Chapter 12
Using Drawings in Knowledge Modeling and Simulation for Science Teaching

Wouter R. van Joolingen, Lars Bollen, and Frank A.J. Leenaars

University of Twente, Faculty of Behavioral Sciences, P.O. Box 217, 7500 AE Enschede, The Netherlands
{w.r.vanjoolingen,l.bollen}@utwente.nl,
f.a.j.leenaars@student.utwente.nl

Abstract. Modeling knowledge in simulation-based inquiry learning requires a model of the domain that is executable, as well as a model of the learners' knowledge about the domain. An intermediate level is formed by models of the domain that are created by students as is done in modeling environments. An approach is presented for generating student created models from drawings. This approach requires drawing segmentation, shape recognition and model generation, which is done based on density-based clustering, elementary shape recognition combined with a shape ontology and model fragment composition respectively. The final result is an executable model that can be used to generate simulation outcomes based on learners' conceptions. The role of such a system is discussed, especially with respect to the diagnosis of misconceptions and the generation of tutoring interventions based on confronting learners with the consequences of their conceptions.

12.1 Introduction

In inquiry learning with the help of simulations and modeling, knowledge is modeled at three levels. First there is the level of the *authored simulation* (van Joolingen and de Jong 2003) in the form of an executable model that drives the simulation as a main resource for inquiry. The second level is that of *models created by students*, in the form of concept maps, system dynamics models or stated hypotheses and conclusions (Novak 1998; Penner 2001; Schwarz et al. 2007; Wilensky and Reisman 2006; Wilensky and Resnick 1999). The third level is that of *models the system makes of learners' knowledge*. Although these levels of knowledge modeling serve different purposes and therefore need to satisfy different requirements, they also have much in common as they rely on similar representations representing relations between variables in the domain. In many cases representations at all three levels need to be simulated. At the level of the simulation this is obvious. It has also been known that for the level of learner created models a simulation based on a learner generated model can have a beneficial effect on the learning process (Alessi 2000; Bliss 1994; Ergazaki et al. 2007;

R. Nkambou et al. (Eds.): Advances in Intelligent Tutoring Systems, SCI 308, pp. 249–264.
springerlink.com

Feurzeig and Roberts 1999; Sins et al. 2005). At the level of the model of learners' knowledge simulation of a model can help to identify conflicts between learners' hypotheses and predictions on one side and the model that the learner is studying on the other. Differences in models generate difference in simulation results which are opportunities to confront and discuss with learners in the knowledge building process.

In the current chapter we focus on the middle level, the representation of models by learners. As noted, learners can use many different kinds of representation to express their own models. Some of those representations, such as system dynamics models allow to be simulated right away, but have the drawback of being quite formal and requiring prior knowledge of the variables and relations in the domain, as well as knowledge about the notation and syntax of system dynamics. Other representations such as concept maps can be helpful, but cannot be simulated. Moreover, concept maps are good for building conceptual structures, but they are not really geared towards representing computational operations. Finally, concept maps also impose a kind of formalism on the kind of representations to use by students. Learning environments such as Cool Modes (Bollen et al. 2002) try to combine different visual languages like system dynamics, UML diagrams, freehand drawing etc. in one workspace, but these languages rarely interoperate, and especially freehand-drawings are only integrated on a visual level.

We try to address the problem for representing learners' models by letting learners make drawings representing their understanding of the domain. Using freehand drawings and sketches provides the most representational freedom, but it usually lacks any form of operational semantics. Recent sketch recognition systems try to include this kind of modeling support to drawings, e.g., for drawing logic diagrams (Alvarado and Lazzareschi 2007) or for recognizing mathematical expressions (LaViola and Zeleznik 2004), but they also inherit the limitations and restriction from the domain they are trying to support and from the language they try to recognize.

The approach presented in this paper brings together representational freedom and operational semantics. This approach allows learners to externalize and visualize their ideas on a phenomenon by using freehand drawings, which can be used to intelligently support the creation of a quantitative model by means of segmentation support to recognize coherent components in a drawing, sketch recognition for detecting basic shapes (e.g. arrows, links between components) and labeling to provide a means to the user to identify and tag relevant characteristics and properties of sketch components.

12.2 Modeling with Inaccurate Drawings

In this section we will describe the main properties of the system to generate models from drawings. We start with describing the context and rationale, and proceed with describing the necessary steps to move from drawing to model. In subsequent sections, the first results of implementing the approach will be presented.

When creating a model, many people, including experts, start by making a drawing that is a more or less schematic representation of the system that is being modeled. Drawings help identifying the main components that need to be included

in a model and be represented as one or more variables (Van Meter and Garner 2005). Therefore, drawings can form a bridge between initial ideas and a formal model of a system. An example may make this clearer.

Suppose learners study the water cycle, that represents the origin of rainfall (water evaporates from the sea, condensates, flows to land where it forms raindrops and it starts to rain). Rain water comes together in rivers that feed back to the sea. A drawing of such a system could look like Fig. 12.1. Such a drawing by no means qualifies as a computational model as it is inaccurate and ambiguous (for instance there are several arrows that all have different meanings, and important concepts, such as the temperature of the water, are not represented. However, the drawing does represent relevant components of the system – sea, water in various states, wind, the sun as energy source – as well as processes (evaporation, flow, condensation) and could help in arriving at a model such as the system dynamics model represented in Fig. 12.2, that can be used to simulate the water cycle and investigate the influence of parameters such as the intensity of the Sun's radiation. On the other hand, the drawing conveys concepts and details that may not be included in a formal system dynamics model, like spatial relations between components (e.g. mountain, river, sea).

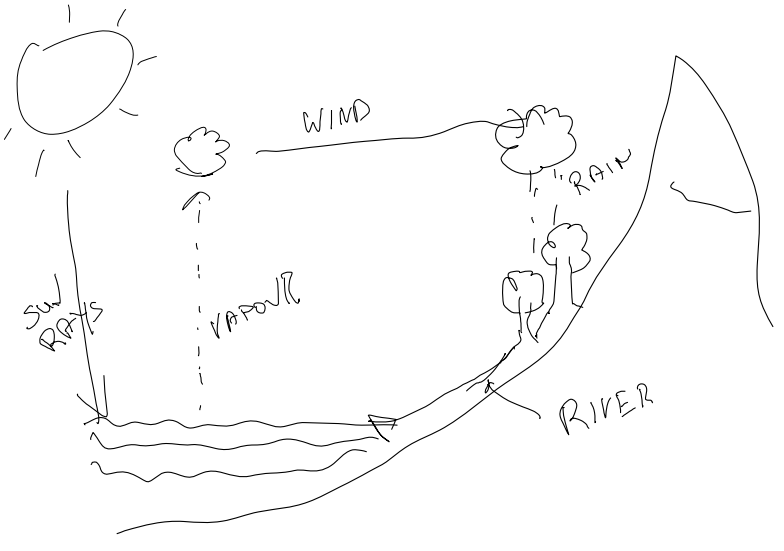

Fig. 12.1 Possible drawing to start modeling the water cycle.

The basic idea of our approach is to support learners in transferring their ideas, as expressed in a drawing, into a formal model; either by translating the drawing into a formal language such as system dynamics, or by adding information to the drawing in such a way that it becomes a computational model.

In the first case, a model such as the one presented in Fig. 12.2 would be created using the drawing as a means of support for creating the model elements, in

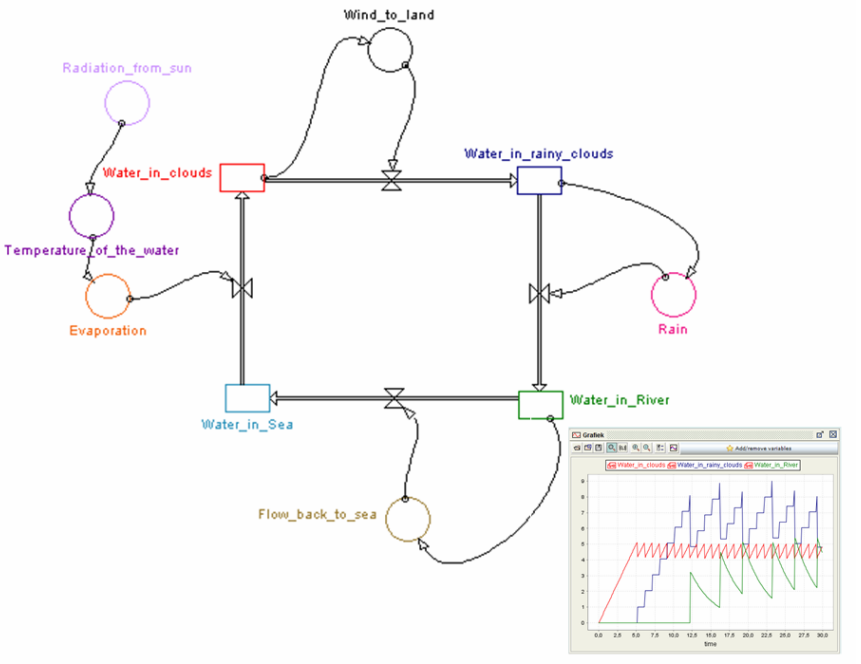

Fig. 12.2 Possible system dynamics model of the water cycle (created with Co-Lab (van Joolingen et al. 2005))

the second case, the drawing itself could be the model, meaning for instance that the drawing elements could be animated, based on the state of the model. For instance, the cloud could grow as more water evaporates and it could move onto the land area.

In order to make this work, the drawing must be used to identify objects, variables and processes that will be components of the model. In our drawing there are objects recognizable as containers of water (the sea, clouds above the sea, clouds above the land, the river). Each of these objects has a pictorial representation. Moreover, there are processes such as heating the sea water, transporting water through wind or flow, and rain. These processes are represented as arrows (such as the wind) or icons (such as the raindrops). An executable model would require formulation in terms of *variables* and *relations* (such as computational functions or differential equations). In the system dynamics formalism, this would come down to *stocks* (variables that represent a state of a container (e.g. the amount of water in a cloud), a *flow*, representing the rate of change of one or two states (e.g. water evaporating from the sea influences the amount of water in the sea as well as the amount of water in the clouds), or an *influence*, such as the temperature of the sea water influencing the evaporation rate.

Understanding learners' drawings can be supported by a system that understands the drawing to a certain extent. For such support to work it is necessary that

the system (1) recognizes elements in the drawing, (2) enters a dialogue with the learner about the meaning of these elements and helps formalizing them into a model and (3) supports the simulation based on the model. The purpose of such a support system would be to bring modeling approaches within the reach of students that have not been trained in formal modeling techniques. Also drawings can support beginning modelers in learning such a formalism and potentially also help experienced modelers in dealing with models of complex systems. The next section will elaborate on the details of this support.

12.2.1 Converting a Drawing into a Model

To achieve the kind of support mentioned in the previous chapter, and to integrate sketching and modeling, a number of different approaches are being implemented and evaluated. As we aim for a generic, domain-independent modeling support, there will be no single, ideal solution, but a number of different approaches will act together to provide flexible, yet powerful assistance to the learner. In the following, various approaches are introduced and discussed. Together they form a step-by-step plan to generate a model out of a learner's drawing.

12.2.2 Segmentation and Grouping

To identify distinct objects in a drawing, clustering algorithms or Bayes classification constitute appropriate techniques. Taking into account information on time, location, color and thickness of each stroke, collections of strokes that belong together can be automatically detected.

This can be regarded as a preparatory step and support for the learner to identify relevant objects in a sketch. First experiences, which are described and illustrated in detail in the next chapter, showed that the results of automatic segmentation approaches are promising, but that this approach is also dependent on the learner's individual drawing style and needs to allow for users' interaction and intervention, e.g. by querying for ambiguous parts of a drawing or by providing manual segmentation and grouping features.

12.2.3 Sketch Recognition and Labeling

A number of sketch recognition systems have been developed and are used in learning and teaching settings) (Alvarado and Lazzareschi 2007; Hammond and Davis 2003; LaViola and Zeleznik 2004), and the increased spreading of pen-based devices, like Tablet PCs, PDAs and smartphones, recently promoted their usage. However, the available applications either focus on recognizing elements from specific domains, like logic diagrams (Alvarado and Lazzareschi 2007), letter recognition (Koile et al. 2007) or they aim at the beautification of strokes as in (Paulson and Hammond 2008).

When dealing with arbitrary sketches of varying domains, no currently available approach would be able to recognize elements of domain-independent

drawings. For example, the trees, the sea and the clouds in Fig. 12.1 are not identifiable with reasonable effort. Still, sketch recognition approaches as proposed in (Tracy Hammond & Davis, 2003) are expected to be helpful to find basic and typical elements like arrows, geometric shapes, connecting lines, etc. that can be used in combination with grouping and labeling approaches as described in the next sections.

Labeling is a manual way to add descriptive tags to elements of a drawing, as shown in Fig. 12.3 below. Adding labels serves two purposes: (1) It helps a learner to externalize his ideas and to think about the meaning and characteristics of his sketch, and (2) it may be used to automatically deduce initial, draft models from a drawing, using the labels as variables and parameters. A leaner may be instructed to use labels in a specific way, as mentioned above.

12.2.4 Model Generation

Once recognized, either through automatic recognition or manual labeling by the learner, the various components need to be converted into elements of a computational model, and be connected in such a way that an executable model emerges. Such an approach has been already used in the qualitative modeling system GARP (Bouwer and Bredeweg 2001) that employs the idea of connecting *model fragments* into complete models. Using a model fragment for each of the identified sketch components, qualitative or quantitative attributes can be assigned to each of them. For instance, to the component representing the sea in Fig. 12.1, a temperature and an evaporation rate can be assigned. Arrows in the figure can be used to identify the way the fragments need to be linked together.

12.2.5 Integration of Modeling and Drawing

The techniques mentioned in the previous sections aim at facilitating a tight integration of drawing and modeling. Elements from a learner's drawing are used to create an initial model and serve as a starting point for the learner's modeling activities. In reverse, a drawing illustrates a model and is able to provide additional information that cannot be expressed in a formal modeling language. Furthermore, first study results indicate that creating a drawing in addition to modeling or text-writing activities can particularly activate learners' prior knowledge.

Fig. 12.1 and Fig. 12.2 intuitively illustrate this argumentation: The drawing, as an external representation of a learner's knowledge, depicts a complex phenomenon and "tells a story". It could be used in a presentation, for own recollection or to explain the issue to peers. Even more, the drawing contains elements that are relevant for the water cycle and weather in general, e.g. the trees and the mountains, but that are lacking in the model. In the system dynamics model, though, other (quantitative and temporal) aspects are represented, e.g. the simulation and the graphs explain why it is raining periodically and not permanently, as could be guessed from the drawing.

A complementary approach to the one outlined above is CogSketch by Forbus (Forbus & Usher, 2002). In contrast to the approach proposed here, in CogSketch the learner explicitly indicates the individual objects in the drawing by creating so-called "glyphs". A glyph can then be labeled and assigned to pre-defined objects from a given knowledge-base. As there is no sketch-recognition feature in CogSketch, it is possible to have large discrepancies between the sketched glyph and the assigned object, that is the glyph does not have to look like the object represented.

CogSketch, however, provides interesting capabilities in terms of reasoning with spatial relationships between drawing elements. Such relationships can be exploited in identifying relations between objects that need to be included in the model. As a consequence, it is possible to provide tutorial support in the form of prompts based on spatial features of the sketch. In a nutshell, we expect that a strong integration of drawing and modeling is beneficial for prior knowledge activation, knowledge externalization and modeling activities.

12.3 Supportive Technologies and First Results

In the current section, the above mentioned supportive technologies for segmentation, labeling, sketch recognition, and model integration will be picked up again and first results will be described.

12.3.1 Grouping and Labeling

For grouping and labeling we performed an exploratory study to gain insight into the way students represent real world systems as freehand drawings. The study deals specifically with drawings created for the purpose of answering questions about certain variables and relations in the described system. Results of this study lead to the following questions:

- How are described systems commonly represented by students?
- How do students use labels in their drawings and how could these labels be used by the system?
- Can these drawings be automatically segmented?

12.3.1.1 Study Setup

Ten participants used a graphics tablet to create their drawings. They worked in a specifically created sketch collection environment, which integrates drawing, labeling and simulation tools. The simulation tool was based on SimQuest (van Joolingen and de Jong 2003; van Joolingen et al. 1997). Descriptions of real world systems were presented to participants as short case texts along with a simulation that allowed participants to manipulate a number of variables. Extensive logs were kept of all the participants' actions in this application. This environment also offered tutorials to familiarize participants with both the graphics tablet and the software. Figure 12.3 shows a screenshot of this environment.

When participants felt comfortable working in this environment, they were asked to create sketch representations of two real world systems. The first system consisted of a toy car with a small engine that was connected to a table with a rubber band. The second system concerned a house, leaking energy to its environment, heated by a radiator that was controlled by a thermostat. Participants were then asked to add labels to those parts of their drawing that they believed had an effect on a specified variable. An English translation of the first case text is given below. The second case text was similar in length and amount of detail.

"A toy car is connected to a table leg with a rubber band. The car contains a small engine that produces a constant forward force on the car. The engine is switched on and the car starts to move away from the table leg, causing the rubber band to be pulled tight. Because the rubber band pulls the car backwards, the car may start to oscillate, but it is also possible that it slowly comes to a halt without oscillating. Make a sketch of this situation and use labels to identify those parts of your drawing that have an effect on whether or not the car will oscillate. Decide whether the variables mentioned below are relevant, and try to think of as many other relevant variables as possible."

An implementation of the locally scaled density based clustering (LSDBC) algorithm, as described in (Biçici and Yuret 2007), was used to locate clusters in the participants' drawings. Each point in the drawing was defined by its X, Y and time coordinates.

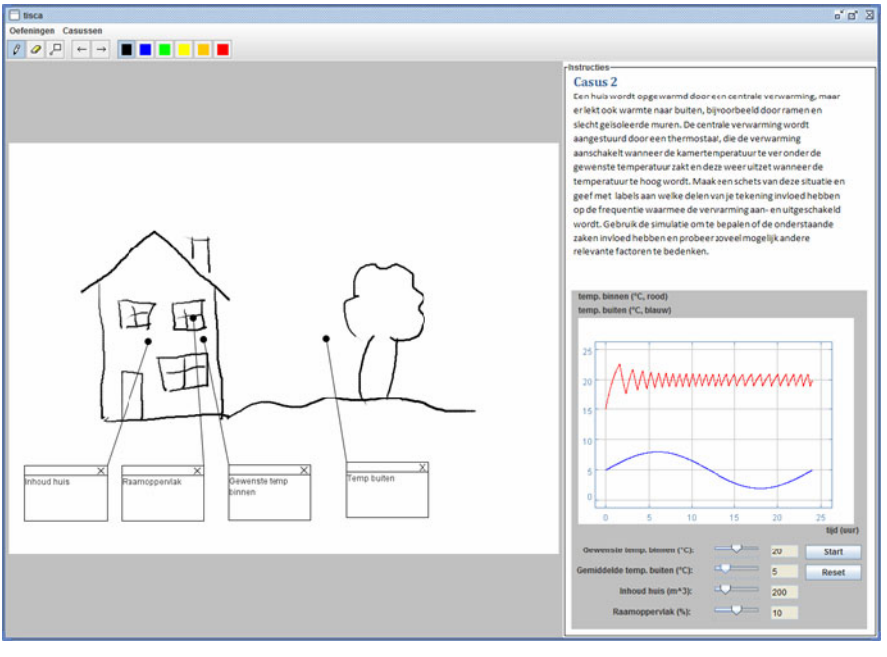

Fig. 12.3 Screenshot of the sketch collection environment. The right side of the picture shows a short case description and a simulation, the left side shows a labeled sketch that was drawn based on this information

12.3.1.2 Results

First results show that although there was quite some variety in the way participants represented the described systems, there were many similarities as well. About half the participants drew a solid house, while the other half drew a house that was 'see-through', so they could show the thermostat and radiator. Another interesting pattern was that all participants represented the table / rubber band / toy car system with the table on the left side and the car driving away to the right. While interesting in itself, this is the kind of result that could be very useful for distinguishing and identifying objects in an automated system.

Participants used labels to indicate which parts of their drawings they believed had an influence on a specified variable. Labels were not used to identify other objects that were drawn. For example the tree in Figure 12.3 is not labeled because it is not believed to influence the temperature in the house, but the window (specifically the surface area of the window) is labeled. During the course of the study participants were asked to give more information in the labels. E.g. instead of just mentioning that the surface area of the windows affects the temperature in the house, participants were later asked to write in more detail about the direction of this effect. When labels are used to identify as many objects as possible in the drawing, this can be of great help during clustering and sketch recognition. During the clustering phase, the labels can serve as seed points for the clustering algorithm and the text in the labels can be scanned for domain specific words to help in the sketch recognition phase. For instance, if a label containing the string 'window' is used to identify a rectangular shape; this is quite strong evidence that this shape in fact represents a window. A drawback of this approach is that it would require domain specific lexicons containing description strings of the objects to be recognized.

Other interesting results have already been found by the clustering algorithm, which was at times able to very accurately detect different parts of the drawing. Figure 12.4 shows the results of applying the LSDBC algorithm to four different drawings of a toy car connected to a table with a rubber band. The clustering algorithm accurately detected different parts of the sketch in all but the bottom-right drawing. The algorithm was able to distinguish the rubber band from the table by using time information. While this leads to good results when the drawing order is table – car – rubber band, it fails when the drawing order is table – rubber band – car, as it was in the bottom-right drawing. The current clustering algorithm could be further improved by using color and stroke information.

Recent developments used the participants' drawings as training data for a Naïve Bayes Kernel Distribution approach, using the RapidMiner / YALE libraries (Mierswa et al. 2006). This approach turned out to be more accurate than the LSDBC clustering approach to identify distinct objects in a sketch.

12.3.2 Sketch Recognition

Once partitioned into separate segments, the objects that have been found need to be identified as components of a model. The approach used for this is based on

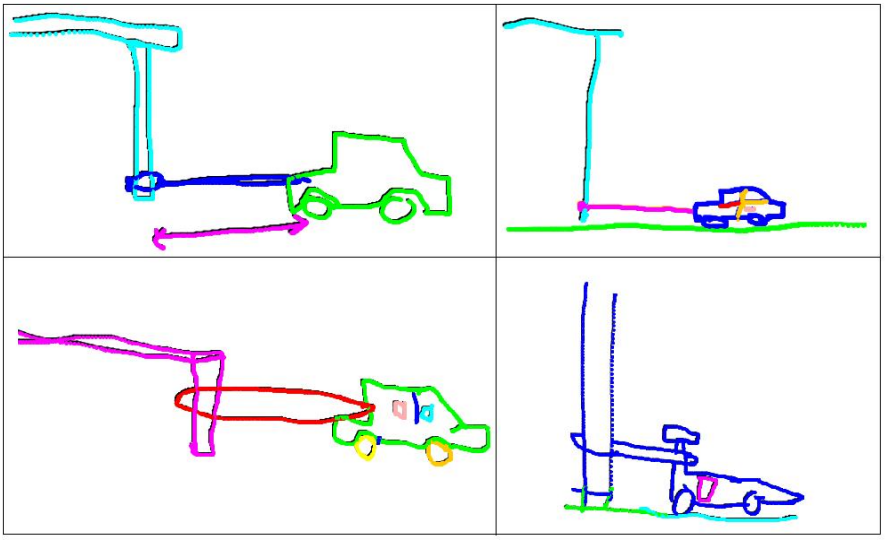

Fig. 12.4 Four drawings of a toy car connected to a table by a rubber band. The colors were added by the clustering algorithm, with each color representing a different cluster.

creating a large library of model elements that can cover a number of domains. In the examples listed above, elements could be, for instance, *house*, *car*, or *cloud.* Apart from manual labeling by students, there are two ways in which shapes can be recognized. The first is based on the sketch recognition algorithm as devised by Hammond (Hammond & Davis, 2005; Paulson & Hammond, 2008). This algorithm identifies elementary shape elements such as lines, ellipses and spirals. These elements can be combined into more complex shapes such as squares, arrows and more by specifying rules in the LADDER language ("Language for Describing Drawing, Display, and Editing for use in sketch Recognition"). LADDER rules use concepts such as above/below, perpendicularity of lines, joint points etc. These complex shapes can again be reused to define even more complex shapes. For instance, a *house* can be defined in terms of a *rectangle,* with a *triangle* on top of it. A rectangle in its turn is defined as four lines, that meet in four points under straight angles. As we are dealing with freehand drawings, all rules include an amount of tolerance. However, as a trade-off, the more objects you define, the less tolerant your rules have to be to avoid ambiguity. As a consequence, a drawing has to be very accurate to be still recognizable.

A second approach for sketch identification would be using a database of drawings that are manually classified and, using data mining techniques, match the characteristics of a learner's drawing to that database. For this, basic shapes such as horizontal and vertical strokes, ellipses, etc. still need to be classified, but there is no need for defining the complex shapes. It may be expected that data mining techniques prove less sensitive to variation in details of the drawing (for instance to the exact shape of a cloud) and more tolerant to mistakes than those based on pure shape recognition.

12.3.3 Converting a Drawing into a Model

To achieve the kind of support mentioned in the previous chapter, and to integrate sketching and modeling, a number of different approaches are being implemented and evaluated. As we aim for a generic, domain-independent modeling support, there will be no single, ideal solution, but a number of different approaches will act together to provide flexible, yet powerful assistance to the learner. In the following, various approaches are introduced and discussed. Together they form a step-by-step plan to generate a model out of a learner's drawing.

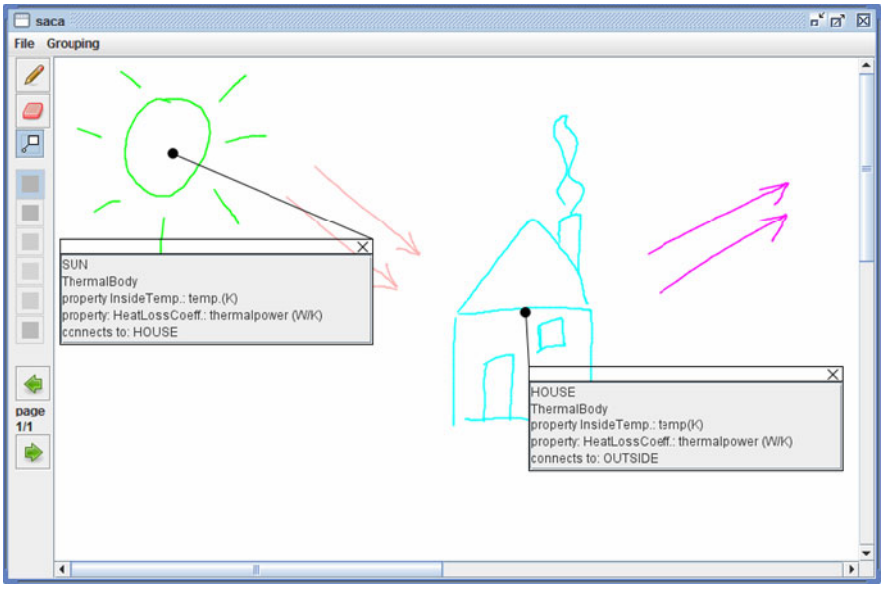

Fig. 12.5 A simple drawing in the domain of heating a house. The coloring originates from automatic grouping, the labels stem from automatic shape recognition

In a trial with data collected from the students in the study described above, a LADDER-based algorithm was capable of positively identifying shapes such as *house*, *sun* and *arrow*. The main addition to the standard LADDER algorithm was that the recognition took place after partitioning the drawing into distinct, smaller objects. The advantage of this approach is that interference of different shapes is avoided, and that parts of the shape (e.g. a window in the house) that are not part of the definition can be included in the shape. This allows for a 'loose' definition of shapes, e.g. the house reducing to just four lines that indicate two walls and a roof. This results in a very accurate recognition of shapes, as is shown in Figure 12.5. The robustness of this approach needs to be tested on larger shape libraries with more shapes.

12.3.4 Model Generation

The final stage in creating an executable model out of the drawing is to link the shapes, identify partial models and to create causal and computational links between the components. In this approach we assume that the system as a whole can be modeled in ordinary differential equations (ODE). ODEs are suitable to model many systems, and modeling systems such as System Dynamics base themselves on ODEs. This approach will not cover all systems (in particular discrete systems such as cellular automata would be excluded) but we! believe similar approaches could be found for such systems as well.

```
ThermalBody
canConnectTo  neighbouringBody : ThermalBody (0..N)
properties
public property      Temperature : temperature (K)
internal property    HeatCapacity : thermalcapacity (J/K)
internal property    ThermalEnery : energy (J)
connection(neighbouringBody) property HeatLossCoefficient :
   thermalpower (W/K)
equations
Temperature = ThermalEnergy / HeatCapacity
Var TotalHeatLoss =
SUM-OVER-ALL(neighbouringBodies) {
    (self<->neighbouringBody).HeatlossCoefficient *
       neighbouringBody.Temperature – Temperature
}
dThermalEnergy/dt = TotalHeatLoss
end ThermalBody
```

Fig. 12.6 Model fragment identifying a thermal body

The basic idea behind the model generation is the use of model fragments and connectors. For the example of the house heating this will be explained. First of all, an object such as *house* as identified by the sketch recognition mechanism is hardly useful for modeling. In order to make it useful, physical properties have to be added. This is done by creating a hierarchical ontology that can classify a house as a *thermal-body*, that associates with a model fragment that identifies the properties of that physical object as well as the connections it can have to other objects. These connections in their turn can have properties of their own. In Fig. 12.6 the model fragment for a thermal body is given. Note statements that specify that a thermal body can connect to any number of other thermal bodies, representing the exchange of heat. This adds flexibility compared to standard functional block approaches in which the number of connections is fixed for each object. Also each connection can have its unique properties, in this case the heat loss coefficient that determines how much heat is transferred over the link per unit of time.

Based on the layout of the drawing, model fragments can be combined. For instance, let us assume that we model a system consisting of a house and its environment. From the drawing a connection between the environment (ENV) and the house (HOUSE) can be inferred. This means that two thermal object model fragments can be instantiated, including a link between them. This instantiation leads to the set of equations that is presented in Fig. 12.7. This is a set of two ODE's and four algebraic equations that can be fed into a simulation engine such as available in – for instance – the SimQuest simulation system (van Joolingen & de Jong, 2003).

```
HOUSE.Temperature = HOUSE.ThermalEnergy / HOUSE.HeatCapacity
HOUSE.TotalHeatLoss = (HOUSE<->ENV).HeatlossCoefficient
    *(ENV.neighbouringBody.Temperature - HOUSE.Temperature)
dHOUSE.ThermalEbergy/dt = HOUSE.TotalHeatLoss
ENV.InsideTemperature = HOUSE.ThermalEnergy / HOUSE.HeatCapacity
ENV.TotalHeatLoss = (ENV<->HOUSE).HeatlossCoefficient *
    (HOUSE.neighbouringBody.Temperature - ENV.Temperature)
dENV.ThermalEbergy/dt = ENV.TotalHeatLoss
```

Fig. 12.7 Set of equations that can be generated from a drawing

This – relatively simple – example shows how the final step from drawing to model can be made. For more complex systems, of course the number of elements and links will grow, but given the complexity of an average drawing, we do not expect computational problems.

12.4 Conclusion and Outlook

In the preceding sections we presented a means of deriving a computational model from a more or less systematic drawing made by a learner. The foreseen benefits are twofold, aiming at the different levels of knowledge modeling addressed in this paper. For the level in which the learner models his own knowledge, generating a simulation based on that knowledge can have a positive effect on the learning process. Learners will be confronted with the consequences of their own ideas, and will be able to adapt these ideas based on this confrontation.

The approach we sketch here extends the possibility of generating computational models beyond using standard modeling languages such as system dynamics, LOGO or NetLogo. Instead of asking learners to learn a representational formalism, we create a system that generates formal models from the representations that learners create spontaneously. The advantages are that (1) in this way the idea of modeling by learners can have a wider application beyond that within dedicated modeling systems and (2) the system can make more extended use of the information it retrieves from learner generated data such as diagrams and models.

An example of the latter would be that the system simulates a learner's model and compares the results with the data that learners obtain from the simulation.

Presenting both simulation results and indicating the differences can result in a dialogue with the learner. Agents in the learning environment can use this information to adapt the learning environment, and for instance suggest experiments that provoke thought on their misconceptions, in the line of Posner and colleagues (Posner et al. 1982). This can best be illustrated by an example. A common misconception in astronomy is that the seasons are caused by the earth being closer to the sun in summers than in winters. This a misconception that can easily be picked up from a drawing. Agents can then respond in several ways:

- Tell the learner that the reality is different.
- Ask questions about the misconception and especially what this would mean for the times of seasons for the northern and southern hemispheres.
- Generate a simulation and run it to show that this would also mean that the summer would become shorter and that the temperature on the whole earth would rise and fall.

The option that would be chosen would be dependent on the pedagogical strategy, and on the level of the learner, but it is clear how the sketch-based modeling would enlarge the repertoire of ITSs, by extending the possibilities for diagnosis and feedback, including simulation-based feedback.

The work presented here is in progress. The segmentation part is up to a level that it can be used in practice. The recognition step can be covered for simple shapes, but there is a need for building and analyzing a large corpus of drawings to extract rules for more complex shapes than houses and suns. Once the corpus has been built, data mining techniques can be used to create and test classification rules. In the long run this should lead to a large library and ontology of learner generated shapes that can be used for identification of drawings, and become a learning system when it is also fed with the results of manual labeling by learners and experts.

The model generation part, finally, is rather straightforward once the shapes and their relations have been identified. Equations like the ones generated in the example presented above can be processed with existing simulation engines such as those available for SimQuest (van Joolingen and de Jong 2003). However, identifying relations is tricky, and strongly dependent on the way learners represent them. So far we need to rely on pre-stored relations and drawn arrows by learners.

In the long run, the outlook for drawing based modeling systems is that they can form an integrated part of inquiry-based environments that offer intelligent support for the learner. A traditional issue in Intelligent Tutoring Systems, including those based on inquiry learning, such as Co-Lab (van Joolingen et al. 2005) and SCY (de Jong et al. in press) is that of estimating the learners' knowledge level. The way we use drawings as described in this chapter provides a natural input for the knowledge modeling systems Drawings can form an excellent means of sharing and communicating knowledge, with fellow learners as well as with the tutoring system. If drawings can be augmented by a simulation, the learning environment can become a partner in the learning process, by detecting and acting

upon misconceptions, similarities and differences between drawings by several learners. As such we believe that drawing-based modeling will provide a valuable addition to the repertoire of ITS design.

References

Alessi, S.M.: The Application of System Dynamics Modeling in Elementary and Secondary School Curricula. Paper presented at the Conference of the Red Iberoamericana de Informática Educativa, RIBIE, Vina del Mar, Chile (2000)

Alvarado, C., Lazzareschi, M.: Properties of Real-World Digital Logic Diagrams. Paper presented at the Proceedings of 1st International Workshop on Pen-based Learning Technologies, PLT 2007, Catania, Italy (2007)

Biçici, E., Yuret, D.: Locally Scaled Density Based Clustering. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.) ICANNGA 2007. LNCS, vol. 4431, pp. 739–748. Springer, Heidelberg (2007)

Bliss, J.: From mental models to modelling. In: Mellar, H., Bliss, J., Boohan, R., Ogborn, J., Tompsett, C. (eds.) Learning with Artificial Worlds: Computer Based Modelling in the Curriculum. The Falmer Press, London (1994)

Bollen, L., Hoppe, H.U., Milrad, M., Pinkwart, N.: Collaborative Modeling in Group Learning Environments. Paper presented at the Proceedings of the XXth International Conference of the System Dynamics Society, Palermoy, Italy (2002)

Bouwer, A., Bredeweg, B.: VisiGarp: Graphical Representation of Qualitative Simulation Models. Paper presented at the Proceedings of 10th International Conference on Artificial Intelligence in Education, AI-ED 2001, San Antonio, Texas (2001)

de Jong, T., van Joolingen, W.R., Anjewierden, A., Bollen, L., d'Ham, C., Dolonen, J., et al.: Learning by creating and exchanging objects: the SCY experience. British Journal of Educational Technology (in press)

Ergazaki, M., Zogza, V., Komis, V.: Analysing Students' Shared Activity while Modeling a Biological Process in a Computer-Supported Educational Environment. Journal of Computer Assisted Learning 23(2), 158–168 (2007)

Feurzeig, W., Roberts, N.: Modeling and simulation in science and mathematics. Springer, New York (1999)

Forbus, K.D., Usher, J.: Sketching for Knowledge Capture: A Progress Report. Paper presented at the IU 2002, San Francisco (2002)

Hammond, T., Davis, R.: LADDER: A Language to Describe Drawing, Display, and Editing in Sketch Recognition. Paper presented at the Proceedings of International Joint Conference on Artificial Intelligence, Hyderabad, India (2003)

Hammond, T., Davis, R.: Ladder, a Sketching Language for User Interface Developers. Computers & Graphics 29, 518–532 (2005)

Koile, K., Chevalier, K., Low, C., Pal, S., Rogal, A., Singer, D., et al.: Supporting Pen-Based Classroom Interaction: New Findings and Functionality for Classroom Learning Partner. Paper presented at the 1st International Workshop on Pen-Based Learning Technologies, PLT 2007, Catania, Italy (2007)

LaViola, J., Zeleznik, R.: MathPad: A System for the Creation and Exploration of Mathematical Sketches. ACM Transactions on Graphics (Proc. SIGGRAPH) 23(3) (2004)

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid Prototyping for Complex Data Mining Tasks. Paper presented at the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006 (2006)

Novak, J.D.: Learning, creating and using knowledge: Concept map ™ as facilitative tools in schools and corporations. Lawrence Erlbaum Associates Inc., Mahwah (1998)

Paulson, B., Hammond, T.: Accurate Primitive Sketch Recognition and Beautification. Paper presented at the Proceedings of the International Conference on Intelligent User Interfaces, IUI 2008, Canary Islands, Spain (2008)

Penner, D.E.: Cognition, Computers, and Synthetic Science: Building knowledge and meaning through modelling. Review of Research in Education 25, 1–37 (2001)

Posner, G.J., Strike, K.A., Hewson, P.J., Gertzog, W.A.: Accomodation of a scientific conception: Towards a theory of conceptual change. Science Education 66(2), 211–227 (1982)

Schwarz, C.V., Meyer, J., Sharma, A.: Technology, Pedagogy, and Epistemology: Opportunities and Challenges of Using Computer Modeling and Simulation Tools in Elementary Science Methods. Journal of Science Teacher Education 18(2), 243–269 (2007)

Sins, P.H.M., Savelsbergh, E.R., van Joolingen, W.R.: The Difficult Process of Scientific Modelling: An analysis of novices' reasoning during computer-based modelling. International Journal of Science Education 27(14), 1695–1721 (2005)

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

van Joolingen, W.R., de Jong, T., Lazonder, A.W., Savelsbergh, E.R., Manlove, S.: Co-Lab: research and development of an online learning environment for collaborative scientific discovery learning. Computers in Human Behavior 21(4), 671–688 (2005)

van Joolingen, W.R., King, S., de Jong, T.: The SimQuest Authoring System for Simulation-Based Discovery Learning. Paper presented at the Proceedings of the International Conference on Artificial Intelligence and Education, AIED 1997 (1997)

Van Meter, P., Garner, J.: The Promise and Practice of Learner-Generated Drawing: Literature Review and Synthesis. Educational Psychology Review 17(4), 285–325 (2005)

Wilensky, U., Reisman, K.: Thinking Like a Wolf, a Sheep, or a Firefly: Learning Biology Through Constructing and Testing Computational Theories — An Embodied Modeling Approach. Cognition and Instruction 24(2), 171–209 (2006)

Wilensky, U., Resnick, M.: Thinking in Levels: A Dynamic Systems Approach to Making Sense of the World. Journal of Science Education and Technology 8(1), 3–19 (1999)