog_01-08.txt
- learnlog_02-10.txt
- learnlog_04-12.txt
- learnlog_05-13.txt
- learnlog_07-14.txt
- learnlog_15-24.txt
- learnlog_16-22.txt
- learnlog_18-26.txt
- learnlog_19-21.txt
- learnlog_20-28.txt
- learnlog_23-29.txt
- learnlog_25-27.txt
- learnlog_31-38.txt
- learnlog_32-39.txt
- learnlog_33-40.txt
- learnlog_36-44.txt
- learnlog_37-43.txt
- learnlog_42-47.txt
- learnlog_01-14.txt

	snelheid		
denk dat je je	snelheid	moet veranderen...	✉
bij lage	snelheid		✉
veel helling nu bij veel	snelheid		✉
waar stond de	snelheid	op in het begin?	✉
nu is het	snelheid		✉
	snelheid	verander	✉
	snelheid	veranderen	✉
	snelheid		✉
veranderde	snelheid	eens	✉
doe eens de	snelheid	op 0	✉
doe eens	snelheid	1	✉
als je de	snelheid	kleiner maakt dan wordt de afstand die	✉
de bal geeft zijn	snelheid	aan de andere bal	✉
hun	snelheid	wordt gelijk	✉
dus hoe hoger de	snelheid	hoe steiler de grafiek	✉
hoe hoger de	snelheid	hoe steiler de grafiek	✉

Figure 4.1: Using the tOKo text analysis tool on the collision chat corpus

Conditional sentences involving qualitative statements can be found using discourse markers and they correspond to the categories called condition and consequence in the MEPA interaction analysis tool (Erkens & Janssen, 2008). We have identified several syntactic patterns in the corpus which mark a conditional sentence. The most frequent is *als ... dan ...* (if, then), also frequent is *hoe ... hoe ...* (the, the; the higher the speed, the lower the mass) and sometimes *... wanneer ...* (when) is used. In the latter case, the condition and the consequence are reversed (speed increases when mass is increased).

We now have a set of syntactic patterns for quantities which we call $quantity$ and one for qualitative relations which is called $relation$. The third step is to define patterns for qualitative statements as a whole, they are:

```
$quantity$ ... $relation$
$relation$ ... $quantity$
```

The first pattern finds phrases in which the quantity appears before the relation and the second pattern finds phrases in which the quantity appears after the relation (increases speed). The ellipses is a construct part of the tOKo pattern matching language. Ellipses match a small amount of intermittent text (at most three words).

In total the selection of the terms and the definition of the patterns took about three days. Once the patterns are specified, the analysis is automatic. tOKo extracts zero or more clauses from each message. For example, from the utterance *the velocity increases* it extracts:

```
domain_term(velocity).
domain_term(increase).
qualitative_statement(velocity, increase).
```

From *speed becomes larger* precisely the same qualitative statement is extracted, although the two phrases have not a single word in common.

It should be noted that the manual part of the analysis appears to be time consuming, but one should realize that this upfront work greatly reduces the amount of work that otherwise would go into a manual analysis of thousands of chat messages (in this study about 18,000). In other words: the investment in identifying syntactic patterns is easily recovered by reducing the investment in the analysis of a large number of chat messages.

We have, of course, omitted several details in the above description. Not every conditional that begins with if also contains the corresponding then, certainly not in chat text, and it still should count as a conditional sentence. Another, relatively common case when mismatches occur is an overrun of one pattern match into another (Dutch readers will recognize an example in the last line in Figure 4.1, under the heading Pattern search: *snelheid hoe groter*). The tOKo pattern language provides a mechanism to suppress such matches.

The reliability of extraction is difficult to assess, the patterns were derived from the corpus and testing on the same corpus provides no information about reliability for other chat corpora. Some of the terms are very domain related, momentum is an example, and it would probably not make any sense to test the patterns on chats from, say, a thermodynamics domain. To provide some idea about reliability we have manually examined 15% of the chats and computed precision (0.94) and recall (0.88) for the conditional sentences and qualitative statements. Domain terms are found using an algorithmic procedure, this can only affect reliability when a domain term is used out of context (*the* **speed** *of your typing is amazing*).

A characterization of the approach we use is that within a limited domain, with a limited vocabulary, it is more or less possible to enumerate the domain terms and use syntactic patterns (Hearst, 1992; de Boer, van Someren, & Wielinga, 2007) to extract meaningful phrases related to the domain. This extraction can be done automatically.

As the result of the analysis outlined, we have for each learner a set of domain statements classified into the three types of contributions. The next question is how to "value" these contributions as they are not equal in their level of contributing to learning about the domain.

4.2.5 Computing the value of domain-related contributions

The level of domain-related contribution for each learner is computed by assigning a score to each message for the learner, summing these scores and dividing the result by the total number of messages for the learner. The result is $D_{msg}(X)$: the average domain-related contribution per message for learner X. The score computed for each message is determined by weights described in the next paragraph and the unit of analysis (Strijbos, Martens, Prins, & Jochems, 2006) for determining $D_{msg}(X)$ is a single chat message.

The weights used in the scoring function are: +2.0 for a conditional sentence, +1.0 for a qualitative statement, and +0.4 for use of a domain term. These weights have been chosen in such a way that they reflect the relative contribution to learning about the domain. The weight -0.5 is given to an agreement. This negative weight penalizes learners who agree more than their peer and can thus be seen as passive and not actively contributing to acquiring domain knowledge. In all cases, a weight is only assigned once to a single message. If a message contains two domain terms it obtains a score of +0.4 (and not +0.8). The scores for the examples above are +1.4 (speed increases) and +3.4 (if the mass becomes higher, the momentum decreases). Because there is some arbitrariness in the scoring function, we conducted a sensitivity analysis by varying all weights independently in a range of -0.5 to +0.5 from the values defined above and this produced no significant change in the overall outcome: who is A stays A and who is B stays B. This shows that the scoring function is robust.

4.3 Results

The results presented below are based on the data obtained by Saab (2005) on the collision domain as described in the Section 4.2. Saab's original process data, consisting of interactions with the simulation environment and the chats, were first integrated. The chats have been normalized by correcting spelling errors, contractions and other textual noise. Dyads for which at least one of the learners a pre-test or post-test was not available were removed, and so was a single dyad who communicated in a language other than Dutch. The process data from the 66 dyads that remained contained 17,967 chat messages. Data with respect to the chat communication, particularly the number and nature of chat messages, is displayed in Table 4.1.

Table 4.1: Averages of chat messages

	Group			
	A (n = 66)		B (n = 66)	
	Mean	SD	Mean	SD
Number of messages expressing				
Agreement	25.83	13.70	33.95	16.71
Domain terms	15.08	9.19	12.08	8.53
Qualitative statements	4.38	3.10	2.64	2.64
Conditional sentences	1.82	1.65	0.94	1.38
Total number of chat messages	138.03	58.72	134.20	56.84
Chat score per message (D_{msg})	0.10	0.10	0.02	0.08

The data regarding chat messages displayed in Table 4.1 were analyzed by means of two-sided Wilcoxon signed rank tests. With regard to agreement it was observed that B's express agreement more frequently compared to A's ($z = -4.28, p < .001$). On the other hand, compared to B's, A's mentioned more domain terms in general ($z = -2.73, p < .01$), made more conditional sentences ($z = -4.00, p < .001$), and more qualitative statements ($z = -4.26, p < .001$). The number of chat messages did not differ between A's and B's ($z = -0.97, p = .33$). By definition, because this was the basis for the grouping, the chat scores differed in favour of the A's ($z = -7.06, p < .001$).

Table 4.2: Results knowledge tests

	Group			
	A (n = 66)		B (n = 66)	
	Mean	SD	Mean	SD
Definitional knowledge				
Pre-test	6.39	2.26	5.98	2.35
Post-test	8.09	2.29	7.36	2.81
Gain from pre to post	1.70	3.06	1.38	3.36
Intuitive knowledge				
Pre-test	4.76	2.12	4.97	1.95
Post-test	7.55	2.29	6.91	2.33
Gain from pre to post	2.79	2.25	1.94	2.41

In order to gain more insight in knowledge gains within and between Group A and Group B, the data displayed in Table 4.2 were analyzed by means of paired samples T-tests. Sidak's correction for multiple comparisons was applied to control for chance capitalization.

With regard to *definitional knowledge*, comparisons were made between Group A and B with regard to their pre-test scores, post-test scores, and definitional knowledge gain. It was found that pre-test scores of Group A and B were equal ($t = 1.02, p = .31$) and so were their post-test scores ($t = 1.84, p = .07$) and observed definitional knowledge gain ($t = .59, p = .56$). The knowledge gain within each group was significant, that is in Group A the post-test scores were higher than the pre-test scores ($t = 4.51, p < .001$) and the same was true for Group B ($t = 3.33, p < .01$).

Regarding *intuitive knowledge*, the knowledge gain within Group A was significant ($t = -9.01, p < .001$) and so was the gain among B's ($t = -6.54, p < .001$). Furthermore, comparison between A's and B's showed that A's intuitive knowledge gains were greater than those of B's ($t = 2.35, p < .05$). A's and B's did not differ from each other with regard to pre-test score ($t = -0.58, p = .57$) and post-test score ($t = 1.94, p = .06$).

These results partly confirm the hypothesis that students who externalize more domain knowledge reach a higher post-test score. The hypothesis holds true for intuitive knowledge but not for definitional knowledge.

Table 4.3: Product-moment correlations between group A (rows) and B (columns)

	Group B											
	Knowledge measures						Chat communication					
Group A	1	2	3	4	5	6	7	8	9	10	11	12
1 Pre-test definitional						–.28*						
2 Pre-test intuitive												
3 Post-test definitional												
4 Post-test intuitive			.29*	.33*		.26*						
5 Gain definitional				.27*								
6 Gain intuitive				.41**		.29*						
7 Agreement							.62**	.25*			.41**	–.30*
8 Domain terms	–.27*						.36**	.48**	.33**	.48**		
9 Conditional sentences	–.26*								.32**	.29*	–.29*	.28*
10 Qualitative statements	–.27*							.34**	.45**	.50**		.31*
11 Number messages							.36**				.41**	
12 Chat score message							–.35**		.40**		–.36**	.48**

* $p < .05$ (2-tailed), ** $p < .01$ (2-tailed)

What students talk about can influence what they learn, and conversely, what they learn (or already know) can influence what they talk about. In order to investigate how A and B's knowledge, learning, and communication relate to each other, Pearson's product-moment correlations between A's and B's have been calculated with regard to knowledge measures, chat messages, and chat scores (see Table4.3).

The *top-left quadrant* of Table 4.3 displays the correlations between A's and B's knowledge measures. It can be observed that particularly A and B's intuitive knowledge gains (6) and their intuitive knowledge scores on the post-test (4) are related to each other.

The *top-right quadrant* is empty, indicating that there are no correlations between A's knowledge measures (1-6) and B's communication (7-12).

The *bottom-left quadrant* shows that there are some correlations between B's knowledge measures (1-6) and A's communication (7-12): B's prior definitional knowledge (1) is slightly and inversely related to A's frequency of expressing domain terms (8), conditional sentences (9), and qualitative statements (10).

The quadrant at the *bottom-right* of the table, displays correlations between A's and B's communication. The correlations on the main diagonal suggest that students within dyads "mirror" each other. There is not only a moderate positive correlation between the number of chat messages uttered by A's and B's (11), but also the nature of the messages (7-10) is found to correlate positively. For example, if one peer expresses agreement (7) often, the other peer is likely to do so as well ($r = .62$); if one peer frequently makes qualitative statements (10), the other peer is likely to frequently make qualitative statements too ($r = .50$), and so on. These results show that intuitive knowledge gain of A's and B's is positively related. Also, a positive and quite strong relation was established with regard to how much A's and B's talk with each other and what they talk about.

4.4 Conclusions

In this study it was found that within dyads, the students who post relatively more domain-related messages often gain more intuitive knowledge than their partners. Nonetheless, the data also showed that gains in intuitive knowledge of A's and B's are positively correlated. Furthermore, A's use domain terms, conditional sentences, and qualitative statements more frequently, in particular when their partners' prior definitional knowledge was rather weak. These partners, the B's, in turn often express more statements reflecting agreement ("yes", "ok", and so on). They seem to leave the externalization of knowledge and ideas to their partners, mostly replying by expressing agreement only. This is also called cumulative talk (e.g., Mercer, 1996). In other studies, the acquisition of intuitive knowledge has been found to be fostered by processes of drawing conclusions, interpretation and sense-making (Reid, Zhang, & Chen, 2003; Zhang, Chen, Sun, & Reid, 2004). Students actively attempting to make domain-related contributions to the communication, instead of mainly agreeing with statements of their partner, are possibly more likely to actively engage in these processes and to externalize them, which might explain their higher gains with respect to intuitive knowledge.

The correlation analysis also indicated that the lower the initial definitional knowledge of B's, the more A's posted domain-related messages, which suggest A's explained the domain to their partners. As stated in the introduction section, providing elaborate explanations is often more beneficial than receiving explanations (e.g., Webb, 1989), because students who are providing elaborate explanations are actively engaged in externalization processes. With regard to communication, it was observed that students within dyads seem to "mirror" each other: if one peer posts more domain-related messages, the other peer is also more likely to post domain-related messages, see Table 4.3. The number of messages posted by A's and B's is positively correlated, the chat scores, which give an indication of domain-relatedness of the chat, of A's and B's were also positively correlated. Moreover, the positive correlations between A's and B's also extend to the level of types of chat messages (e.g., conditional sentences, qualitative statements): if one peer posts more qualitative statements, the other peer is also more likely to post qualitative statements, and so on.

As for future research, the analysis used in this paper cannot answer the question how these different types of messages are distributed over time and how messages of A's and B's relate to each other. This analysis can shed light on how the interaction between partners in a dyad develops over time. For example, it is interesting to investigate if the number of relatively high-level contributions increases or decreases during the interaction. From this it could be inferred how long a fruitful collaborative learning session should last. If the number of high-level contributions starts to decrease, one could argue that continuing the session will, in general, not contribute much more to better learning results. Another question is if there are differences in learning outcomes for balanced and unbalanced dyads. A balanced dyad is a dyad in which the contribution of each partner is at approximately the same (higher) level, for an unbalanced dyad the number of high-level contributions of one partner substantially exceeds the contributions of the other partner. One of the assumptions behind collaborative learning is that pairing high-level contributors with low-level contributors will benefit both in terms of learning results, but maybe the low-level contributor benefits more. Analyzing the learning results of balanced and unbalanced dyads could confirm or reject this assumption.

Chapter 5

Summary and discussion

5.1 Summary

In this section we summarize our explorations in fine-grained learning analytics. As stated in the introduction, the objective of the thesis was to contribute to dynamic adaptation of inquiry learning environments. Adaptation can result from process data, the activities learners perform, it can result from an analysis of the products learners create, and it can be based on an analysis of communication and collaboration between learners. The thesis is organized around these three types of data, and each of the three main chapters addresses the analytics of one of these types of data.

Inquiry learning environments, in which students actively discover knowledge, offer an attractive approach to science education. For example, in the collision simulation environment we used, students can learn about momentum and elasticity by running virtual experiments: bouncing balls against walls and against each other (Saab, 2005). In the modelling environment we used, students learn about laws of physics, for example Ohm's law, by designing, simulating and inspecting a small electrical circuit containing two flashing lights (Mulder et al., 2011, 2012). Although inquiry learning environments are attractive, and effective when compared to other forms of instruction (e.g., Alfieri et al., 2011), they are not without challenges. Students have to learn about scientific processes, like defining a hypothesis, designing an experiment, and interpreting the results, and they have to think about the domain.

In both the collision and the electrical circuit environments students do get support. In the collision domain, students can see the effect of changing the speed of a ball for instance, and they get feedback when they answer a multiple choice question. In the electrical circuit domain, students can obtain information from a simulation and by running the model they created. The support and feedback the environments provide is pre-defined and independent of the skills of the individual learner.

In this thesis we have investigated whether it is possible to dynamically adapt the learning environment, based on the log data of the student's interaction as suggested by de Jong (2006). Logged data (log files, chats and products), are the (only) sources to base adaptation on. In Chapter 2 and 4 we used log data from the collision domain, and in Chapter 3 data from the electrical circuit domain.

We identified three steps to get from log data to adaptation (see also Figure 1.1, page 16). First, it is necessary to analyse the data to obtain an understanding of how students use a learning environment and to identify patterns of behaviour. Next, of the behavioural patterns found in the previous step, those that are of pedagogical relevance are selected, and the selected patterns are associated with the appropriate adaptation. Finally, a mechanism that monitors online learner activity, and provides the pattern-based adaptation, has to be implemented. All of these three steps, analytics, selection, and monitoring, were applied to the electrical circuit domain (Section 3.4). For this domain, two "pedagogical agents" were developed that monitored students while they created models. A model evaluation agent gave students feedback about the quality of their model. A model progression agent decided whether the model was of sufficient quality. If this was the case, the modelling environment was adapted and the student entered a richer and more difficult modelling phase. For the collision domain, the first two steps (analytics and selection) were conducted. The monitoring step was not possible for practical reasons (no experiments with the learning environment were planned), and, with regard to Chapter 2, technical reasons, as the monitoring step requires human coding.

In Chapter 2 the focus was on the analysis of the activities learners perform, and the order in which these activities are performed. Often, log data contains too much detail and an initial step to re-code learner activity is required. Once the activities are represented as a discrete sequence of (pedagogically) meaningful actions, sequence analysis can be applied. In the chapter we describe sequence analysis methods. The application of these methods to the collision domain uncovered several patterns that can be used for adaptation. Some of the sequence analysis methods used, in particular regular expression matching and entropy-related measures, have not been applied to learner data often. Given that these methods uncovered pedagogically interesting patterns, broader application is suggested. Having several different methods available in a single interactive tool makes finding patterns easier and more likely, compared to applying a single method.

In Chapter 3 the analysis of products learners create was the focus. The products we looked at are the models created in the electrical circuit domain. There was a real need for automatic analysis of the models, see the quote in Section 3.4.1 from Mulder et al. (2009). In the chapter we presented a generic approach to analyse and compare models based on specifications provided by researchers. Models can be formally represented as graph-like structures with nodes and edges. For the SCY project, the approach was slightly extended to cater for the analysis of concept maps, also graph-like structures. An additional contribution of this chapter is that pedagogical researchers, rather than programmers, can specify how the analysis of graph-like structures takes place. They can specify the terminology of the domain, the structure of the graph that is expected or desirable, and they can specify how the evaluation takes place. The latter appears to be new. Another contribution of this chapter is that a relatively simple technique, a variant on edit distance, turned out to be very important. In the electrical circuit domain, students have to think of labels for the elements of their model. We cannot expect students to spell correctly all the time, neither can we expect that they use the same terms as experts. The extensive attention given to syntactic detail has contributed to the success of the studies in which the model evaluation agent and the model progress agent were used (Mulder et al., 2011, 2012; Mulder, 2012).

In Chapter 4 communication and collaboration was the focus. In the collision domain, learners work in dyads and chat communication is used to facilitate collaboration. We describe how the chats can be analysed on domain-related content, based on an ontology of the domain and syntactic constructs students often use. When the domain of discourse is limited, which applies to the collision domain, it becomes possible to determine the "quality" of the chat messages and thus quantify the contribution of a student. The level of contribution can then be used as a basis for feedback. For the analysis in this chapter a method was needed to analyse the contribution of the individual learners in a dyad. Because learners in a dyad collaborate, the contribution of one member of a dyad influences the contribution of the other member. The method that has allowed us to analyse the individual contributions is given in Section 4.2.2 and has potential for broader applicability.

5.2 Discussion

In this section we reflect on the past and consider the future. The aim of adaptation is to contribute to making learning environments better and easier to use for students. The question arises how, in general, the design and functioning of learning environments can be improved. An example of a learning environment that is pedagogically interesting and well engineered is Betty's Brain.[1] The current version of Betty's Brain took many years and many iterations to develop, with contribu-

[1]Available via `www.teachableagents.org`.

tions from researchers with different backgrounds. Something similar is the case for the "process" of achieving adaptation. This "process" also takes time and, perhaps even more important, it requires researchers with different backgrounds working together. Adaptation has a clear technical component, developing methods and techniques to find patterns in the data. There is also a pedagogical angle, identifying patterns that are interesting, and associating patterns with feedback. What may be needed are "pedageers", a contraction of pedagogical engineers. A pedageer, as a singular noun, does not exist, but we can think of pedageering as the science of designing learning environments that result from thoughtful cooperation between researchers with a pedagogical or an engineering background.

An interesting question is whether learning environments would be designed and implemented differently when adaptation is taken into account from the start. Given that adaptation has to be based on the analysis of data, the direction is clear. Higher-level data, and data that is easier to interpret, makes analysis easier, and consequently more focussed. One way the data can be made easier to analyse is to design the learning environment in such a way that pedagogically interesting information is included in the log data. An example of such information is the expected "difficulty" of an assignment. When sequence analysis detects that a "difficult" assignment is answered immediately by a student, an adaptation of the learning environment to "really difficult" assignments could be selected. Another direction is to consider the learner also as a source, rather than just as the subject, of adaptation. For example, when the learning environment highlights words it recognises, learners might be tempted to use language more carefully and data quality would improve as a consequence. There are undoubtedly many other cases, where pedagogical mark-up of data or involving the learner, can simplify analysis and thus make adaptation more accurate.

For the collision domain we looked at a combination of types of data to some extent. In Chapter 2, the process was analysed, but some actions representing the process, the manually coded chats, belong to analysis of communication. There is also an opportunity to consider process analysis for the for the electrical circuits domain, and relate this to the product analysis of Chapter 3. We know that students used the feedback from the agents based on the quality of the models they created. We do not know how students used the feedback and an analysis of the process data may provide an explanation. Another direction for future research are finding patterns related to pedagogical concepts like "floundering" and "(un)systematic" behaviour. Generally, teachers can detect these behaviours relatively easily. They are, however, very difficult to define formally. We know that both floundering and other forms of unsystematic behaviour occurred for some students in the electrical circuits domain. Hopefully, the sequence analysis methods of Chapter 2 are also of value here.

The studies with the collision and electrical circuits environments were conducted in secondary schools in The Netherlands. The researchers had to book computer rooms in these schools, install the software, negotiate with system managers to obtain the necessary permissions, and other logistical operations. The students had to move from the classroom to the computer rooms. In other words, integrating inquiry learning in the school curriculum was nontrivial. Fortunately, this may be changing rapidly. Several secondary schools in The Netherlands give tablets to their students and for the majority of science subjects study books are available for these devices. Students do not have to move to a special computer room anymore, and the use of educational software can thus be more easily integrated in the curriculum. If inquiry learning environments are made available on these tablets, the implications may be significant, for both science education and inquiry learning. Not a few hundred, for the collision and electrical circuit environments, but thousands of students would have easy access. The social relevance for making learning environments adaptive increases likewise.

We would like to conclude this thesis with the thought that, sometimes, even teachers find it difficult to understand what learners do.

Chapter 6

Nederlandse samenvatting

Elektronische leeromgevingen leggen data over leerlingen en het leerproces vast. Interactieve leeromgevingen, zoals computersimulaties en modelleeromgevingen, registreren alle acties van de leerlingen en de producten die ze maken. Normaal gesproken wordt deze data geanalyseerd met de bedoeling om achteraf te begrijpen wat de leerlingen hebben gedaan. In dit proefschrift hebben we gekeken naar methoden die het mogelijk maken de analyse van deze data online uit te voeren, met als doel de leeromgeving aan te passen aan de capaciteiten van de individuele leerling op basis van patronen in het gevonden gedrag (de Jong, 2006).

Onderzoekend leeromgevingen maken het voor leerlingen mogelijk om zelfstandig kennis te vergaren. Onderzoekend leren is een attractieve instructiemethode voor onderwijs in natuurwetenschappelijke domeinen. Een voorbeeld hiervan is de computersimulatie over botsingen, die we hebben gebruikt in Hoofdstuk 2 en 4, waarin leerlingen bijvoorbeeld kunnen ontdekken wat impuls is. Dit doen ze door in experimenten balletjes tegen de wand en tegen elkaar te laten botsen en de resultaten van deze experimenten te interpreteren (Saab, 2005). In een modelleeromgeving, die we hebben gebruikt in Hoofdstuk 3, kunnen leerlingen bijvoorbeeld de wet van Ohm ontdekken door met een simulatie van een elektrische schakeling te experimenteren, en hier vervolgens een model van bouwen (Mulder et al., 2011, 2012). Ondanks dat onderzoekend leren effectiever is dan andere vormen van instructie (bijv., Alfieri et al., 2011), is het gebruik van onderzoekend leeromgevingen in het onderwijs niet zonder uitdagingen. Leerlingen moeten zowel het onderzoeksproces leren, waaronder het definiëren van een hypothese, het ontwerpen van een experiment en het interpreteren van de resultaten, en ze moeten nadenken over het domein.

In zowel het botsingen als het elektrische schakelingen domein krijgen leerlingen ondersteuning. Bij de botsingen kunnen leerlingen het effect zien van het veranderen van de snelheid van een bal, en ze krijgen uitleg na het invullen van een antwoord op een meerkeuzevraag. In het elektriciteits domein kunnen leerlingen de uitkomsten van een simulatie bekijken in een grafiek of tabel, en kunnen ze het gemaakte model runnen. De ondersteuning die in beide gevallen wordt gegeven is voorgedefinieerd en dus niet afhankelijk van de vaardigheden en activiteiten van een individuele leerling.

Aanpassing van de leeromgeving aan de capaciteiten van de leerling kan op verschillende manieren. Dit kan impliciet (het veranderen van de moeilijkheidsgraad van de leeromgeving), directief (het geven van specifieke instructies) en informatief. Er zijn in leeromgevingen drie typen data die kunnen bijdragen aan adaptatie. Dit is data over het proces (de activiteiten van de leerling), producten die leerlingen maken, en data die betrekking heeft op de communicatie en samenwerking tussen leerlingen. We geven eerst een beschrijving van de verschillende typen data en daarna de mogelijkheden om de leeromgeving aan te passen.

Proces. De manier waarop leerlingen een leeromgeving gebruiken wordt geregistreerd in logfiles. Logfiles bevatten normaal gesproken alle acties van een leerling, zoals het invullen van de waarde van een variabele, het starten van een simulatie, of het verbinden van twee concepten in een concept map. De analyse van deze acties kan informatie opleveren over het gedrag van leerlingen. Zijn ze bijvoorbeeld systematisch bezig en gebruiken ze alle mogelijkheden van de omgeving? In Hoofdstuk 2 beschrijven we methoden voor het analyseren van zowel korte als langere sequenties van leerlingacties. De sequentieanalyse kan patronen opleveren die voor adaptatie van de leeromgeving kunnen worden gebruikt. Bijvoorbeeld in de vorm van hints of als een indicator in een dashboard.

Producten. Producten zijn, mogelijkerwijs complexe, structuren die leerlingen maken in een leeromgeving. Voorbeelden van producten zijn modellen, concept maps, essays, tekeningen, enzovoort. De analyse van producten is vaak domein specifiek en de te gebruiken analyse methoden is afhankelijk van het type product. In het algemeen wordt de analyse van producten binnen leeromgevingen vaak gedaan aan de hand van een "expert oplossing" waarin de structuur van het product wordt gespecificeerd. Terugkoppeling naar de leerling op basis van de analyse kan door indicatoren (het aantal correcte concepten in een concept map), of directief door de leerling op een specifieke fout te wijzen. Hoofdstuk 3 beschrijft een algemene benadering voor het analyseren van en het geven van feedback op basis van producten die als een (wiskundige) graaf met knopen en verbindingen kunnen worden gerepresenteerd, zoals concept maps en modellen.

Communicatie en samenwerking. In moderne leeromgevingen speelt samenwerking een belangrijke rol. Leerlingen lossen samen problemen op en geven commentaar op het werk van elkaar. In veel omgevingen zit een chat functie waarmee leerlingen kunnen communiceren. De analyse van de chat berichten kan informatie over de onderwerpen die leerlingen bespreken, en dus aanleiding geven tot adaptatie. In Hoofdstuk 4 analyseren we de semantiek van chats in een onderzoekend leren omgeving.

Impliciete feedback. De leeromgeving wordt veranderd op basis van een analyse van het leerlinggedrag zonder dat de leerling daarvan op de hoogte wordt gesteld. Als een leerling veel fouten maakt, kunnen eenvoudiger opdrachten worden aangeboden. Of, bij een computersimulatie kan het aantal variabelen worden verminderd. Natuurlijk kan de leeromgeving ook uitdagender worden gemaakt.
Directieve feedback. Dit is het geven van instructies aan leerlingen op basis van het leerlinggedrag. Een voorbeeld is het aan de leerling vertellen dat er nog bepaalde onderdelen ontbreken in een model, of de suggestie om te gaan samenwerken met een bepaalde medeleerling. Directieve feedback zijn vaak hints die de leerling een stapje verder helpen.
Informatieve feedback. Analyse van de data kan ook worden gebruikt om de leerlingen informatie te geven. Een veel gebruikte manier is om aan de leeromgeving een "dashboard" te koppelen waarin bepaalde "leer" indicatoren dynamisch veranderen als gevolg van acties van leerlingen. Leerlingen zijn zelf verantwoordelijk voor het interpreteren van informatieve feedback.

Om van de analyse van de log data tot adaptatie te komen werden drie stappen onderscheiden, zie ook Figuur 1.1 (p. 16). Ten eerste is het nodig om de data te analyseren om een indruk te krijgen hoe leerlingen de leeromgeving gebruiken en om patronen in het gedrag te vinden. Vervolgens moeten de gevonden patronen worden geselecteerd op pedagogische relevantie en worden gekoppeld aan adequate feedback voor leerlingen. Ten slotte moet een mechanisme dat online de activiteiten van leerlingen volgt, de patronen herkend en feedback initieert, worden geïmplementeerd. Al deze drie stappen, analyse, selectie en monitoren, zijn uitgevoerd voor het elektriciteits domein. Voor dit domein werden twee "pedagogische agents" ontwikkeld die de activiteiten van leerlingen volgden terwijl ze bezig waren met het maken van een model. Een model evaluatie agent gaf leerlingen informatieve feedback over de kwalititeit van hun model in vergelijking tot een "expert model". Een model progressie agent bepaalde of het model van de leerling voldoende ontwikkeld was. In dat geval, werd de modelleeromgeving aangepast en kreeg de leerling een rijkere en moeilijkere modelleerfase. Voor het botsingen domein werden de eerste twee stappen uitgevoerd (analyse en selectie). Het monitoren van het gedrag van leerlingen was niet mogelijk vanwege praktische redenen (geen experimenten met de leeromgeving waren gepland), en met betrekking tot de resultaten van Hoofdstuk 2 technische redenen (voor een gedeelte van de monitoring is menselijk coderen noodzakelijk).

In Hoofdstuk 2 lag de nadruk op de analyse van de activiteiten die leerlingen uitvoerden, en met name de volgorde waarin die activiteiten werden uitgevoerd. Nadat de activiteiten waren gerepresenteerd als een (discrete) sequentie van pedagogisch betekenisvolle acties kon sequentieanalyse worden toegepast. In het hoofdstuk worden diverse sequentieanalyse methoden beschreven. Het toepassen van deze methoden op het botsingen domein leverde een aantal patronen op die kunnen worden gebruikt voor adaptatie. Een aantal sequentieanalyse methoden die werden gebruikt, in het bijzonder reguliere expressies en entropy gerelateerde maten, worden zelden toegepast op leerdata. Aangezien deze methoden pedagogisch interessante patronen opleverden is de suggestie om deze methoden vaker in te zetten. Het ter beschikking hebben van meerdere sequentieanalyse methoden in een interactieve omgeving maakt het vinden van patronen kansrijker, in vergelijking tot het toepassen van een enkele methode.

In Hoofdstuk 3 lag de nadruk op de analyse van producten die leerlingen maken. De producten waar we naar hebben gekeken zijn de modellen die leerlingen maakten in het elektriciteits domein. We presenteerden een algemene benadering voor het analyseren en vergelijken van modellen op basis van specificaties. Modellen kunnen formeel worden gerepresenteerd als graaf-achtige structuren met knopen en verbindingen. Deze benadering is voor het SCY project verder uitgewerkt om de analyse van concept maps mogelijk te maken. Een bijdrage van het hoofdstuk is dat onderwijskundige onderzoekers, en niet alleen programmeurs, kunnen specificeren hoe de analyse van graaf-achtige structuren die leerlingen maken moet plaatsvinden. Onderzoekers kunnen de domeinterminologie en de gewenste structuur van de grafen specificeren, alsmede hoe de evaluatie moet worden uitgevoerd. Daarnaast werd duidelijk dat een relatief simpele techniek, een variant van *edit distance*, zeer belangrijk bleek. In het elektriciteits domein moeten leerlingen zelf de elementen van het model benoemen, en we kunnen niet verwachten dat de spelling altijd correct is of dat de leerlingen dezelfde termen gebruiken als domeinexperts. De aandacht voor deze “details” heeft wezenlijk bijgedragen aan het succes van de studies waarbij de model evaluatie en de model progressie agent zijn gebruikt.

In Hoofdstuk 4 lag de nadruk op communicatie en samenwerking. In het botsingen domein werkten groepjes van twee leerlingen via een chat kanaal samen aan opdrachten. We beschrijven hoe de chats kunnen worden geanalyseerd op de mate van domein gerelateerde inhoud met behulp van een domein ontologie en syntactische constructies die leerlingen vaak gebruiken. Als het domein een relatief beperkt vocabulair heeft, zoals het botsingen domein, wordt het mogelijk om het semantische “niveau” van de chats te bepalen. Deze analyse kan vervolgens worden gebruikt voor directieve of informatieve feedback naar de leerlingen. Voor de statistische analyse in het hoofdstuk was het noodzakelijk om de bijdrage van individuele leerlingen in een groepje te bepalen. Omdat de leerlingen samenwerken, beïnvloed de inhoudelijke bijdrage van een leerling, de inhoudelijke bijdrage van de andere leerling. De methode die we hebben gebruikt in Sectie 4.2.2 is mogelijk ook voor andere samenwerkend leren onderzoek bruikbaar.

In de rest van deze samenvatting blikken we terug en kijken we vooruit. De doelstelling van adaptatie is om een bijdrage te leveren aan de praktische bruikbaarheid van leeromgevingen. De vraag doet zich voor hoe, in het algemeen, het ontwerp en het functioneren van leeromgevingen kan worden verbeterd. Een voorbeeld van een leeromgeving die onderwijskundig interessant en praktisch bruikbaar is is *Betty's Brain*.[1] De huidige versie van Betty's Brain is het resultaat van jarenlange ontwikkeling, met bijdragen van onderzoekers met diverse achtergronden. Het ontwikkelen van adaptiviteit vraagt ook om samenwerking tussen onderzoekers uit verschillende disciplines. Adaptatie heeft een duidelijke technische component, het ontwikkelen van methoden en technieken om patronen te vinden in de data. Voor de identificatie van interessante patronen waarop feedback kan worden gebaseerd is echter ook pedagogische expertise nodig. Wellicht hebben we "pedageurs", pedagogische ingenieurs, nodig. Een pedageur bestaat niet, maar we kunnen wel denken over pedageuren als de wetenschap van het ontwerpen en ontwikkelen van leeromgevingen door de eendrachtige samenwerking tussen onderzoekers met een pedagogische en een technische achtergrond.

Een interessante vraag is of leeromgevingen anders zouden kunnen worden ontworpen met adaptatie in gedachten. Gegeven dat adaptatie gebaseerd moet zijn op de analyse van leergedragdata is de zoekrichting bepaald. Als de data eenvoudiger geïnterpreteerd kan worden, wordt de analyse eenvoudiger. Een manier om dat te bereiken is het toevoegen van pedagogisch relevante informatie aan de data. Een voorbeeld van dergelijke informatie is het aangeven van de moeilijkheidsgraad van een opdracht als onderdeel van de data. Is de opdracht moeilijk en de leerling heeft het antwoord snel gevonden, dan kan worden overwogen om de leeromgeving voor deze leerling nog moeilijker te maken. Een andere aanpak om de data eenvoudiger te kunnen analyseren is het inzetten van de leerling. Een voorbeeld is om in chats aan te geven welke woorden worden herkend, zodat spelfouten sneller worden gezien door de leerling. Er zijn ongetwijfeld nog veel meer gevallen waarbij pedagogische informatie en de leerlingen zelf, de analyse kan vereenvoudigen.

In het botsingen domein analyseerden we een combinatie van typen data. In Hoofdstuk 2 werd het proces geanalyseerd, maar sommige acties in het proces, de gecodeerde chats, waren afkomstig van communicatie. Er zijn meer mogelijkheden om

[1] `www.teachableagents.org`.

niet een enkel datatype te analyseren, maar meerdere typen tegelijkertijd. De analyse van het proces kan, bijvoorbeeld, ook worden toegepast op het elektriciteits domein, gerelateerd aan de productanalyse. We weten dat leerlingen de feedback van de evaluatie agent hebben gebruikt. We weten niet precies hoe leerlingen de feedback hebben gebruikt en een analyse van de acties na feedback kan hierover uitsluitsel geven. Een andere richting voor verder onderzoek is het vinden van pedagogisch gefundeerde concepten zoals *floundering* en andere vormen van niet systematisch gedrag. Docenten kunnen dit gemakkelijk herkennen, maar het is niet eenvoudig om dit soort gedrag formeel te definiëren. We weten dat floundering voorkomt in het elektriciteits domein en kunnen de sequentieanalyse technieken in Hoofdstuk 2 wellicht gebruiken om dit gedrag te identificeren.

De studies met het botsingen en het elektriciteits domein vonden plaats in het voortgezet onderwijs. De onderzoekers moesten onder andere computerlokalen reserveren op de scholen, de software installeren, en overleggen met systeem beheerders om de noodzakelijke permissies te krijgen. Leerlingen moesten van de klas naar het computerlokaal om aan de studies te kunnen deelnemen. Met andere woorden, het integreren van computer ondersteund onderzoekend leren in het curriculum is niet eenvoudig. Misschien dat dit snel kan veranderen. Diverse scholen hebben wifi netwerken aangelegd en tablets aan leerlingen gegeven. Voor alle vakken in het voortgezet onderwijs zijn lesboeken beschikbaar die op de tablets kunnen worden gelezen. De implicaties van deze veranderingen zouden wel eens groot kunnen zijn, zowel voor het natuurwetenschappelijk onderwijs als voor onderzoekend leren. Aan de botsingen en elektriciteits studies hebben een paar honderd leerlingen meegedaan, maar als de leeromgevingen op tablets in de klas zouden draaien dan hebben duizenden leerlingen toegang. Deze ontwikkeling zal de maatschappelijke relevantie van het adaptief maken van leeromgevingen verhogen.

We sluiten dit proefschrift af met de observatie dat, soms, zelfs ervaren docenten het moeilijk vinden om te begrijpen wat leerlingen doen.

Bibliography

Acton, W. H., P. J. Johnson, & T. E. Goldsmith (1994). Structural knowledge assessment: comparison of referent structures. *Journal of Educational Psychology 86*, 303–311.

Ainsworth, S. & I. Iacovides (2005). Learning by constructing self-explanation diagrams. In *Paper presented at the EARLI*.

Alfieri, L., P. J. Brooks, N. J. Aldrich, & H. R. Tenenbaum (2011). Does discovery-based instruction enhance learning? *Journal of Educational Psychology 103*, 1–18.

Ali, L., M. Hatala, D. Gašević, & J. Jovanić (2012). A qualitative evaluation of evolution of a learning analytics tool. *Computers & Education 58*, 470–489.

Anjewierden, A. (2006). tOKo and Sigmund: text analysis support for ontology development and social research. http://www.toko-sigmund.org.

Anjewierden, A. & H. Gijlers (2008). An exploration of tool support for categorical coding. In P. Kirschner, F. Prins, V. Jonker, & G. Kanselaar (Eds.), *International Conference for the Learning Sciences (ICLS 2008)*, Utrecht, The Netherlands, pp. 35–44.

Anjewierden, A., B. Kollöffel, & C. Hulshof (2007, September). Towards educational data mining: Using data mining methods for automated chat analysis to understand and support inquiry learning processes. In C. Romero, M. Pechenizkiy, T. Calders, & S. R. Viola (Eds.), *International Workshop on Applying Data Mining in e-Learning (ADML 2007)*, Crete, Greece, pp. 27–36.

Baker, R. (2010). Data mining for education. In P. Peterson, E. Baker, & B. McGaw (Eds.), *International Encyclopedia of Education* (3rd ed.)., pp. 112–118. Oxford: Elsevier.

Baker, R., T. Barnes, & J. E. Beck (Eds.) (2008, June). *1st International Conference on Educational Data Mining (EDM 2008)*, Montréal, Canada.

Baker, R. & K. Yacef (2009). The state of Eductional Data Mining in 2009: A review and future visions. *Journal of Educational Data Mining 1*, 3–17.

Biswas, G., D. Schwartz, K. Leelawong, & N. Vye (2005). Learning by teaching: a new agent paradigm for educational software. *Applied Artificial Intelligence 19*, 363–392.

Bravo, C., W. R. van Joolingen, & T. de Jong (2006). Modeling and simulation in inquiry learning: checking solutions and giving intelligent advice. *Simulation 82*, 769–784.

Bravo, C., W. R. van Joolingen, & T. de Jong (2009). Using Co-Lab to build system dynamics models: Students' actions and online tutorial advice. *Computers & Education 53*, 243–251.

Campbell, J. P. (2007). *Utilizing Student Data within the Course Management System to Determine Undergraduate Student Academic Success: An Exploratory Study*. Ph. D. thesis, Purdue University.

Chen, Z. & D. Klahr (1999). All other things being equal: Acquisition and transfer of the control of variables strategy. *Child Development 70*, 1098–1120.

Christoph, N., A. Anjewierden, J. Sandberg, & B. J. Wielinga (2003). Measuring learning behaviour in a collaborative gaming-simulation learning environment for the domain of knowledge management. In *ED-Media (Educational media, hypermedia and telecommunications, Learning and Objects Symposium)*, Honolulu, Hawaii, USA.

Cline, B. E., C. C. Brewster, & R. D. Fell (2010). A rule-based system for automatically evaluating student concept maps. *Expert Systems with Applications 37*, 2282–2291.

Cohen, E. G. (1994). Restructuring the classroom: Conditions for productive small groups. *Review of Educational Research 64*, 1–35.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement 20*, 37–46.

Cox, R. (1999). Representation construction, externalised cognition and individual differences. *Learning and Instruction 9*, 343–363.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM 7*, 171–176.

de Boer, V., M. van Someren, & B. J. Wielinga (2007). A redundancy-based method for the extraction of relation instances from the web. *International Journal of Human-Computer Studies 65*, 816–831.

de Boer, V., M. W. van Someren, B. J. Wielinga, & A. Anjewierden (2010, October). Exploiting redundancy for pattern-based relation instantiation using tOKo. In S. Pinto & P. Cimiano (Eds.), *Knowledge Engineering and Knowledge Management by the Masses (EKAW 2010)*, Lisbon, Portugal, pp. 11–15.

de Jong, T. (2006). Technological advances in inquiry learning. *Science 321*, 532–533.

de Jong, T. & W. R. van Joolingen (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research 68*, 179–202.

de Jong, T. & W. R. van Joolingen (2007). Model-facilitated learning. In J. M. Spector, M. D. Merrill, J. J. G. van Merriënboer, & M. P. Driscoll (Eds.), *Handbook of research on educational communication and technology* (3rd ed.)., pp. 457–468. Lawrence Erlbaum.

de Jong, T., A. Weinberger, I. Girault, A. Kluge, A. W. Lazonder, M. Pedaste, S. Ludvigsen, M. Ney, B. Wasson, A. Wichmann, C. Geraedts, A. Giemza, A. Hovardas, R. Julien, W. R. van Joolingen, A. Lejeune, C. Manoli, Y. Matteman, T. Sarapuu, A. Verkade, V. Vold, & Z. C. Zacharia (2012). Using scenarios to design complex technology-enhanced learning environments. *Educational Technology Research and Development*. In press.

de Jong et al., T. (2010). Learning by creating and exchanging objects: the SCY experience. *British Journal of Educational Technology 41*, 909–921.

de Moor, A. & A. Anjewierden (2007). A socio-technical approach for topic community member selection. In C. Steinfield, B. T. Pentland, M. Ackerman, & N. Contractor (Eds.), *Communitities and Technologies 2007*, pp. 225–244. Springer.

de Wever, B., T. Schellens, M. Valcke, & H. van Keer (2006). Content analysis schemes to analyze transcripts of online asynchronous discussions groups: A review. *Computers & Education 46*, 6–28.

D'Mello, S., A. Olney, & N. Person (2010). Mining collaborative patterns in tutorial dialogues. *Journal of Educational Data Mining 2*, 1–37.

D'Mello, S., R. S. Taylor, & A. Graesser (2007). Monitoring affective trajectories during complex learning. In D. S. McNamara & J. G. Trafton (Eds.), *Proceedings of the 29th Annual Cognitive Science Society*, Austin, TX, pp. 203–208. Cognitive Science Society.

Duval, E. (2011). Attention please! learning analytics for visualization and recommendation. In *1st International Conference on Learning Analytics and Knowledge (LAK 2011)*, Banff, Canada, pp. 9–17. ACM.

Dyke, G., K. Lund, & J. Girardot (2009, June). Tatiana: an environment to support the CSCL analysis process. In C. O'Malley, D. Suthers, P. Reimann, & A. Dimitracopoulou (Eds.), *Computer Supported Collaborative Learning Practices – Proceedings of the 9th international conference on CSCL - Volume 1*, Rhodes, Greece.

Edelson, D. C., D. N. Gordin, & R. D. Pea (1999). Addressing the challenges of inquiry-based learning through technology and curriculum design. *Journal of the Learning Sciences 8*, 391–450.

Erkens, G. & J. Janssen (2008). Automatic coding of dialogue acts in collaboration protocols. *International Journal of Computer-Supported Collaborative Learning 3*, 447–470.

Eysink, T. H. S., T. de Jong, K. Berthold, B. Kollöffel, M. Opferman, & P. Wouters (2009). Learner performance in multimedia learning arrangements: An analysis across instructional approaches. *American Educational Research Journal 46*, 1107–1149.

Fischer, F. & H. Mandl (2005). Knowledge convergence in computer-supported collaborative learning - the role of external representation tools. *Journal of the Learning Sciences 14*, 405–441.

Forrester, J. (1961). *Industrial dynamics*. Waltham, MA: Pegasus Communications.

Gelernter, D. (1985). Generative communication in Linda. *ACM Transactions on Programming Languages and Systems 7*, 80–112.

Gijlers, H. & T. de Jong (2009). Sharing and confronting propositions in collaborative inquiry learning. *Cognition and Instruction 27*, 239–268.

Gijlers, H., A. M. van Dijk, & A. Weinberger (2011). How can scripts and awareness tools orchestrate individual and collaborative drawing of elementary students for learning science? In *Paper presented at CSCL 2011*, Hong Kong.

Hagemans, M. G., H. van der Meij, & T. de Jong (2012). The effects of a concept map-based support tool on simulation-based inquiry learning. Submitted.

Harris, R. L. (1999). *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pp. 529–545.

Hendrikse, H. P. (2008, May). *Wiskundig actief: Het ondersteunen van onderzoekend leren in het wiskunde onderwijs*. Ph. D. thesis, University of Twente.

Hmelo-Silver, C. E. (2003). Analyzing collaborative knowledge construction: multiple methods for integrated understanding. *Computers & Education 41*, 397–420.

Hoppe, H. U., J. Engler, & S. Weinbrenner (2012). The impact of structural characteristics of concept maps on automatic quality measurement. To appear in Proceedings of the International Conference for the Learning Sciences.

Hoppe, H. U. & K. Gassner (2002). Integrating collaborative concept mapping tools with group memory and retrieval functions. In G. Stahl (Ed.), *Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL 2002)*, Boulder, USA, pp. 716–725. Lawrence Erlbaum Associates.

Hübscher, R. & S. Puntambekar (2008, June). Integrating knowledge gained from data mining with pedagogical knowledge. In *1st International Conference on Educational Data Mining (EDM 2008)*, Montréal, Canada, pp. 97–106.

Hulshof, C. D. (2004). Log file analysis. In *Encyclopedia of Social Measurement*, Volume 2, Manchester, UK, pp. 577–583. Elsevier.

Ifenthaler, D. (2010). Relational, structural, and semantic analysis of graphical representations and concept maps. *Educational Technology Research and Development 58*, 81–97.

Ifenthaler, D., I. Masduki, & N. M. Seel (2011). The mystery of cognitive structure and how we can detect it: tracking the development of cognitive structure over time. *Instructional Science 39*, 41–61.

Janssen, J. (2008, March). *Using visualizations to support collaboration and coordination during computer-supported collaborative learning*. Ph. D. thesis, University of Utrecht.

Kanungo, T. (1999). UMDHMM: Hidden Markov model toolkit. In A. Kornai (Ed.), *Extended Finite State Models of Language*. Cambridge University Press.

Kardan, S. & C. Conati (2011, July). A framework for capturing distinguishing user interaction behaviours in novel interfaces. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, & J. Stamper (Eds.), *EDM 2011: 4th International Conference on Educational Data Mining*, Eindhoven, The Netherlands, pp. 159–168.

Kay, J., N. Masionneuve, K. Yacef, & P. Reimann (2006a). The big five and visualisations of team work activity. In *Proceedings Intelligent Tutoring Systems (ITS 2006)*, LNCS 4053, Springer, pp. 197–206.

Kay, J., N. Masionneuve, K. Yacef, & P. Reimann (2006b, January). Wattle tree: What'll it tell us? Technical report, School of Information Technologies, University of Sydney.

Kay, J., N. Masionneuve, K. Yacef, & O. Zaïane (2006, July). Mining patterns of events in students' teamwork data. In *Proceedings of the Workshop on Educational Data Mining (ITS 2006)*, Jhongli, Taiwan, pp. 45–52.

Kinchin, I. M. & D. B. Hay (2000). How a qualitative approach to concept map analysis can be used to aid learning by illustrating patterns of conceptual development. *Educational Research 42*, 43–57.

Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. In C. E. Shannon & J. McCarthy (Eds.), *Automata Studies*, pp. 3–42. Princeton University Press.

Köck, M. & A. Paramythis (2010, November). Towards adaptive learning support on the basis of behavioural patterns in learning activity sequences. In F. Xhafa, A. Pombortsis, V. Dagdilelis, S. Demetriadis, & N. Besis (Eds.), *Proceedings of the Second International Conference on Intelligent Networking and Collaborative Systems*, Thessaloniki, Greece, pp. 100–107.

Koedinger, K., K. Cunningham, A. Skogsholm, & B. Leber (2008, June). An open repository and analysis tools for fine-grained, longitudinal learner data. In R. Baker, T. Barnes, & J. E. Beck (Eds.), *1st International Conference on Educational Data Mining (EDM 2008)*, Montréal, Canada, pp. 157–166.

Kollöffel, B. (2008, December). *Getting the picture: The role of external representations in simulation-based inquiry learning*. Ph. D. thesis, University of Twente.

Kuhn, D., J. Black, A. Keselman, & D. Kaplan (2000). The development of cognitive skills to support inquiry learning. *Cognition and Instruction 9*, 285–327.

Kullback, S. & R. A. Leibler (1951). On information and sufficiency. *Annals of Mathematical Statistics 22*, 79–86.

Leelawong, K. (2005, August). *Using the learning-by-teaching paradigm to design intelligent learning environments*. Ph. D. thesis, Vanderbilt University, Nashville, USA.

Leelawong, K. & G. Biswas (2008). Designing learning by teaching agents: The Betty's Brain system. *International Journal of Artificial Intelligence in Education 18*, 181–208.

Leemkuil, H., T. de Jong, R. de Hoog, & N. Christoph (2003). KM Quest: a collaborative internet-based simulation game. *Simulation & Gaming 31*, 89–111.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Dokla 10*, 707–710.

Löhner, S. (2005). *Computer based modeling tasks: the role of external representation*. Ph. D. thesis, University of Amsterdam.

Long, P., G. Siemens, G. Conole, & D. Gašević (Eds.) (2011). *1st International Conference on Learning Analytics and Knowledge (LAK 2011)*, Banff, Canada. ACM.

Lou, Y., P. C. Abrami, & S. d'Apollonia (2001). Small group and individual learning with technology: A meta-analysis. *Review of Educational Research 71*, 449–521.

Manning, C. D. & H. Schütze (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press.

Marshall, B., H. Chen, & T. Madhusudan (2006). Matching knowledge elements in concept maps using a similarity flooding algorithm. *Decision Support Systems 42*, 1290–1306.

Martinez, A., A. Harrer, & B. Barros (2005, November). Library of interaction tools. Kaleidoscope Deliverable 31.2.

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? the case for guided methods of instruction. *American Psychologist 59*, 14–19.

Melnik, S., H. Garcia-Molina, & E. Rahm (2001). Similarity flooding: a versatile graph matching algorithm. Technical report, Stanford University.

Melnik, S., H. Garcia-Molina, & E. Rahm (2002). Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In G. Siemens & P. Long (Eds.), *Proceedings of the 18th International Conference on Data Engineering (ICDE 2002)*, San Jose, CA, pp. 117–128. IEEE Computer Society.

Mercer, N. (1996). The quality of talk in children's collaborative activity in the classroom. *Learning and Instruction 6*, 359–377.

Mostow, J. (2004, August). Some useful design tactics for mining ITS data. In *Workshop on Analyzing Student-Tutor interaction logs to improve educational outcomes (ITS 2004)*, Maceiò, Alagoas, Brazil, pp. 20–28.

Mulder, Y. G. (2012). *Learning science by creating models*. Ph. D. thesis, University of Twente.

Mulder, Y. G., A. W. Lazonder, & T. de Jong (2009). Finding out how they find it out: An empirical analysis of inquiry learners' need for support. *International Journal of Science Education 32*, 2033–2053.

Mulder, Y. G., A. W. Lazonder, & T. de Jong (2011). Comparing two types of model progression in an inquiry learning environment with modelling facilities. *Learning and Instruction 21*, 614–624.

Mulder, Y. G., A. W. Lazonder, T. de Jong, A. Anjewierden, & L. Bollen (2012). Validating and optimizing the effects of model progression in simulation-based inquiry learning. *Journal of Science Education and Technology In Press*.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys 33*, 31–88.

Nesbit, J. C. & K. Nakayama (1990). Response markup with an edit distance algorithm: A technique for providing learners with feedback on misspellings. *Computers & Education 14*(3), 271–279.

Novak, J. & D. B. Gowin (1984). *Learning how to learn*. Cambridge, UK: Cambridge University Press.

Pechenizkiy, M., T. Calders, C. Conati, S. Ventura, C. Romero, & J. Stamper (Eds.) (2011, July). *EDM 2011: 4th International Conference on Educational Data Mining*, Eindhoven, The Netherlands.

Perera, D., J. Kay, I. Koprinska, K. Yacef, & O. Zaïane (2009). Clustering and sequential pattern mining to support team learning. *IEEE Transactions on Knowledge and Data Engineering 21*, 759–772.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program 14*, 130–137.

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 257–286.

Reid, D. J., J. Zhang, & Q. Chen (2003). Supporting scientific discovery learning in a simulation environment. *Journal of Computer Assisted Learning 19*, 9–20.

Romero, C. & S. Ventura (2007). Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications 33*, 135–146.

Romero, C., S. Ventura, P. G. Espejo, & C. Hervas (2008, June). Mining algorithms to classify students. In R. Baker, T. Barnes, & J. E. Beck (Eds.), *1st International Conference on Educational Data Mining (EDM 2008)*, Montréal, Canada, pp. 8–17.

Romero, C., S. Ventura, N. Pechenizkiy, & R. Baker (2011). *Handbook of Educational Data Mining*. Boca Raton: CRC Press.

Rosé, C. P., Y.-C. Cui, J. Arguello, A. Weinberger, & K. Stegmann (2008). Analyzing collaborative learning processes automatically: Exploiting the advances of computational linguistics in computer-supported collaborative learning. *International Journal of Computer-Supported Collaborative Learning 3*, 327–371.

Ruiz-Primo, M. A. & R. J. Shavelson (1996). Problems and issues in the use of concept maps in science assessment. *Journal of Research in Science Teaching 33*, 569–600.

Ruiz-Primo, M. A., R. J. Shavelson, & S. E. Schultz (1997, July). On the validity of concept map-base assessment interpretations: An experiment testing the assumption of hierarchical concept maps in science. CSE Technical Report 455, Stanford University.

Saab, N. (2005). *Chat and Explore: The role of support and motivation in collaborative scientific discovery learning*. Ph. D. thesis, University of Amsterdam.

Sabelli, N. H. (2005). Complexity, technology, science, and education. *Journal of the Learning Sciences 15*, 5–9.

Schreiber, G., H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Vandevelde, & B. Wielinga (1999). *Knowledge Engineering and Management: The CommonKADS Methodology*. London: The MIT Press.

Schvaneveldt, R. W. (1990). *Pathfinder Associative Networks: Studies in Knowledge Organization*. Norwood, NJ: Ablex.

Scott, J. (1991). *Social network analysis*. London: Sage.

Shanabrook, D. H., D. G. Cooper, B. P. Woolf, & I. Arroyo (2010, July). Identiying high-level student behavior using sequence-based motif discovery. In R. S. J. d. Baker, A. Merceron, & P. I. P. jr (Eds.), *Proceedings of the 3rd International Conference on Educational Data Mining*, Pittsburgh, PA, pp. 191–200.

Shannon, C. E. & W. Weaver (1949). *The mathematical theory of communication*. Urbana, IL: University of Illinois Press.

Shavelson, R. J., H. Lang, & B. Lewin (1994, August). On concept maps as potential 'authentic" assessments in science. CSE Technical Report 388, CRESST, University of Santa Barbara.

Siemens, G. (2011). Learning and knowledge analytics. www.learninganalytics.net.

Slavin, R. E. (1994). Student teams-achievement divisions. In S. Sharan (Ed.), *Handbook of cooperative learning methods*, pp. 3–19. Westport: Greenwood press.

Srikant, R. & R. Agrawal (1996). Mining sequential patterns: generalizations and performance improvements. In M. Wolpers, P. Kirscher, M. Scheffel, S. Lindstaedt, & V. Dimitrova (Eds.), *Proceedings 5th International Conference on Extending Database Technology: Advances in Database Technology*. Springer Berlin / Heidelberg.

Strijbos, J. W., R. L. Martens, F. J. Prins, & W. M. G. Jochems (2006). Content analysis: What are they talking about? *Computers & Education 46*, 29–48.

Swaak, J. & T. de Jong (2001). Discovery simulations and the assessment of intuitive knowledge. *Journal of Computer Assisted Learning 17*, 284–295.

Teasley, S. D. (1995). The role of talk in children's peer collaborations. *Developmental Psychology 31*, 207–220.

Thompson, K. (1968). Regular expression search algorithm. *Communications of the ACM 11*, 419–422.

van Borkulo, S. P. (2009, June). *The assessment of learning outcomes of computer modeling in secondary science education*. Ph. D. thesis, University of Twente.

van Boxtel, C., J. van der Linden, & G. Kanselaar (2000). Collaborative learning tasks and the elaboration of conceptual knowledge. *Learning and Instruction 10*, 311–330.

van der Linden, J. L., G. Erkens, H. Schmidt, & P. Renshaw (2000). Collaborative learning. In P. R. J. Simons, J. L. van der Linden, & T. Duffy (Eds.), *New Learning*, pp. 33–48. Dordrecht: Kluwer.

van Drie, J., C. van Boxtel, J. Jaspers, & G. Kanselaar (2005). Effects of representational guidance on domain specific reasoning in CSCL. *Computers in Human Behavior 21*, 575–602.

van Joolingen, W. R. & T. de Jong (2003). Simquest: Authoring educational simulations. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring tools for advanced technology educational software: Toward cost-effective production of adaptive, interactive, and intelligent educational software*, pp. 1–31. Dordrecht: Kluwer Academic Publishers.

van Joolingen, W. R., T. de Jong, A. W. Lazonder, E. Savelsbergh, & S. Manlove (2005). Co-Lab: Research and development of an on-line learning environment for collaborative scientific discovery learning. *Computers in Human Behaviour 21*, 671–688.

Webb, N. M. (1989). Peer interaction and learning in small groups. *International Journal of Educational Research 13*, 21–39.

Webb, N. M., S. H. Farivar, & A. M. Mastergeorge (2002). Productive helping in cooperative groups. *Theory into Practice 41*, 13–20.

Webb, N. M., K. M. Nemer, A. W. Chizhik, & B. Sugrue (1998). Equity issues in collaborative group assessment: group composition and performance. *American Educational Research Journal 35*, 607–651.

Weinberger, A. & F. Fischer (2006). A framework to analyze argumentative knowledge construction in computer-supported collaborative learning. *Computers & Education 46*, 71–95.

Weinbrenner, S. & A. Anjewierden (2011). SQLSpaces interface for Prolog. http://sqlspaces.collide.info/prolog.

Weinbrenner, S., A. Giemza, & H. U. Hoppe (2007, July). Engineering heterogeneous distributed learning environments using tuple spaces as an architectural platform. In J. M. Spector, D. G. Sampson, T. Okamoto, Kinshuk, S. A. Cerri, M. Ueno, & A. Kashihara (Eds.), *Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies (ICALT 2007)*, Niigata, Japan, pp. 434–436.

Wielemaker, J. (2003, December). An overview of the SWI-Prolog programming environment. In F. Mesnard & A. Serebenik (Eds.), *Proceedings of the 13th International Workshop on Logic Programming Environments*, Heverlee, Belgium, pp. 1–16. Katholieke Universiteit Leuven.

Zhang, J. W., Q. Chen, Y. Q. Sun, & D. J. Reid (2004). Triple scheme of learning support design for scientific discovery learning based on computer simulation: experimental research. *Journal of Computer Assisted Learning 20*, 269–282.

Appendix A

Guidelines and specifications

A.1 Guidelines for specifying a term set

This appendix provides guidelines for researchers who want to specify a term set.

Introduction. A term set contains a specification of which labels students can use in graph-like structures such as concept maps or system dynamics models. Below, some guidelines are provided on how to construct a term set. The terms in a term set are referred to in a reference model. It is recommended to use English names for terms (e.g., concept names and names of variables) in both the term set and the reference model.

Example. The simplest term specification is the following:

```
<term name="water">       % Term being specified: water
  <string>water</string>  % String students use for the term
</term>
```

Foreign languages. This defines the term "water" for English students. When students use the string "water" the term is matched. If the students are from Estonia, the term specification might be as follows.

```
<term name="water">       % Term being specified: water
  <string>vesi</string>   % String Estonian students use
</term>
```

Synonyms, alternate spellings. In many cases terms can have synonyms, can be abbreviated or have other terminological variations. In such cases additional strings can be added to the term, the order in which the strings appear is not important.

```
<term name="water">
```

```
  <string>water</string>
  <string>h2o</string>
</term>
```

Multi-word terms. If a term consists of multiple words, then separate these words by a space. This informs the algorithm that the order of the words can be different than specified. For example, students might write either "biomass growth" or "growth of biomass".

```
<term name="biomass growth">
  <string>biomass growth</string> % Matches: growth of biomass
</term>
```

In some languages, for example Dutch and German, the rule for compound noun-noun terms is to contract them by removing the space. In the term set it is better to keep the space such that other word orders also match. An example in English is the term "workplace". If this is also the string then "place of work" will not match. So, it is better to insert the space.

```
<term name="workplace">
  <string>work place</string>      % Space between work and place
</term>
```

Descriptive labels, sub-strings. Students sometimes use descriptive labels like "a lot of energy". If such a label is allowed it can be matched by specifying the sub-string "energy":

```
<term name="energy">
  <sub_string>energy</sub_string>  % Matches labels containing energy
</term>
```

The algorithm uses sub-string matching only when all else fails. The sub-string construct is therefore a relative safe mechanism to catch descriptive labels.

Short terms. For very short terms, by default less than five characters, the algorithm expects an exact match. For example, the label "cats" does not match the string "cat". The reason for demanding an exact match for very short words is that there are many other words that look very similar, for instance "cat", "hat", "chat" and so forth.

```
<term name="cat">
  <string>cat</string>
  <string>cats</string>
</term>
```

Plurals and other inflected forms. The algorithm is language independent and does not have any rules for plurals or other inflections. In most languages inflections look very similar to the base word and "spelling correction" will suffice. If a term and one of its inflections must match different terms then specify them separately. The matching algorithm prefers the best match.

```
<term name="consumer">
  <string>consumer</string>
</term>

<term name="consumers">
  <string>consumers</string>
</term>
```

Upper case and special characters. The default is to ignore case and remove all non-alphanumeric characters. If an exact match is required add the exact is true attribute to a string.

```
<term name="hash symbol">
  <string>hash</string>
  <string exact="true">#</string> % Hash (#) symbol
</term>
```

A.2 Specifications for the system dynamics study

The specifications for the system dynamics study in Section 3.4 are given below.

A.2.1 Term set

```
<term_set name="mulder_2009" word_order="any">

 <term name="lading">
  <term>lading</term>
  <term type="stock">condensator</term>
  <term>doosje</term>
  <term>condensator lading</term>
 </term>

 <term name="spanning_over_de_weerstanden">
  <term type="auxiliary">spanning weerstand</term>
  <term type="auxiliary">spanning over de weerstanden</term>
 </term>

 <term name="spanning_over_de_condensator">
  <term type="auxiliary">spanning over de condensator</term>
  <term type="auxiliary">spanning circuit</term>
  <term type="auxiliary">spanning</term>
  <term type="auxiliary">condensator spanning</term>
  <term type="auxiliary">delta spanning</term>
  <term type="auxiliary">totaal volt</term>
  <term type="auxiliary">voltage</term>
  <term type="auxiliary">spanning condensator</term>
 </term>

 <term name="stroomsterkte">
  <term>stroom</term>
  <term>stroom sterkte</term>
  <term>stroom na weerstanden</term>
  <term>stroom over de condensator</term>
  <term>instroom</term>
  <term>stroom in</term>
  <term>toename</term>
  <term>invoer</term>
  <term>input</term>
  <term>opladings snelheid</term>
  <term>oplaad snelheid</term>
  <term>lading per seconde</term>
 </term>

 <term name="vervangingsweerstand">
  <term>vervangings weerstand</term>
  <term>weerstand vervanging</term>
  <term>parallel weerstand</term>
  <term>weerstand samen</term>
  <term>weerstand totaal</term>
  <term>vervanging lamp</term>
  <term>lamp totaal</term>
 </term>

 <term name="lampje_1">
  <term>weerstand</term>
  <term>weerstand 1</term>
```

```
  <term>variabele weerstand</term>
  <term>lamp</term>
  <term>lampje</term>
  <term>lamp 1</term>
  <term>lampje 1</term>
  <term>weerstand boven</term>
  <term>weerstand links</term>
  <term>lampje boven</term>
  <term>lampje links</term>
  <term>licht 1</term>
 </term>

 <term name="lampje_2">
  <term>weerstand 2</term>
  <term>lamp 2</term>
  <term>lampje 2</term>
  <term>weerstand onder</term>
  <term>weerstand rechts</term>
  <term>lampje onder</term>
  <term>lampje rechts</term>
  <term>licht 2</term>
 </term>

 <term name="bronspanning">
  <term>bron</term>
  <term>bron spanning</term>
  <term type="constant">spanning</term>
  <term>batterij</term>
  <term>plus batterij</term>
  <term>min batterij</term>
  <term>power bron</term>
  <term>stroom bron</term>
 </term>

 <term name="condensator_capaciteit">
  <term>condensator capaciteit</term>
  <term>capaciteit</term>
  <term type="constant">condensator</term>
  <term type="auxiliary">condensator</term>
 </term>
</term_set>
```

A.2.2 Reference model

```
<reference_model name="mulder_2009" notation="sdm" term_set="mulder_2009">

 <node term="lading" type="stock"/>
 <node term="stroomsterkte" type="auxiliary"
```

```
      formula="spanning_over_de_weerstanden / vervangingsweerstand"/>
<node term="spanning_over_de_condensator"
      type="auxiliary"
      formula="lading * condensator_capaciteit"/>
<node term="spanning_over_de_weerstanden"
      type="auxiliary"
      formula="bronspanning - spanning_over_de_condensator"/>
<node term="vervangingsweerstand"
      type="auxiliary"
      formula="1 / ((1/lampje_1) + (1/lampje_2))"/>

<node term="condensator_capaciteit" type="auxiliary; constant"/>
<node term="lampje_1" type="auxiliary; constant"/>
<node term="lampje_2" type="auxiliary; constant"/>
<node term="bronspanning" type="auxiliary; constant"/>
<node term="inflow" type="flow"/>

<edge tail="lading" head="spanning_over_de_condensator"
      qualitative="linear"/>
<edge tail="condensator_capaciteit" head="spanning_over_de_condensator"
      qualitative="inverse"/>
<edge tail="spanning_over_de_condensator" head="spanning_over_de_weerstanden"
      qualitative="inverse"/>
<edge tail="bronspanning" head="spanning_over_de_weerstanden"
      qualitative="linear"/>
<edge tail="spanning_over_de_weerstanden" head="stroomsterkte"
      qualitative="linear"/>
<edge tail="vervangingsweerstand" head="stroomsterkte"
      qualitative="inverse"/>
<edge tail="lampje_1" head="vervangingsweerstand"
      qualitative="asymptotical"/>
<edge tail="lampje_2" head="vervangingsweerstand"
      qualitative="asymptotical"/>

<edge tail="stroomsterkte" head="inflow" qualitative="unspecified"/>
<edge tail="inflow" head="lading" type="flow"/>

<t_edge tail="vervangingsweerstand" head="inflow"
        excluded="stroomsterkte" qualitative="unspecified"/>
<t_edge tail="spanning_over_de_condensator" head="inflow"
        excluded="stroomsterkte, spanning_over_de_weerstanden"
        qualitative="unspecified"/>
<t_edge tail="spanning_over_de_weerstanden" head="inflow"
        excluded="stroomsterkte" qualitative="unspecified"/>
<t_edge tail="lampje_1" head="stroomsterkte"
        excluded="vervangingsweerstand" qualitative="inverse"/>
<t_edge tail="lampje_2" head="stroomsterkte"
        excluded="vervangingsweerstand" qualitative="inverse"/>
<t_edge tail="bronspanning" head="stroomsterkte"
```

```
         excluded="spanning_over_de_weerstanden" qualitative="linear"/>
 <t_edge tail="spanning_over_de_condensator" head="stroomsterkte"
         excluded="spanning_over_de_weerstanden" qualitative="inverse"/>
 <t_edge tail="condensator_capaciteit" head="stroomsterkte"
         excluded="spanning_over_de_weerstanden, spanning_over_de_condensator"
         qualitative="asymptotical"/>
 <t_edge tail="lading" head="stroomsterkte"
         excluded="spanning_over_de_condensator, spanning_over_de_weerstanden"
         qualitative="inverse"/>
 <t_edge tail="condensator_capaciteit" head="spanning_over_de_weerstanden"
         excluded="spanning_over_de_condensator" qualitative="inverse"/>
 <t_edge tail="lading" head="spanning_over_de_weerstanden"
         excluded="spanning_over_de_condensator"
         qualitative="inverse"/>
</reference_model>
```

A.2.3 Rule set

```
<rules>
 <rule name="ordertophase2rule" result_type="logical">
  <count at_least="1" at_most="2">
   <node type="flow"/>
  </count>

  <true at_least="4">
   <node term="lampje_1" status="present"/>
   <node term="lampje_2" status="present"/>
   <node term="bronspanning" status="present"/>
   <node term="condensator_capaciteit" status="present"/>
   <node term="lading" status="present"/>
  </true>

  <true at_least="1">
   <node term="stroomsterkte" status="present"/>
   <node term="spanning_over_de_condensator" status="present"/>
   <node term="spanning_over_de_weerstanden" status="present"/>
   <node term="vervangingsweerstand" status="present"/>
  </true>

  <or>
   <true at_least="5">
    <edge tail="lading" head="spanning_over_de_condensator"
          direction="correct"/>
    <edge tail="condensator_capaciteit" head="spanning_over_de_condensator"
          direction="correct"/>
    <edge tail="spanning_over_de_condensator" head="spanning_over_de_weerstanden"
          direction="correct"/>
    <edge tail="bronspanning" head="spanning_over_de_weerstanden"
```

```
       direction="correct"/>
 <edge tail="spanning_over_de_weerstanden" head="stroomsterkte"
       direction="correct"/>
 <edge tail="vervangingsweerstand" head="stroomsterkte"
       direction="correct"/>
 <edge tail="lampje_1" head="vervangingsweerstand"
       direction="correct"/>
 <edge tail="lampje_2" head="vervangingsweerstand"
       direction="correct"/>

 <t_edge tail="lampje_1" head="stroomsterkte"
         direction="correct"/>
 <t_edge tail="lampje_2" head="stroomsterkte"
         direction="correct"/>
 <t_edge tail="bronspanning" head="stroomsterkte"
         direction="correct"/>
 <t_edge tail="condensator_capaciteit" head="stroomsterkte"
         direction="correct"/>
 <t_edge tail="lading"head="stroomsterkte"
         direction="correct"/>
 <t_edge tail="spanning_over_de_condensator" head="stroomsterkte"
         direction="correct"/>
 <t_edge tail="condensator_capaciteit" head="spanning_over_de_weerstanden"
         direction="correct"/>
 <t_edge tail="lading" head="spanning_over_de_weerstanden"
         direction="correct"/>
</true>
<and>
 <true at_least="4">
  <edge tail="lading" head="spanning_over_de_condensator"
        direction="correct"/>
  <edge tail="condensator_capaciteit" head="spanning_over_de_condensator"
        direction="correct"/>
  <edge tail="spanning_over_de_condensator" head="spanning_over_de_weerstanden"
        direction="correct"/>
  <edge tail="bronspanning" head="spanning_over_de_weerstanden"
        direction="correct"/>
  <edge tail="spanning_over_de_weerstanden" head="stroomsterkte"
        direction="correct"/>
  <edge tail="vervangingsweerstand" head="stroomsterkte"
        direction="correct"/>
  <edge tail="lampje_1" head="vervangingsweerstand"
        direction="correct"/>
  <edge tail="lampje_2" head="vervangingsweerstand"
        direction="correct"/>
  <t_edge tail="lampje_1" head="stroomsterkte"
          direction="correct"/>
  <t_edge tail="lampje_2" head="stroomsterkte"
          direction="correct"/>
```

```
     <t_edge tail="bronspanning" head="stroomsterkte"
             direction="correct"/>
     <t_edge tail="condensator_capaciteit" head="stroomsterkte"
             direction="correct"/>
     <t_edge tail="lading"head="stroomsterkte"
             direction="correct"/>
     <t_edge tail="spanning_over_de_condensator" head="stroomsterkte"
             direction="correct"/>
     <t_edge tail="condensator_capaciteit" head="spanning_over_de_weerstanden"
             direction="correct"/>
     <t_edge tail="lading" head="spanning_over_de_weerstanden"
             direction="correct"/>
    </true>

    <true at_least="1">
     <edge tail="stroomsterkte" head_type="flow"/>
     <t_edge tail="vervangingsweerstand" head_type="flow"/>
     <t_edge tail="spanning_over_de_condensator" head_type="flow"/>
     <t_edge tail="spanning_over_de_weerstanden" head_type="flow"/>
    </true>
   </and>
  </or>
 </rule>

 <rule name="ordertophase3rule" result_type="logical">
  <count at_least="1" at_most="2">
   <node type="flow"/>
  </count>

  <true at_least="4">
   <node term="lampje_1" status="present"/>
   <node term="lampje_2" status="present"/>
   <node term="bronspanning" status="present"/>
   <node term="condensator_capaciteit" status="present"/>
   <node term="lading" status="present"/>
  </true>

  <true at_least="1">
   <node term="stroomsterkte" status="present"/>
   <node term="spanning_over_de_condensator" status="present"/>
   <node term="spanning_over_de_weerstanden" status="present"/>
   <node term="vervangingsweerstand" status="present"/>
  </true>

  <or>
   <true at_least="5">
    <edge tail="lading" head="spanning_over_de_condensator"
          direction="correct" e_qualitative="correct"/>
    <edge tail="condensator_capaciteit" head="spanning_over_de_condensator"
```

```
      direction="correct" e_qualitative="correct"/>
 <edge tail="spanning_over_de_condensator" head="spanning_over_de_weerstanden"
      direction="correct" e_qualitative="correct"/>
 <edge tail="bronspanning" head="spanning_over_de_weerstanden"
      direction="correct" e_qualitative="correct"/>
 <edge tail="spanning_over_de_weerstanden" head="stroomsterkte"
      direction="correct" e_qualitative="correct"/>
 <edge tail="vervangingsweerstand" head="stroomsterkte"
      direction="correct" e_qualitative="correct"/>
 <edge tail="lampje_1" head="vervangingsweerstand"
      direction="correct" e_qualitative="correct"/>
 <edge tail="lampje_2" head="vervangingsweerstand"
      direction="correct" e_qualitative="correct"/>
 <t_edge tail="lampje_1" head="stroomsterkte"
        direction="correct" e_qualitative="correct"/>
 <t_edge tail="lampje_2" head="stroomsterkte"
        direction="correct" e_qualitative="correct"/>
 <t_edge tail="bronspanning" head="stroomsterkte"
        direction="correct" e_qualitative="correct"/>
 <t_edge tail="condensator_capaciteit" head="stroomsterkte"
        direction="correct" e_qualitative="correct"/>
 <t_edge tail="lading"head="stroomsterkte"
        direction="correct" e_qualitative="correct"/>
 <t_edge tail="spanning_over_de_condensator" head="stroomsterkte"
        direction="correct" e_qualitative="correct"/>
 <t_edge tail="condensator_capaciteit" head="spanning_over_de_weerstanden"
        direction="correct" e_qualitative="correct"/>
 <t_edge tail="lading" head="spanning_over_de_weerstanden"
        direction="correct" e_qualitative="correct"/>
</true>
<and>
 <true at_least="4">
  <edge tail="lading" head="spanning_over_de_condensator"
       direction="correct" e_qualitative="correct"/>
  <edge tail="condensator_capaciteit" head="spanning_over_de_condensator"
       direction="correct" e_qualitative="correct"/>
  <edge tail="spanning_over_de_condensator"
       head="spanning_over_de_weerstanden"
       direction="correct" e_qualitative="correct"/>
  <edge tail="bronspanning" head="spanning_over_de_weerstanden"
       direction="correct" e_qualitative="correct"/>
  <edge tail="spanning_over_de_weerstanden" head="stroomsterkte"
       direction="correct" e_qualitative="correct"/>
  <edge tail="vervangingsweerstand" head="stroomsterkte"
       direction="correct" e_qualitative="correct"/>
  <edge tail="lampje_1" head="vervangingsweerstand"
       direction="correct" e_qualitative="correct"/>
  <edge tail="lampje_2" head="vervangingsweerstand"
       direction="correct" e_qualitative="correct"/>
```

```
     <t_edge tail="lampje_1" head="stroomsterkte"
             direction="correct" e_qualitative="correct"/>
     <t_edge tail="lampje_2" head="stroomsterkte"
             direction="correct" e_qualitative="correct"/>
     <t_edge tail="bronspanning" head="stroomsterkte"
             direction="correct" e_qualitative="correct"/>
     <t_edge tail="condensator_capaciteit" head="stroomsterkte"
             direction="correct" e_qualitative="correct"/>
     <t_edge tail="lading"head="stroomsterkte"
             direction="correct" e_qualitative="correct"/>
     <t_edge tail="spanning_over_de_condensator" head="stroomsterkte"
             direction="correct" e_qualitative="correct"/>
     <t_edge tail="condensator_capaciteit" head="spanning_over_de_weerstanden"
             direction="correct" e_qualitative="correct"/>
     <t_edge tail="lading" head="spanning_over_de_weerstanden"
             direction="correct" e_qualitative="correct"/>
    </true>
    <true at_least="1">
     <edge tail="stroomsterkte" head_type="flow" />
     <t_edge tail="vervangingsweerstand" head_type="flow" />
     <t_edge tail="spanning_over_de_condensator" head_type="flow" />
     <t_edge tail="spanning_over_de_weerstanden" head_type="flow" />
    </true>
   </and>
  </or>
 </rule>

</rules>
```