Conducting a Social Constructivist Epistemology for Students of Computing Disciplines

Brennen Frisque and Ankur Chattopadhyay
Department of Information and Computing Sciences (ICS)
University of Wisconsin - Green Bay (UWGB)
Green Bay, WI 54311
frisbj21@uwgb.edu and chattopa@uwgb.edu

Abstract - **There have been some social constructivist learning approaches used in computing sciences (CS) education, but none of them has actually validated the student reflection process, which is a critical component of the action learning associated with them. The step of reflection within the social constructivist pedagogy has a crucial role in building the learner's mental model and in refining the existing mental model. This paper presents a novel research case study on the employment of an innovative process of student reflection and analysis. This innovative process promotes a unique, non-traditional approach of employing a social interaction based dialogue of inquiry that helps engage each learner in a self-reflective assessment prior to the action learning. The process is in line with the social constructivist model that is based upon inquiry based knowledge building. It helps students debug their conceptual understandings, which make up their respective cognitive models. It also enables them to identify problems or limitations within that mental model before the actual knowledge assimilation or accommodation takes place. The presented case study strives to address a key issue of the social constructivist model i.e. how to ensure that students successfully self-reflect upon what they are learning. In order to so, this research performs a comparative analysis between the results obtained through the presented social constructivist self-reflection assessment and the outcomes of a software based traditional assessment. This paper discusses and analyzes learning analytics data collected over several semesters from multiple instances of social constructivist epistemology conducted for students of two different computer programming classes at the CS1 and CS2 levels respectively. The main contributions of this research experiment are evaluations of the process of student reflection, validations of student assumptions and presentation of an authentic action-learning problem for students in the form of their conceptual gaps. The experimental findings represent evidences of successful validation of student beliefs as well as knowledge discovery process though an epistemic investigation.**

Keywords - social constructivism; epistemology; computing; reflection analysis; conceptual gaps; inquiry; action learning; assessment; validation

I. INTRODUCTION

A. Social Constructivism in Computing Education

Traditional computing technology based assessment procedures [1, 2, 4, 8, 9, 10, 11, 12, 14, 16] have been typically used in computing sciences (CS) education to gauge the cognitive learnings of students and probe their understanding levels. Nevertheless, there have been a few non-conventional constructivist pedagogies and approaches proposed as well [1, 9, 11]. Social constructivism, as rooted in the discipline of philosophy and reasoning, refers to the phenomenon by which learner knowledge is gained and shaped through social interactions and peer exchanges. Unlike traditional evaluation based assessment methods (like exams, test artifacts), the social constructivist assessment techniques involve direct feedback from a subject in an interactive dialogue based environment, where future actions are formulated in response to the nature and sentiment of the feedback. According to the social constructivist model [2, 4, 8, 10, 12, 14], building upon the prior knowledge base and experience accounts for the process of assimilation and accommodation related to the computing student's cognitive model. What makes each learner based social constructivism distinct is the subjective experience (or phenomenology) through which the learner constructs the mental model. The level of assumptions made by computing students represent their epistemic beliefs [4, 6, 7, 8, 15] that can be analyzed at an individual level with a non-traditional, social constructivist approach involving an action learning cycle [1, 4, 8, 10]. A social constructivist assessment technique can provide information on the student's overall knowledge base and learning process.

B. Action Learning in Constructivist Computing Education

One critical issue that remains is the evaluation of the student reflection process, which constitutes a vital part of the action learning cycle (as seen in Figure 1) and which has not been analyzed by any of the previous social constructivism based research in computing education. This paper describes a unique research experiment that is conducted based upon social constructivist epistemology [2, 4, 8] within the computing discipline in order to validate the conceptual beliefs of the students as well as to ensure that students are actually reflecting upon their learnings. Literature in computing educational research suggests that the computing discipline faces a pedagogical shift towards a more socially active learning model, which allows students to derive their own cognitive levels through action learning [1, 4, 8, 10]. This action learning cycle represents the foundational basis for other constructivist approaches in computing education,

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

including the model of social constructivist epistemology as presented in this paper. It allows a student to make immediate as well as extended learning actions during the course of a semester, and then reflect upon as well as self-assess their learning process. This form of comprehension based action learning is valuable because it provides the means by which a computing student can initiate a self-reflective learning action. It is in this context that this paper shows that student epistemology can be fostered in a knowledge-building computing environment by performing a comparative analysis between a socially interactive process of reflection that determines formed conceptual notions, and a software tool based technical assessment, which probes conceptual understandings and brings out weaknesses. The design of this model case study enables validation of a learner's socially constructed beliefs or opinions with the reports generated by a traditional software assessment.

The research discussed in this paper is intended for the academic community in computing, showing how students can successfully achieve a unique level of comprehension and validate their assumptions in a learning environment based upon social constructivism. The presented case study can be potentially meaningful and insightful for learning in computing disciplines at the CS1 and CS2 levels, where students have to build upon their foundational understandings of prerequisite topics. Additionally, making students deal with their own conceptual gaps as part of the social constructivist action learning cycle is a very relevant and authentic recipe for challenging them cognitively [1, 4, 10].

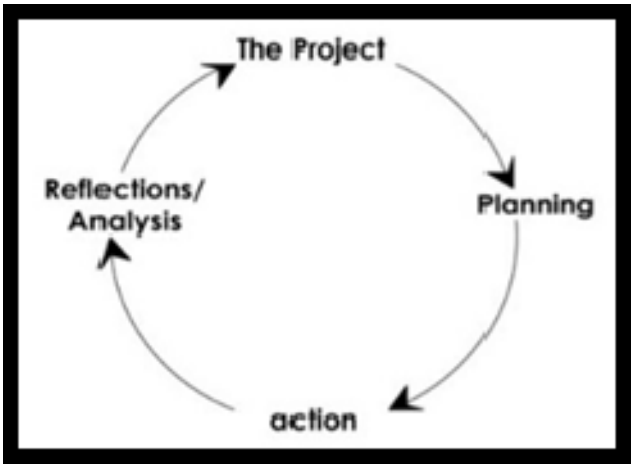

Fig. 1. Action Learning Cycle [1]

C. Social Constructivist Student Epistemology Process

The constructivist approach in teaching spans throughout many non-computing disciplines [3, 6, 7, 13, 15, 17] along with instances of epistemology experiments [6, 7, 15], where the process has been valued as a contributing asset that provides insight regarding the effectiveness of knowledge refinement and creation processes. However, there are no prior instances of social constructivist epistemology driven research involving an individual self-reflective student assessment action learning cycle in computing disciplines.

The research case study described and explained in this paper is meant for the computing academia in an effort to be helpful towards determining the efficiency of a student reflection and analysis in a social constructivist learning process. Specifically, a method and assessment model is implemented to study the validation process of student's epistemic beliefs revealed through multiple steps of interactive dialogue based self-reflection, as part of the social constructivist assessment process. Through a self-driven action learning cycle, students can reflect upon their respective epistemic beliefs and validate them to derive better understanding. These epistemic beliefs provide justification and rationale to an individual and the process of self-reflection allows the learner to define his/her own progress. The goal of the discussed research is to aid students in the process of cognitive debugging leading to identification of conceptual gaps and building of a knowledge foundation by which students can derive comprehensive learning and improve upon their conceptual shortcomings.

The presented methodology allows students to get involved in an action learning mechanism based upon the conceptual gaps confirmed through the mentioned social construction based epistemology process (as illustrated in Figure 2). This research project is intended to translate into a process of knowledge discovery and knowledge creation for deeper understanding, critical thinking and conceptual synthesis. Thus, this paper is motivated by the need of determining whether computing students can reflect successfully upon their learned concepts before they actually initiate the process of building upon their prior knowledge in a socially constructive learning environment. The overall research is also an attempt towards mitigating the levels of attrition within the computing field and improve student retention plus success by advancing students in their holistic knowledge levels.

II. METHODOLOGY

This research focuses particularly on students at the CS1 and CS2 levels in the UWGB ICS department. Students, who were part of this case study, were taken from two courses: Advanced Software Design (CS 316) and Advanced Object Oriented Design (CS 371) at the CS1 and CS2 levels respectively. CS 316 is a 300 level programming course, which covers many of the advanced fundamental programming topics in Java. The students in this course are taught to use basic data types, programming structures, and program design. CS 371 is a 300 level advanced course which continues to develop the coding concepts and programming knowledge, but focuses on more primitive techniques of data manipulation while emphasizing the importance of data structure. The conducted research study consisted of three main stages (or phases), as every class student was encouraged to embrace the process of epistemology, which has been modelled as per Figure 2 and described next.

A. Student Self-Reflection Process Instance

The first phase of the process was to gather a baseline analysis of self- reflection data for each class student. Using a social constructivist approach, students were involved in a user-friendly inquiry driven questionnaire that enabled the learner to declare the individual strengths and weaknesses with the pre-requisite concepts of the course. Additionally, as part of the social interaction based questionnaire, students were given the freedom of expressing individual opinions and revealing their epistemic beliefs about their confidence on certain topics or curricular concepts. Through this unique social constructivist dialogue based question-answer exercise, a portfolio was made for each participating class student indicating his or her strengths and weaknesses.

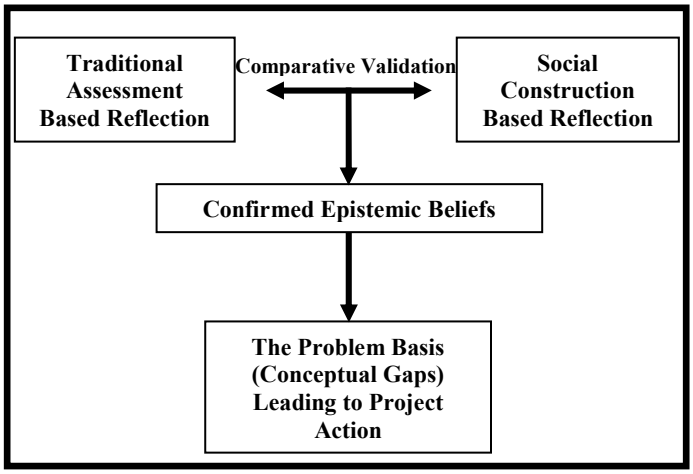

Fig. 2 Proposed Model for Process of Conducting Epistemology

B. Traditional Software Assessment Process

The second phase of the process consisted of a traditional learner skill set assessment driven by a software tool. Traditional computing technology based assessment procedures [1, 2, 4, 8, 9, 10, 11, 12, 14, 16] have been typically used in computing education to determine the cognitive learnings of students and figure out their individual understandings. This research utilized a quiz driven software tool that was specifically developed in-house at UWGB ICS for the traditional individual assessment of the students based on current topics taught in the CS1 and CS2 level courses. The tool used for this assessment is called the GAHPFinder, where GAHP stands for Gray Areas, Hazy Patches, and is intended to symbolize areas of programming where students may need additional study.

This GAHPFinder software tool serves as a technical analyst by assisting in identification of the weak and strong knowledge areas within the curriculum. The GAHPFinder based assessment in the second phase brought out details of the student's performance, and reported the conceptual gaps of each student that needed improvement. Examples of questions that students usually encounter during the use of GAHPFinder are provided in Figures 3, 4, 5, 6, 7, 8 and 9. This is followed by some screenshot images of the actual software assessment tool, as seen in Figures 10, 11, 12 and 13.

The completion of the second stage of this process allowed students to have a holistic view of their subject comprehension, including compilation of their socially constructed self-reflections in the first phase with the results of the traditional assessment tool. This comparative analysis helped to affect a genuine determination of every student's true knowledge elements and a validation of the student's epistemic beliefs. The courses that implemented this study model eventually provided a class project or case study through which the students got the opportunity to address their conceptual gaps collaboratively in a team activity as per the social constructivist model. Therefore, in the process, the students actually got the chance to confirm their cognitive holes and to make those holes the basis of their action learning problem.

C. Survey Phase - Reflective Inquiry

The last phase of this research process comprised of a final inquiry based reflection analysis. The students at the end of the course were subjected to a survey that collected feedback about the use of the traditional software driven assessment in the second phase as well as the correlation of the student's interactive reflection process in the first phase with the results of the software assessment. This final step of the process was vital to the drawing of the experimental data about the conducted student epistemology process and offered a means by which students could individually reflect upon the entire assessment process (steps one through three).

The last phase also helped to gather data about the accuracy, efficiency, and effectiveness of the overall epistemic investigation. This aids in completing the action learning cycle through the approach of reflective analysis. The students use this information to form a basis from which they will focus their studies in future courses and projects. The epistemology based analysis process will be instrumental towards debugging and further probing the cognitive models of the individual students. This epistemic model of reflection, assessment and validation has yielded promising results through class experiments over a period of two years in the UWGB ICS program, and has helped many students evolve their weaknesses into strengths. Some of the sample questions posed in the survey are listed in Figure 14.

The 75 learners, who took part in this research case study over multiple semesters, were students in the two previously mentioned courses at the CS1 and CS2 levels. They were put through the described process of epistemic investigation in an effort to assess their individual levels of comprehension in regard to the pre-requisite class concepts. For this case study, the investigative analysis was based upon the student skills, concepts and ability to code in the Java programming language. Each and every student was asked to specify and justify their areas of strengths and weaknesses in the reflection and analysis step. The overall investigation clearly pointed to the Java programming concepts that a student was comfortable with as well as the areas that the student lacked understanding in. The overall information collected through the conducted experimental case study will potentially help the academic community in computing by potentially educating them about

Which of the following if statements includes a condition that evaluates to true?

A. if (attendance == "true") { . . . }
B. if (attendance) { . . . }
C. if (failed) { . . . }
D. if (attendance == failed) { . . . }

Fig. 3. Example of a Conditional Question (Adapted From [5])

Which of the following statements about helper classes is true?

A. Helper classes must be inner classes.
B. Helper classes must implement interfaces.
C. Helper classes help reduce coupling.
D. Helper classes cannot contain instance variables.

Fig 4. Example of an Interface Question (Adapted From [5])

Consider the recursive method shown below:

```
public static int strangeCalc(int bottom, int top)
{
  if (bottom > top)
     {
        return -1;
     }
  else if (bottom == top)
     {
        return 1;
     }
  else
     {
        return bottom * strangeCalc(bottom + 1, top);
     }
}
```

What value will be returned with a call to strangeCalc(2,3)?

A. 24
B. 2
C. 1
D. -1

Fig. 5. Example of a Recursion Question (Adapted From [5])

What is the result of executing this code snippet?

```
int[] marks = { 90, 45, 67 };
for (int i = 0; i <= 3; i++)
  {
     System.out.println(marks[i]);
  }
```

A. The code snippet does not give any output.
B. The code snippet displays all the marks stored in the array without any redundancy.
C. The code snippet causes a bounds error.
D. The code snippet executes an infinite loop.

Fig. 6. Example of an Array Question (Adapted From [5])

Assuming that inputFile is a Scanner object used to read words from a text file, select an expression to complete the following code segment, which counts the number of words in the input file.

```
int count = 0;
while (inputFile.hasNext())
{
   String word = _____;
   count++;
 }
System.out.println(count);
```

A. inputFile.next()
B. inputFile.nextInt()
C. inputFile.nextDouble()
D. inputFile.nextString()

Fig. 7 Example of a File Reading Question (Adapted From [5])

Consider the following code snippet:

```
public class Cycle extends Vehicle
{
  private String model;
  ...
  public Cycle(int numberAxles, String modelName)
  {
     super(numberAxles);
     model = modelName;
   }
}
```

What does this code do?

A. It invokes the constructor of the Vehicle class from within the constructor of the Cycle class.
B. It invokes the constructor of the Cycle class from within the constructor of the Vehicle class.
C. It invokes a private method of the Vehicle class from within a method of the Cycle class.
D. This code will not compile.

Fig 8. Example of an Inheritance Question (Adapted From [5])

What is the first and last value of i to be displayed by the following code snippet?

```
int n = 20;
for (int i = 0; i <= n; i++)
{
  for (int j = 0; j <= i; j++)
  {
    System.out.println("" + i);
  }
}
```

A. 0 and 20
B. 1 and 20
C. 0 and 19
D. 1 and 19

Fig. 9. Example of a Loop Question (Adapted From [5])

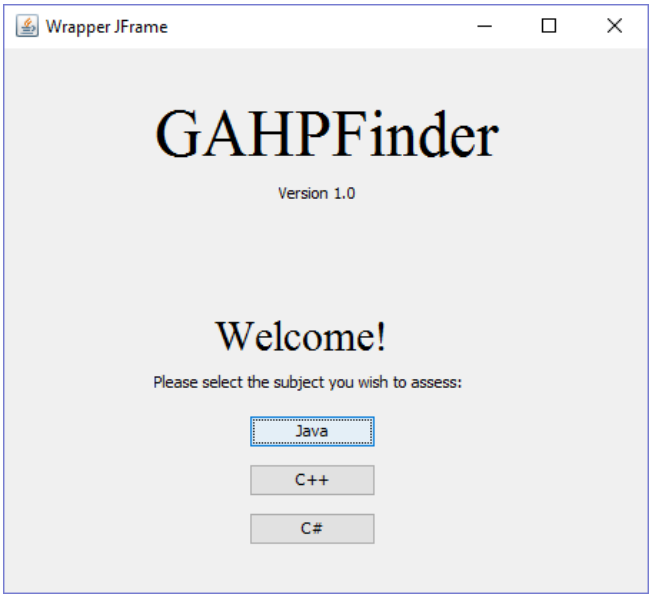

Fig. 10. Screenshot Image 1 of GAHPFinder Software

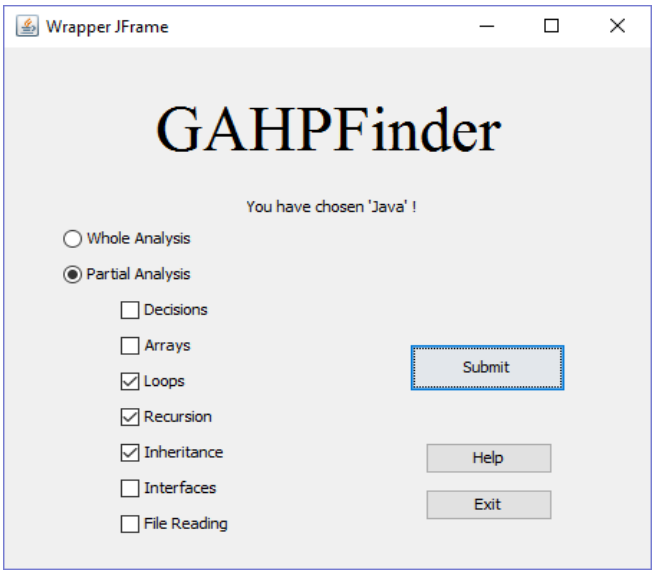

Fig. 11. Screenshot Image 2 of GAHPFinder Software

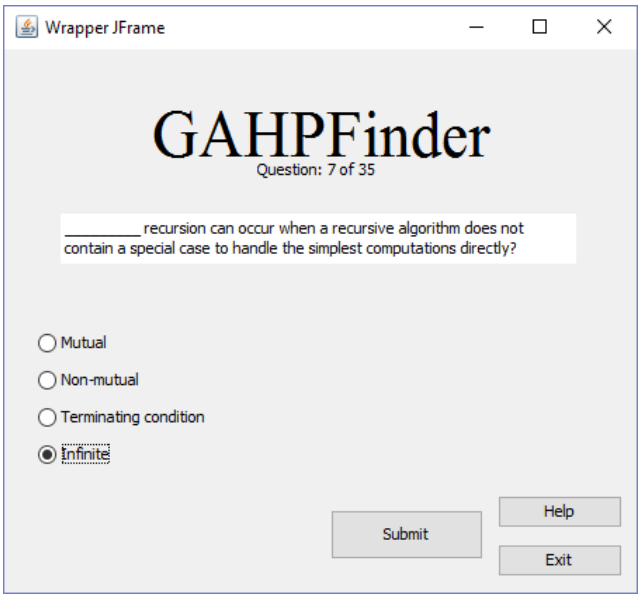

Fig. 12. Screenshot Image 3 of GAHPFinder Software

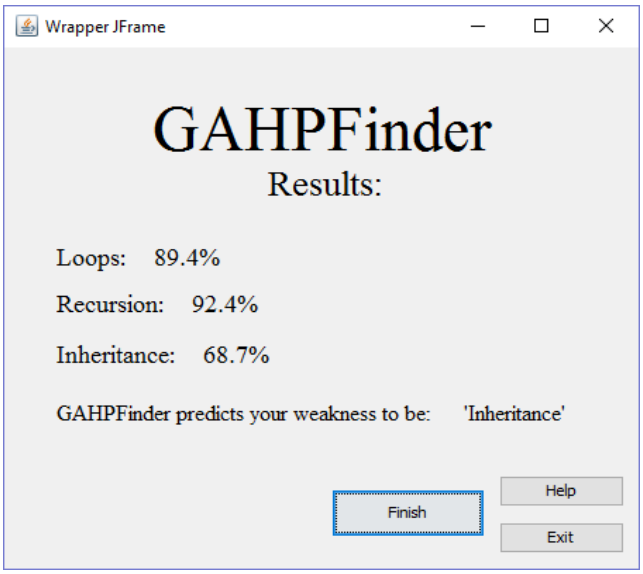

Fig. 13. Screenshot Image 4 of GAHPFinder Software

1. **Yes or No:** Prior to running the software, did you have a preconceived GAHP (Gray Area, Hazy Patch)?
2. **Yes or No:** Did the GAHPFinder's results support your preconceived GAHP(s)?
3. **Scale 1-10:** How well do you feel the software assessment was able to identify your GAHP(s)
4. **Scale 1-10:** How useful do you feel the GAHPFinder tool is to your cognitive development?
5. **Multiple Choice:** Overall, how would you rate the GAHPFinder tool in terms of usefulness in identifying your GAHP(s)?

Fig. 14. Sample Survey Questions

the conceptual understandings and gaps at the CS1 and CS2 levels.

III. RESULTS

The results from the two semesters of research analysis can be seen in Figures 15, 16, 17 and 18. Figures 15 and 17 represent the results of the students of CS 316 (the Advanced Software Design class), while Figures 16 and 18 reveal the results of the students from CS 371 (the Advanced Object Oriented Design course). These figures indicate that there are clear and concise areas of comprehension that resonate throughout the students in these CS1 and CS2 courses respectively. Figure 15 suggests that a majority of students in CS 316 share an epistemic belief that their conceptual gap lies in the topic of 'arrays', while the conceptual holes of students in CS371 (Figure 16) are primarily split between 'interfaces and 'recursion'. Similarly, Figure 17 reveals a general conceptual hole in 'class design' for CS 316 students, and a weakness in 'inheritance' (Figure 18) for CS371 students.

The overall results obtained from the reflection analysis and assessment of the 75 case study participants were able to provide a factual basis for validation of epistemic beliefs (epistemology) as expressed by the students themselves. From these results (as shown in Figure 19), it was revealed that approximately 65% and 53% of students from spring and fall semesters were able to successfully identify their cognitive areas of strength and weakness using a socially constructive self-reflection and analysis process. These numbers indicate that prior to any form of cognitive assessment, an average of 59% of the students were able to constructively derive their intellectual standing in a particular computing skillset or topic through reflection analysis. The software driven assessment reported that a successful match between self-reflection and traditional assessment was prevalent with 83% and 78% of students. This ultimately revealed that an average of 81% of students believed that the traditional assessment tool accurately supported their socially constructed competencies through self-reflection.

This experimentally obtained conceptual gap validation rate indicates a promising student epistemology performance in a computing learning environment supported by social constructivism. These results also assist in reconfirming the emphasis and importance placed on student reflection analysis that is pivotal to the action learning process within the social construction model. Lastly, these data show that regardless of preconceived conceptual gaps in step 1, 85% and 76% of students expressed that they were successfully able to achieve an epistemology level through the process of reflection analysis and assessment. A graphical representation of the overall research case study results have been presented through Figures 15, 16, 17, 18 and 19.

IV. FUTURE WORK AND SCOPE

This research study has been used to refine action learning at the UWGB ICS department for ensuring that the action learning cycle is completed in its entirety (as demonstrated via Figure 20). To assist with this, the CS1 and CS2 computing students are encouraged to give particular attention to their subject weaknesses as well as address their respective conceptual gaps, which have been identified and confirmed through the epistemic investigation, as described in this paper. As part of the social constructivist action learning process, students are challenged to execute semester projects, both individually as well as in teams leading to both individual enhancements and improvised peer learning.

By focusing on the identified individual weaknesses and embracing a peer-learning course structure, the students get the opportunity to refine their individual weaknesses into strengths and even share their knowledge with other student peers, thereby helping each other overcome each other's conceptual gaps. The benefits of conducting this social constructivist epistemology can be been seen at the individual student as well as at the curricular level. The student survey data have indicated that self-reflection, traditional assessment, and personal attention to their own self-discovered conceptual gaps have led to higher levels of epistemology. The promising results obtained through this experimental research case study can potentially set up an example for other computing educators, thereby inspiring them to conduct similar socially constructive epistemology experiments, using the presented model of epistemic investigation as a vehicle to deliver results.

V. CONCLUSION

This research paper illustrates a unique and specific instance of a novel epistemic investigation in computing education. It also presents an innovative scheme of reflective analysis for computing students, at the CS1 and CS2 levels specifically, within a social constructivist learning environment. This first of its kind case study produced results that clearly show that students were able to achieve significant levels of validated comprehension as part of the enhanced social constructivist action learning process. By adopting a mechanism of self-reflective analysis through interactive dialogue driven assessment, this experimental study was able to track and validate the epistemic progress of CS1 and CS2 computing students as they advanced their knowledge.

Furthermore, this study paves the path for an improved model of computing knowledge building in order to drive student comprehension and performance by employing an improvised action learning platform for the non-traditional social constructivist pedagogy that correlates with conducting a meaningful epistemology for students. Future scope of research work includes further extending this experimental case study for completing the enhanced action learning cycle (as illustrated in Figure 20). Another potential research direction is implementation of the presented social construction based epistemology process within other computing classes as well as expanding this research practice to other non-computing disciplines. The challenge ahead would be to apply the proposed mechanism of epistemic investigation successfully within a different curriculum and/or discipline, but with a similar social constructivist learning environment.

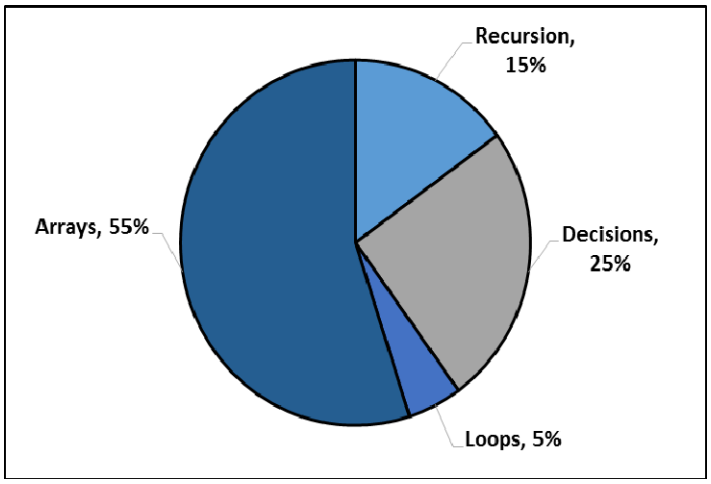

Fig. 15. Spring 2016 CS-316 Epistemic Opinions of Students

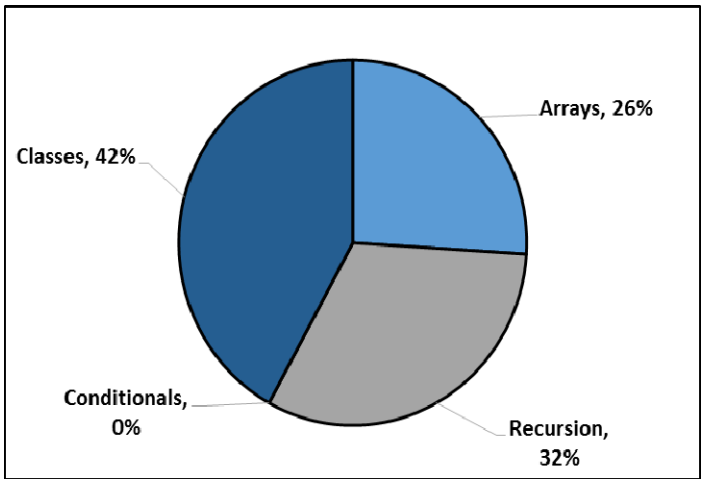

Fig. 17. Fall 2016 CS-316 Epistemic Opinions of Students

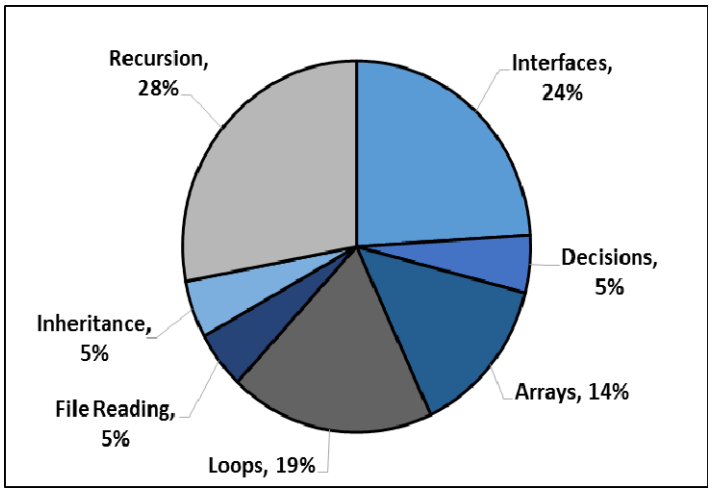

Fig. 16. Spring 2016 CS-371 Epistemic Opinions of Students

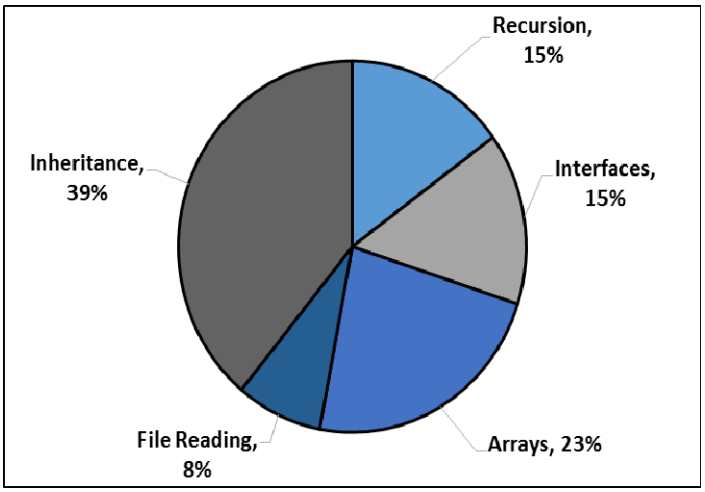

Fig. 18. Fall 2016 CS-371 Epistemic Opinions of Students

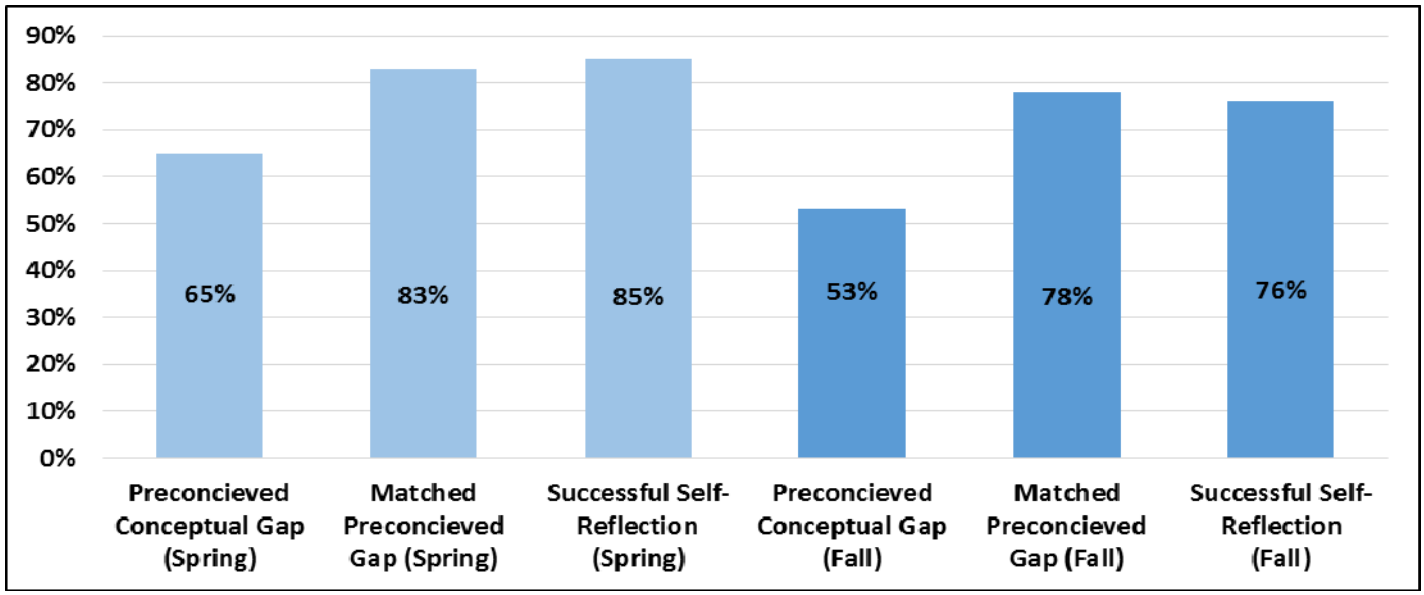

Fig. 19. Social Constructivist Epistemology For Computing Students - Experimental Case Study Results In 2016 At CS1 & CS2 Levels

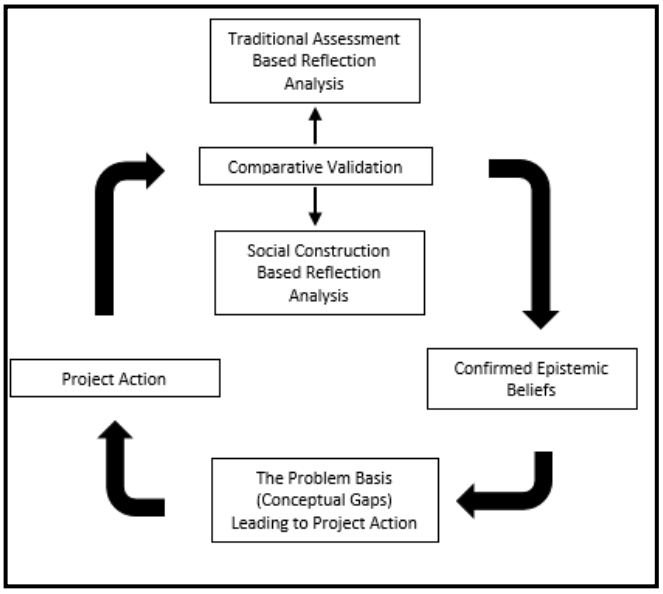

Fig. 20. Future Process of Epistemology Based Action Learning

REFERENCES

[1] Ben-Ari, M., "Constructivism in computer science education," *Journal of Computers in Mathematics and Science Teaching* 20.1, 45-74, 2001.

[2] Bilbokaitė, R., "Computer Based Visualization in the Context of Cognitive and Social Educational Constructivism," 2015.

[3] Bond, M., "Constructivist Teaching Strategies for First-Year Criminal Justice Students", 2016.

[4] Frisque, B., and Chattopadhyay, A., "Can We Conduct A Social Construction Based Epistemology for CS1 and CS2 Students?," *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 2017.

[5] Horstmann, C., "Java Concepts: Early Objects," *John Wiley & Sons*, 2014.

[6] Kun, L., Chan, C., and Wang, S., "Fostering Students' Epistemic Beliefs and Scientific Understanding through Knowledge Building," 2016.

[7] Lam, C., "Fostering conceptual change and epistemic changes in chemistry through collaborative knowledge-building inquiry," *HKU Theses Online*, 2012.

[8] Lister, R., "Toward a Developmental Epistemology of Computer Programming," *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*. ACM, 2016.

[9] Luo, D., "Using constructivism as a teaching model for computer science," *The China Papers*, 36-40, 2005.

[10] Machanick, P., "A social construction approach to computer science education," *Computer Science Education* 17.1, 1-20, 2007.

[11] Meetoo-Appavoo, A., "Constructivist framework for teaching computer science," *International Journal of Computer Science & Information Security* 9.8, 25, 2011.

[12] Pear, J., and Crone-Todd, D., "A social constructivist approach to computer-mediated instruction," *Computers & Education* 38, no. 1, 221-231, 2002.

[13] Rufii, R., "Developing Module on Constructivist Learning Strategies to Promote Students' Independence and Performance," *International Journal of Education* 7.1, 18, 2015.

[14] Stojkovski, T., "Computer-mediated learning in a social constructivist environment," 2010.

[15] Sung, L., "Experienced College Instructors' Epistemology in Teaching Social Studies & Their Perception of Using Technology," *Philosophy Study* 1.5, 360, 2011.

[16] Teague, D., "Neo-Piagetian Theory and the novice programmer," *Diss. Queensland University of Technology*, 2015.

[17] Valentine-Maher, S., et. al., "Teaching population health and community-based care across diverse clinical experiences: Integration of conceptual pillars and constructivist learning," *Journal of Nursing Education* 53.3, S11-S18, 2014.