Towards Supporting Scientific Inquiry in Computer Science Education

Sandra Schulz
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin
saschulz@informatik.hu-berlin.de

Niels Pinkwart
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin
pinkwart@informatik.hu-berlin.de

ABSTRACT

Experimentation is one of the key techniques to gain knowledge in science. Physical computing is increasingly gaining attention in computer science; it shares several features with experimentation so that it seems plausible to build on established models of experimentation for physical computing based computer science education. This paper presents a theoretically derived physical computing model and compares it to established models of scientific inquiry, pointing out common elements such as similar phases, but also potential differences. In a physical computing pilot study, we analyzed student behavior based on the theoretically derived model in order to see if this model – when used to analyze student behavior – is (1) reasonably applicable, and (2) potentially helpful for teachers. The results of the study generally confirm the appropriateness of the model as an analytical lens for describing student activities in physical computing exercises. At the same time, the study results also motivate slight modifications of the model. Finally, the study results may serve as a guide for teachers who want to conduct physical computing lessons.

CCS Concepts

•**Social and professional topics** → **K-12 education;** •**Computer systems organization** → **Embedded and cyber-physical systems;** *Robotics;*

Keywords

Physical computing; experimentation; scientific inquiry; computer science education

1. INTRODUCTION

Physical computing is increasingly used in schools and is becoming more and more a focus of research on computer science education. Physical computing can be characterized by using devices such as robotics, Raspberry Pis and Arduinos in education. As put by Przybylla and Romeike in their definition of physical computing, "Interactive Objects [...] perceive their environment with sensors, which in turn deliver data to be processed by the microcontroller. According to the configuration of the systems these data are processed and passed on to the actuators. In this way, interactive objects communicate with their environment" [22]. Using physical computing in education can improve motivation, in particular for girls [8], and there is initial evidence for positive learning outcomes of physical computing approaches in computer science and other STEM subjects [28].

STEM education has a long tradition of scientific inquiry and inquiry learning as problem solving strategies that are used to gain knowledge in science and to learn epistemological views, scientific reasoning and practical skills [15]. One specific working technique of scientific inquiry is experimentation, which is very popular in science. Experimentation shares many surface features with physical computing. We were thus interested in more deeply investigating if physical computing can indeed be conceived as a working technique for scientific inquiry. In section 2 of this paper, we review the relevant literature on physical computing and elicit a phase model for physical computing processes. We compare this model to established models of scientific inquiry to identify commonalities and differences. Subsequently, we describe a pilot study we conducted. In this study, we analyzed student behavior in a physical computing task based on the theory-derived model. The analysis confirms the general applicability of the model to describe physical computing processes, yet at the same time also gives initial evidence for some refinements and adjustments to the model. The analysis of student behavior in the study (and its relation to the phases in the model) also yields some results that may serve as guidance for computer science teachers who want to introduce physical computing in their classrooms.

2. THEORETICAL BACKGROUND

In this section we first describe the theory of experimentation in natural sciences and selected results in the field of science education. Experimentation is a key component in the general theory of scientific inquiry (SI). The boundary between SI and inquiry learning (IL) will be explained to give an overview independent of domains. Next, we analyze the role of SI specifically for computer science (CS) education and compare this to physical computing. Finally we derive our physical computing (PhC) model from theory and compare our model to an existing model of experimentation in STEM.

WiPSCE '16, October 13-15, 2016, Münster, Germany

DOI: http://dx.doi.org/10.1145/2978249.2978255

2.1 SI and Experimentation in STEM

SI and IL are based on a general theory of problem solving, which is widely accepted in the natural science community [15]. The purpose of SI is to learn about gaining knowledge in science, which includes skills in working techniques and ways of thinking. Different models of SI exist. Bybee describes the core of SI as consisting of the following five phases: observation, hypothesis, interference, test and feedback [2]. He also defines **observation** and **experimentation** as working techniques for SI. **Modeling** is a widespread method for SI [32] and in CS. This paper focuses on experimentation as a working technique. SI is often explained but less explicitly modeled as a cycle. It is obvious that after one pass through the five phases, typically not all relevant knowledge is gained with respect to a problem or phenomenon at hand. In the process, new hypotheses are often generated which need to be investigated. Passing the cycle exactly one time is thus a rather theoretical case.

SI is well researched in natural sciences and includes experimentation as a key element and working technique. Schreiber et al. developed a model of experimentation in science [27]. This model is based on Hammann [7] who derived from Klahr's model [10, p. 21–39] and expanded this model with empirical results. Hammann divided the process into three phases: searching and forming a hypothesis, planning and performing experiments, and analyzing data. Schreiber et al. differentiated this model further, using the categories *preparation*, *performance* and *evaluation*. *Preparation* includes developing a question as a result of observing an unknown phenomena. Clarifying a question could be a starting point if a phenomenon is observed during the first step or if the question for investigation is given. Afterwards expectations about the outcomes need to be formulated, leading to a hypothesis. The shift between the *preparation* phase and the *performance* phase is characterized by a change from an abstract to a real model of an experimental design and an experimental draft. The control of different parameters is important for that. To start the *performance* phase it is necessary to make sure that all devices are available. Then the experimental design should be built and tested. This is an iteration of either the whole phase or sub-phases from the creation of an experimental design to testing this design, until it seems to be accurate. After the planning, the experiment is ran and data is collected and documented. The following part, handling problems and errors, can either belong to the *preparation* or the *evaluation* phase as it may involve producing further data (performance) or processing data (evaluation). In the *evaluation* phase, the data is prepared, for example in table form or graphically. Afterwards the data is processed and analyzed according to the hypothesis which was set up in the beginning. It is important that there is no fixed order within these three phases. Not every sub-phase needs to occur (e.g. existence of errors), sub-phases can be re-ordered, or may be iterated.

Hammann also constructed a model for experimental competences. Indeed, several such models exist, but they often only focus on selected aspects. Schreiber et al. focus on performance, while Mayer concentrates on preparation and building hypotheses. In Mayer's description, performance is a part of practical work and not directly part of the experimentation process.

Patzwaldt et al. validated phases and sub-phases for experimental competences in chemistry [19]. They identified almost all phases from literature, yet the evaluation phase was sometimes shortened or omitted. In a study they divided the task into sub-tasks which correlate to the SI process. Findings are: (1) cognitive processes are parallel to sub-tasks; (2) the phases of preparation, performance and evaluation are iterative, and (3) in performance there are oscillating steps backwards to preparation.

2.2 Scientific Inquiry and Inquiry Learning

SI is considered as a part of general education [2] and a key competence in the sciences [20, p. 31][29, p. 53]. SI is a general theory of gaining knowledge in science independent of domains. It is divided into different phases and sub-phases, which need to be passed in the learning process (see above). The term Inquiry Learning is related to SI, but focuses more on specific educational interventions that do not necessarily have to include all phases of the SI process. Computer simulations are common to support students by giving domain specific feedback to improve inquiry. Related literature can be seen in section 2.3.

There is evidence that students of different ages often have problems acquiring SI competences – which is a key motivation for conducting IL to help learn specifically those competences that students have trouble with. Some of the related research results are presented in the following paragraph. We selected results from literature reviews [6, 33]. Students frequently have difficulties in building hypotheses in general. In particular, this involves forming hypotheses which are testable and drawing right conclusions from results. Combining hypotheses with results is critical if the results contradict student's expectations or experience [3]. Assembling the experimental design to test desired variables is also a problem for many students [12]. Students also often change more than one variable at once [9]. They also frequently choose wrong variables to test [34]. Students are often not aware of how they should do an experiment: During experimentation, they try to reach a certain goal state, not realizing they are no longer testing a hypothesis [25]. The general preparation is often abbreviated and important tasks are skipped [14]. In evaluating data, students have different tendencies when collected data does not fit to their theory (anomalous data). This includes ignoring data, rejecting data or assuming that the theory has changed [4]. It is possible that students sometimes react irrationally when considering data. In general, they have difficulties to distinguish data (or rather evidence) from theory [5]. Students have problems in adapting their theory and build a new one if they have not enough knowledge in this context [24]. In conclusion, these results show that students need support when acquiring SI competence, which motivates IL approaches. Strategies like control of variables are often used as a investigative strategy. Other scaffoldings like choosing one hypothesis from a set of hypotheses have been used to support inquiry [30].

2.3 The Role of Computers in SI

Computers are often used in an IL context. This includes the use of learning tools to support group processes and to gain knowledge by combining modeling and learning by collaborative inquiry. Frequently, the computer based tools provide scaffolding or give feedback to students [21].

Another IL approach involving computers as tools is a creation of a learning environment where simulations can be

modeled or built in a remote laboratory to support collaborative discovery learning in natural sciences [31].
Resnick et al. commented on the role of building and designing own instruments beside observing and measuring in scientific inquiry [23]. They argue that Galileo constructed his own telescope to conduct research and so did other scientists. The benefit of this process is that they understand the functions of the instrument and limits: "By building their own instruments [...] scientists have historically gained deeper insights into nature of the phenomena under investigation" [p.8]. This illustrates a strong connection to scientific inquiry. Building devices should produce transparency for learners. Constructing own devices can thus be an important component for designing experiments. When building their own (specifically also computer based) devices, students may thus become more critical and obtain a better understanding for suitable measures and formulate possible reasons for unexpected or anomalous data – as compared to using off-the-shelf devices. If devices are computer based, the design and implementation necessarily involves CS skills like handling sensors and actuators and programming.
We conclude that very frequently computer based tools play a role in SI and IL processes. Yet, the improvement of CS competences beyond the mere use of existing digital devices is relatively rare. A working technique which improves CS and SI skills would have several benefits – and physical computing seems to be an appropriate choice, as we shall argue in the next sections.

2.4 Identifying the Process of PhC

In literature we find only few explicit descriptions of typical PhC processes. Existing publications show a spectrum of different and very creative projects which can be built with sensors and actuators, with a growing number of educational questions and approaches covered by PhC. Yet, an overall process model of physical computing has not been proposed. One considerable characterization is given by O'Sullivan and Igoe [17, p. xix–xxix, 49–86]. Their starting point is a description of what should happen by the interaction of an arbitrary device with the environment. The focus is on what the person who builds or uses the device can see, hear, or feel when the interaction happens. The interacting person gets in the foreground of the process laid out by O'Sullivan and Igoe. In their approach, the description is formulated in natural language without a specific programming language. The overall process occurs in a loop. In the preparation phase it is necessary to find an approach or have a first idea how to solve a given problem. If it is possible, depending on experience in PhC, problems in later phases should be anticipated at the beginning. At this time it is practical to share problems in a community, according to O'Sullivan and Igoe: There are a lot of existing platforms, e.g. for makers [13], so that it is likely that other users may have had similar problems and solutions. A more general recommendation of O'Sullivan and Igoe is to divide the main task into sub-tasks for switching around if a person is stuck.
After some iterations during *preparation* the *implementation* phase starts. The first step is thinking about specific devices and input, output and processing. At this point, decisions should be taken about which input and output should be used and whether analog or digital is appropriate. For processing it should be clear which process needs to be serial or parallel. After planning in detail, coding starts. Reading sensor values, turning actuators on and off and sending messages to other devices is included in the programming activities. After running the program it is necessary to identify problems, because it is unlikely that software and hardware are running completely correct by the first try. This is well known as debugging in CS. Therefore in every step only one variable should be changed to make sure about the problem. Changes can be done by replacing single pieces of hardware or software to exclude sources of error.

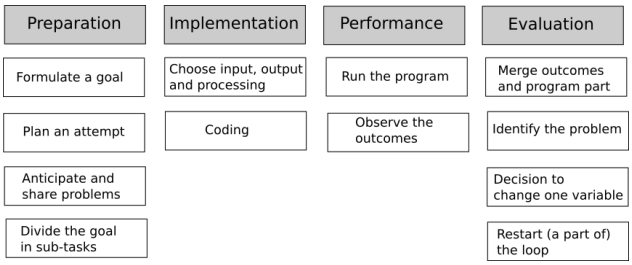

Figure 1: Physical computing phases model derived from literature [17, 16, 1]

Focusing on the PhC process, Banzi states: "Arduino was born to teach Interaction Design, a design discipline that puts prototyping at the centre of its methodology" [1, p. 5]. "Interaction Design" focuses on the interfaces between humans and objects, encouraging iterative working processes. This position also emphasizes the importance of debugging in PhC when the quality of prototypes is increasing. Banzi explained, "classic engineering relies on a strict process for getting from A to B; the Arduino Way delights in the possibility of getting lost on the way and finding C instead" and influences of prototyping, tinkering and patching on the Arduino Way. This idea is more focused on outcomes as a product with clear goal orientation.
Okita choses a more process oriented approach in problem solving with robots [16]. She investigated the so-called recursive feedback which is necessarily provided in robotics activity. This kind of feedback occurs due to the previous formulation of rules on how a robot should act (in a programming environment). After programming, students run the program and observe the outcomes without having influence on their program at this time. In the *performance* phase, students observe outcomes and need to figure out discrepancies between their formal program and the observable outcomes. Therefore it is necessary to find implications of the program code that were not intended. Then they match which part of the program is responsible to solve the problem this way. One important difference to other processes is that there is a relation between student's effort and feedback provided. Okita argues that recursive feedback improves the anticipation of visible outcomes in performance and in this way learning conditions for problem solving. If the students have no ideas about possible outcomes it is very challenging to use the given feedback.
The problem solving context for robotics was provided earlier by Papert, who postulated that design, iterations and testing are important to solve real problems [18].
Based on the above suggestions of typical PhC process elements derived from literature, we have constructed the model shown in Fig. 1. This model includes phases and sub-phases of PhC (while not every sub-phase has to be in-

cluded in every PhC process). During a PhC activity, the process is traversed iteratively.

2.5 Comparison of Theories and Aim of Study

Next we compare our model of PhC developed above to models of experimental competence. Table 1 gives an overview of explicitly formulated phases and sub-phases in experimentation (as formulated by Schreiber et al. [27]) and compares this to the elements of our own PhC model. The absence of sub-phases does certainly not mean that this phase can not be included in the process, but it was not explicitly foreseen by the authors.

The preparation phase is quite similar, most importantly pointing out the idea and forming the hypothesis, but the intention for the process is different. Both set the goal to gain knowledge. In addition in PhC a product is often built to improve daily life or to explore technology.

The idea of sharing problems is widely spread and established in the nature of CS. In PhC often no phenomena are observed. More frequently, the task is an individual choice of a state which is declared as the goal. That is possible in science as well. The division of tasks into sub-tasks in preparation seems to be special for PhC.

Sub-phases of an experimental design, selection and assembling of devices (performance in experimentation) in PhC are included in the implementation phase. In both models there are iterations in preparation and transition to performance (testing experimental design) or rather in preparation and implementation.

Performing experiments, for instance in physics, implies collection and documentation of data. In PhC documentation and collection is usually implicit. It is important to pay attention to outcomes in performance. Handling problems or errors is only done in experimentation.

Preparing and processing data is not very prominent in PhC. The evaluation phase in PhC is short or sometimes even not included at all. After evaluation, transferring results back to the hypothesis is important for gaining knowledge in science. Iterations of the process within sub-phases is included in both models. In PhC it is unlikely to have no repetition of the cycle. In experimentation, an iteration between performance and going back to the hypothesis is often important.

In summary, based on the findings available in literature, PhC and scientific experimentation processes share many elements (but also show some differences) which may warrant interpreting PhC as a CS specific technique of scientific experimentation. Our next goal was to substantiate this by testing the applicability of the process model in a small-scale pilot study.

Table 1: Comparison of models for experimentation according to Schreiber et al. [27] and our present model for PhC (Fig. 1).

Model according to Schreiber et al.		Present work	
Phase	**Experim.**	**PhC**	**Phase**
Prep.	Clarify question/ develop question	Clarify goal	**Prep.**
	Form hypothesis	Plan an attempt	
	Express expectations	Express expectations	
		Divide task	
		Share problems	
Prep./ Perf.	Create experimental design	Choose input, output and processing	**Impl.**
Perf.	Collect devices		
	Assemble experimental setup		
		Coding	**Perf.**
	Perform measurements	Run program	
	Document measurements	Observe outcomes	
Perf./ Eval.	Cope with problems and errors		
Eval.	Prepare data		**Eval.**
	Process data		
	Interpret results		
		Merge outcomes and program	
		Identify problems	
		Change variable	
		Restart loop	

3. PILOT STUDY DESIGN

In order to analyze typical student behavior in a physical computing task (without explicit process guidance), we designed a study which was divided into two main phases as schematically illustrated in Fig. 2. At first the students received an introduction to the LEGO Mindstorms programming environment that uses a block based GUI without programming code included. In every block there are several variables to define actions. For example one "motor block" has options to turn a motor on, off, set time in seconds, set quantity how often wheels rotate or a degree for rotation. This block is able to control two motors at the same time – for two wheels on a vehicle. Using several simple tasks, the students were informed how to handle the programming environment, the hardware and to anticipate the outcomes of their programs. These tasks were simple movements, the use of touch sensors or identifying limits of ultrasonic sensors. They were also introduced to a second tool within this environment. This tool is called "experimentation environment" and can visualize sensor values in graphs. After this first introduction phase, the second main phase consisted of a single task, which provides data for the study. In this phase, students got the task to program a robot that is able to drive around a box (approx. $30cm \times 50cm \times 20cm$) using one ultrasonic sensor. The only demand for the algorithm was that the robot should be capable to drive around every possible box. The students were asked to choose a good starting position of the robot related to the box. They had 70 minutes to solve the task. A worksheet was provided as support (but no explicit instructions on phases they would have to follow). This sheet was divided into four sub-tasks: (1) "How should the robot act to solve the problem? When is this problem solved?", (2) "What should your program do? How is it changing?", (3) "What do you observe during performance?", and (4). "Do you need to make additional changes in order to solve the problem?". Students were asked to fill in the blanks in the worksheet. They only received help from the experimenter in the following three

cases: (A) they explicitly asked for help; (B) there was no interaction between the students or with the technical devices for several minutes; or (C) the students were obviously getting discouraged. Students were not allowed to use the Internet to get help.

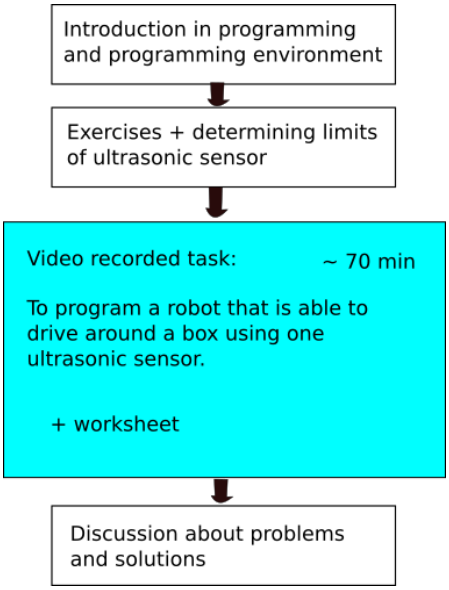

Figure 2: Study design

3.1 Participants

In the spring of 2016, we conducted the study described above with nine participants aged between 14 and 17 years ($m = 15$), six girls and three boys. The students were attending secondary school and participating in a voluntary robotics workshop at Humboldt-Universität zu Berlin. Only one of them had experience in programming LEGO Mindstorm robots. Two not had any computer science lessons in school, five were currently in a optional CS course and two had already passed a CS course successfully. The students were divided into four dyads and one student (the one with experience in robots programming) working individually. To get an idea of his thoughts, he was requested to talk about what he was thinking about. He got help in the same way as the others.

3.2 Methodology of Evaluation

Dyads were video recorded in the working phase. These videos were transcribed, focusing on verbal interaction within the groups and also on important changes in the implementation and on the outcomes in performance. The transcript was coded using the above described phases as codes: preparation, implementation, performance and evaluation. In addition we coded other noteworthy aspects we could not find in computer science literature or in other STEM research. At first we did a qualitative analysis which was quantified afterwards. To back our coding we calculated an inter-coder-reliability using Cohen's Kappa. Therefore a second coder observed 10% of the transcripts from three groups. We reached $\kappa = 0.74$ for implementation, $\kappa = 0.68$ for hypothesis, $\kappa = 0.86$ for performance and $\kappa = 0.68$ for evaluation. Compared to Landis and Koch the consistency is substantial to almost perfect [11]. In the qualitative analysis we defined which action was included in which phases to characterize the sub-phases for physical computing that were derived from literature. Then we observed the changes between phases and the order of the phases. Therefore we investigated which phases were often connected or what were possible ways to pass or bypass the cycle. Then we quantified the phases by a counting of coded phases for the main task to interpret how students worked.
Qualitative analysis was conducted with the MaxQDA software, where the phases preparation, implementation, performance and evaluation were coded as follows:

- **Preparation**: all thoughts about general functions and getting an idea about a solution without thinking about concrete implementation. Anticipation of problems is included here too.
- **Implementation**: starting an interaction with the programming environment, like observing available blocks or with the robot (for instance, checking inputs and outputs). Coding and thinking about producing or changing code. Reconstructing hardware components like position of sensors or wires to improve functionality.
- **Performance**: running a program or parts of it. One successful run (starting program and robot is moving) is counted as one phase. Use of experimental environment to check sensor data is included here too.
- **Evaluation**: direct reaction of students while or after observing outcomes of performance. Suggesting concrete interventions like changing any values or hardware after performance or refining hypothesis. Description, interpretation or evaluation of outcomes. If a completely new hypothesis is generated or a task is divided into sub-tasks then this evaluation phase leads to a new preparation phase.
- **Problems and errors**: changing hardware or software components suspected as the source of error. Sensor errors or problems handling the sensor because of construction (cable in front of the sensor) or inattention of students (hand in front of the sensor).

4. RESULTS

We next first describe the general observations during the study, followed by the results of the qualitative and quantitative phase based analysis.

4.1 General Observations

In the first two groups, no intervention by the experimenter was conducted. Students only got help in cases stated above. The evaluation phase of these groups was very short or not existent, though. As a result, we decided to inquire to get more information about students' observations and the reasons of their ensuing decisions. So, in the other three groups, the experimenter inquired what problems were recognized in performance in case the group did not start an evaluation phase on their own.
With this exception, the study proceeded as planned. All students were interested in the tasks and tried to solve them. Few technical problems occurred when running the program,

because there were several programs with the same name on one robot. After the first pass of the loop almost no dyad ever looked at the worksheet again and thus forgot about filling the blanks. The students asked for help only rarely. They tried to solve their problems without help and asked the experimenter only if they suspected a technical problem might have occurred. The one student without partner asked the teacher two times, when he felt completely at loss about how to go on. The students had problems finding the source of error when changes in the program did not influence the outcomes. Sometimes the wrong program was executed. In the first 35 minutes there were few interventions by the experimenter. But he needed to intervene several times in the last 35 minutes. Three of five groups got discouraged and needed help to find a new way to solve their problems. The students needed support in drawing conclusions from the observed outcomes and in finding concrete interventions to solve their problem.

4.2 Qualitative Phase Analysis

To examine how the robot needs to act in order to drive around the box, four out of the five groups came up with an idea first. Subsequently, they generated a hypothesis or specifically an algorithm based on this idea. During this process they made suggestions and some constructed a picture or pointed out their idea with gesture or nearby material. At first, building a hypothesis often took a long time. Within dyads they discussed several times until both partners were sure that they had the same concept of a solution and the functionality of the robot. They anticipated diverse potential problems. One exemplary problem was their uncertainty about the limits of the sensor and the resulting starting point of the robot – either in front or parallel to the box. The only possibility for sharing problems was asking the experimenter, which no group did during the first **preparation phase**. Within this phase they often switched from forming a hypothesis by anticipating a hidden problem to going back building a new hypothesis or rethink other sub-phases. In the first run all dyads tried to solve the problem with one single program without dividing the problem into sub-tasks. They revisited the preparation phase several times, but only during the first half of the whole session. When the groups were sure that their solution would be realizable with the given code boxes provided by the program or their theoretical part was finished, they approached the **implementation phase**. Only one group started without planning of their approach and tried to develop an idea while programming. Most groups thought deeper about the problem and developed a solution iteratively by improving their code. They thought about concrete input and output, and implemented these in their program. Decisions about serial and parallel processes needed to be made. Three groups had some problems to handle the programming environment and predefined variables, e.g. "threshold value" which defines in which distance to the box the robot should react. Group D did not understand the function of loops and if-else-clauses and mixed these up in their implementation several times. The same students had difficulties to understand the functionality of the ultrasonic sensor. Finally they concluded falsely that this sensor had a kind of searching function by using given parts of the program. Group D suggested "we need to choose a starting point where the robot sees the box, then he is aware of the box position for the whole time". Some groups built a program including a lot of commands. This made it impossible for them to check the correctness of the program or to find possible problems within the code. When students decided to run the program, this was either to pass the cycle several times to get feedback or because of their increasing tiredness. Then they positioned the robot to start. **Performance** is started by running the program. Several groups chose a wrong program and the robot did not move. This case was not counted for the performance phase. During the performance the students could not interact (e.g. discuss or evaluate). They observed the outcomes and went back to their workplace to discuss the results. Group A, B, C and D ran the program several times without leaving the phase because they started no evaluation. In this case the starting position was sometimes changed by the students. Group C executed the program very often in different scenarios to localize the existing problem. It was not clear in all cases if the students were aware of what their program was doing. They changed the driving direction of the robot without changing the program although mostly the program was designed for one specific direction. Only group C changed the real world test environment and thereby shortened their way. They did not use the box when it was not necessary. Instead they tested simple functions at a wall near their workplace or kept the robot in the air while testing the orders to drive forward and backward without leaving their workplace. They observed only the direction the wheels turned into. All groups interacted with the robot during performance. If the robot was not able to drive around a corner students dragged the robot backwards to retry. When the program was running and the students instantaneously decided to move the robot closer to the box they did not start the program again. Only one group restarted the program for every test. After finishing performance the students evaluated their outcomes and sometimes formed a new hypothesis or went back to the implementation phase. The **evaluation phase** began directly after performance. In group A, D and E the phase was not always existent, very short (like "no") or there was no visible connection to the outcomes. Most interventions which were planned during the evaluation included the manipulation of values e.g. to improve the accuracy of a bend. All groups checked their program, remembered what they saw and tried to find out which output was produced by which part of their program. Some groups had many problems while identifying problems within the outcomes, others saw many possible approaches to intervene after the first run.

4.3 Quantitative Phase Analysis

We applied an analysis on the qualitative data. The coded phases were counted (as shown in Fig. 3). All five groups are illustrated separately. The frequencies of the single phases are: preparation $(6,3\%)$, implementation $(28,6\%)$, performance $(37,4\%)$ and evaluation $(27,7\%)$. Most groups were in the preparation phase 4 to 6 times and formed multiple hypotheses. Implementation happened 17 to 23 times, which is considerably more than preparation. Four out of five groups did more performance than implementation. Only one group did one more implementation than performance. Eye-catching is group C, which performed 74 times. Other groups passed this phase 16 to 28 times. Evaluation happened 16 to 47 times. Quantitatively groups A, B, D and E were quite similar. It seems that D and E worked a bit

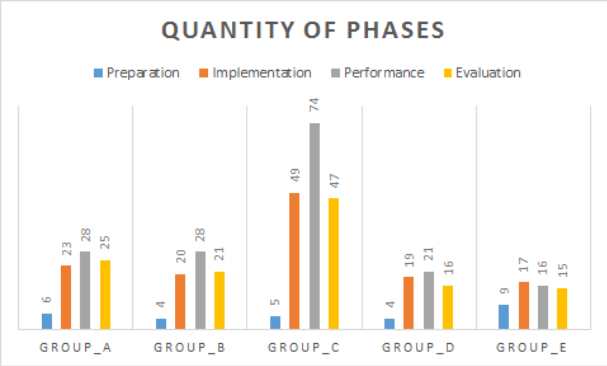

Figure 3: Quantitative analysis

slower because of fewer passes of the cycle in the same time. Group C worked quite fast and tested a lot, sometimes they only changed values to perform better. There seems to be a correlation between implementation and evaluation. Possibly was clear to students that they needed to evaluate their code and rework it to have better outcomes.

4.4 Adjusted PhC Model

The observations in our study confirm most of the elements and phases of the initial PhC model (Fig. 1). Yet, our results also give some initial evidence for suggesting adjustments and refinements of the model. We did find most of the initially proposed PhC phases and also other anticipated phases. After gathering data some phases were reordered. In addition to reordering some sub-phases, we expanded the initial model with the phases that we observed in our study but that were not contained in the initial model. The slightly revised model, shown in Fig. 4, is divided into four categories: preparation, implementation, performance and evaluation. The highlighted sub-phases have been added to the model based on the empirical insights. We found supportive

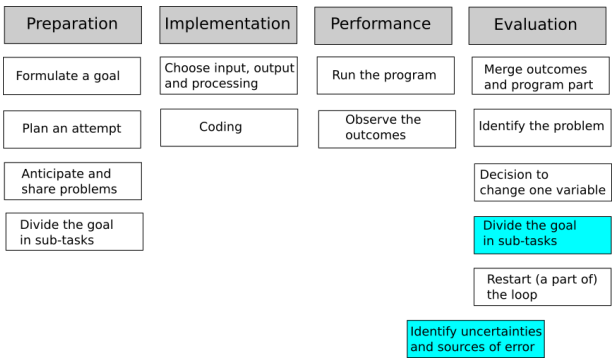

Figure 4: Adjusted physical computing model

evidence for all of the phases described in the literature with the one exception of sharing problems – this was not possible for the participants in the study design that we employed. Formulating a goal or an algorithm and finding a way how to construct a solution was observable. In the implementation phase, the solution was defined and decisions about input, output and processing were made. Running the program and observing the output are fundamental sub-phases for PhC. Afterwards the observed outcomes needed to be merged with program parts and problems were identified. Therefore it was necessary to decide which variable should be changed in a next iteration of the cycle. The division into sub-tasks was theoretically classified as preparation, but in reality happened only during the evaluation phase since the outcomes were discussed in conjunction with planning next steps. After the evaluation the students went back to restart the cycle using different new entry points. A look at these new entry points after the evaluation phase is interesting. Next, we will thus concentrate on **iterations in the PhC process**. Three different cases for outcomes of activities in the evaluation phase are possible (as shown in Fig. 5). One case (which is very unlikely) is getting the solution of a task correct after one pass through all phases. The usual case is that students have no complete solution for the task and need to go back in other phases: (1) going back to the performance phase to see outcomes again or to test if the problem is reproduced if the starting point for the machine is changed. (2) Changing the implementation because it was observable that some smaller changes of values could be necessary or (because of no other ideas) students tested different values and code parts on a seemingly random base. (3) A new hypothesis is built based on the outcomes. Case three is characterized by not focusing on the original task again at the beginning of the second cycle. Here the problem is divided into sub-tasks which need to be solved for attaining the goal. This could be a program which stops if the robot is in a specific distance to the box. This approach seemed to be usual for many PhC activities, though not reported on in the general IL literature oftentimes. We found

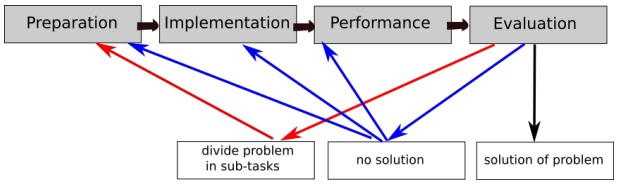

Figure 5: Observed steps after evaluation

(measurement) uncertainties and sources of error. In the model, we included these in the transition between performance and evaluation because the observations are made in the performance phase and mostly interpreted in the evaluation phase. We modeled this as a separate sub-phase of PhC because it is an important part which has big impact on the PhC process, learning about situations and practical use as well. As with the other process models, the revised model does not prescribe a strict order. Most sub-phases will likely be included in every PhC process, but not necessarily all sub-phases will be, and possibly the ordering may change.

5. DISCUSSION AND CONCLUSION

In this paper, we have presented a PhC process model elicited from literature. We found a lot of commonalities in the structure and contents of this model when comparing it to established models of scientific experimentation. We therefore have initial evidence that PhC can be conceived as a working technique that is related – though not entirely the same – to experimentation in SI.

In a small-scale pilot study, we analyzed the appropriateness

of the theoretically derived PhC process model for describing student activities. On a theoretical level, the results of the study confirm the general appropriateness of the model, but also motivate some possible refinements and adjustments of the model. In order to empirically validate the PhC process model, further (and more controlled) research studies will be needed.

On a practical level, the results of the study may be relevant for teachers of physical computing classes. One important observation we made relates to student's handling of anomalous data. If the results are not compatible with their hypothesis, students oftentimes ignored the data. Sometimes they were apparently biased based on their everyday knowledge and trusted their intuition more than the data which was produced by themselves. It was difficult for some students to implement their ideas and make a hypothesis testable. While programming, they changed diverse variables which made a new evaluation difficult as well. Most of the problems we identified were related to observing outcomes and drawing a conclusion. Students needed several runs of the program to identify a problem and the quality of conclusion was often poor. Finding the sources of errors was difficult for the students.

To address some these issues, it may be helpful for teachers to make the process model of PhC explicit to students, thereby encouraging them to reflect on the different phases. Specifically, methods for systematically testing a variable and evaluate changes carefully (to avoid random testing) could thus be learned, thereby also supporting the acquisition of competences related to evaluation phases in the process model.

In our future work, we intend to conduct studies which employ more explicit forms of process guidance. This can potentially take many forms such as different task assignments, changes to the worksheet, modifications to the role of the teacher/experimenter, or even process guidance provided by the computer that the student is using. In our investigations, we will likely focus on the evaluation phase, given its importance in the process and the difficulties that students encountered in the phase in our pilot study. We expect to gain further insights for the educational design of physical computing lessons from these studies. We also intend to more deeply investigate the connection between CS and STEM with PhC. SI literature differentiates between experimentation in natural sciences and engineering [26]. This difference may be interesting when further examining PhC in its relation to SI, especially since CS shares properties of engineering with properties of natural sciences.

6. REFERENCES

[1] M. Banzi. *Getting started with Arduino*. O'Reilly Media, Inc., Sebastopol, 2011.

[2] R. W. Bybee. Scientific inquiry, student learning, and the science curriculum. *Learning science and the science of learning*, pages 25–35, 2002.

[3] C. A. Chinn and W. F. Brewer. The role of anomalous data in knowledge acquisition: A theoretical framework and implications for science instruction. *Review of educational research*, 63(1):1–49, 1993.

[4] C. A. Chinn and W. F. Brewer. An empirical test of a taxonomy of responses to anomalous data in science. *Journal of Research in Science teaching*, 35(6):623–654, 1998.

[5] C. A. Chinn and W. F. Brewer. Models of data: A theory of how people evaluate data. *Cognition and Instruction*, 19(3):323–393, 2001.

[6] T. De Jong. Technological advances in inquiry learning. *Science*, pages 532–533, 2006.

[7] M. Hammann. Kompetenzentwicklungsmodelle Merkmale und ihre Bedeutung - dargestellt anhand von Kompetenzen beim Experimentieren. *Der mathematische und naturwissenschaftliche Unterricht*, pages 196–203, 2004.

[8] F. Kaloti-Hallak, M. Armoni, and M. M. Ben-Ari. Students' attitudes and motivation during robotics activities. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, pages 102–110. ACM, 2015.

[9] A. Keselman. Supporting inquiry learning by promoting normative understanding of multivariable causality. *Journal of Research in Science Teaching*, 40(9):898–921, 2003.

[10] D. Klahr. *Exploring science: The Cognition and Development of Discovery Processes*. MIT Press, 2000.

[11] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[12] A. E. Lawson. Sound and faulty arguments generated by preservice biology teachers when testing hypotheses involving unobservable entities*. *Journal of Research in Science Teaching*, 39(3):237–252, 2002.

[13] Maker Media, Inc. Makerspace. https://makerspace.com/, last accessed 2016-08-29.

[14] S. Manlove, A. W. Lazonder, and T. D. Jong. Regulative support for collaborative scientific inquiry learning. *Journal of Computer Assisted Learning*, 22(2):87–98, 2006.

[15] J. Mayer. Erkenntnisgewinnung als wissenschaftliches Problemlösen. In *Theorien in der biologiedidaktischen Forschung*, pages 177–186. Springer, 2007.

[16] S. Y. Okita. The relative merits of transparency: Investigating situations that support the use of robotics in developing student learning adaptability across virtual and physical computing platforms. *British Journal of Educational Technology*, 45(5):844–862, 2014.

[17] D. O'Sullivan and T. Igoe. *Physical computing: sensing and controlling the physical world with computers*. Course Technology Press, Mason, 2004.

[18] S. Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, 1980.

[19] K. Patzwaldt, M. Kambach, A. Upmeier zu Belzen, and R. Tiemann. Experimentierkompetenz erfassen: Zur Entwicklung und Auswertung von Experimentieraufgaben. In *Naturwissenschaftliche Bildung zwischen Science- und Fachunterricht. Gesellschaft für Didaktik der Chemie und Physik*, Jahrestagung in München 2013, pages 183–185. IPN, 2014.

[20] J. W. Pellegrino, M. R. Wilson, J. A. Koenig, A. S. Beatty, et al. *Developing assessments for the next generation science standards*. National Academies

Press, Washington, 2014.

[21] N. Pinkwart, A. Harrer, and M. Kuhn. Process support for collaborative inquiry learning. *Research and Practice in Technology Enhanced Learning*, 5(3):185–203, 2010.

[22] M. Przybylla and R. Romeike. Overcoming issues with students' perceptions of informatics in everyday life and education with physical computing. In *Informatics in School: Situation, Evolution and Perspectives*, ISSEP 2014, pages 9–20, 2014.

[23] M. Resnick, R. Berg, and M. Eisenberg. Beyond black boxes: Bringing transparency and aesthetics back to scientific investigation. *The Journal of the Learning Sciences*, 9(1):7–30, 2000.

[24] T. Ruffman, J. Perner, D. R. Olson, and M. Doherty. Reflecting on scientific thinking: Children's understanding of the hypothesis-evidence relation. *Child Development*, 64(6):1617–1636, 1993.

[25] L. Schauble, R. Glaser, R. A. Duschl, S. Schulze, and J. John. Students' understanding of the objectives and procedures of experimentation in the science classroom. *The journal of the Learning Sciences*, 4(2):131–166, 1995.

[26] L. Schauble, L. E. Klopfer, and K. Raghavan. Students' transition from an engineering model to a science model of experimentation. *Journal of research in science teaching*, 28(9):859–882, 1991.

[27] N. Schreiber, H. Theyßen, and H. Schecker. Experimentelle Kompetenz messen?! *PhyDid A-Physik und Didaktik in Schule und Hochschule*, 3(8):092–101, 2009.

[28] S. Schulz and N. Pinkwart. Physical computing in stem education. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, WiPSCE '15, pages 134–135, New York, NY, USA, 2015. ACM.

[29] B. Trilling and C. Fadel. *21st century skills: Learning for life in our times*. John Wiley & Sons, San Francisco, 2009.

[30] W. R. van Joolingen and T. de Jong. Exploring a domain with a computer simulation: Traversing variable and relation space with the help of a hypothesis scratchpad. In *Simulation-based experiential learning*, pages 191–206. Springer, 1993.

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

[32] B. Y. White and J. R. Frederiksen. Inquiry, modeling, and metacognition: Making science accessible to all students. *Cognition and instruction*, 16(1):3–118, 1998.

[33] C. Zimmerman. The development of scientific thinking skills in elementary and middle school. *Developmental Review*, 27(2):172–223, 2007.

[34] C. Zimmerman and R. Glaser. Testing positive versus negative claims: A preliminary investigation of the role of cover story on the assessment of experimental design skills. *CSE Technical Report 554*, 2001.