A Framework for the Automated Assessment of Consistency Between Code and Design

Alan Hayes[1], Pete Thomas[2] , Neil Smith[3], Kevin Waugh[4]

[1]*University of Wales Newport, Newport NP20 5XR,United Kingdom, alan.hayes@newport.ac.uk*

[2]*Open University, Milton Keynes UK MK7, United Kingdom, p.g.thomas@open.a c.uk*

[3]*Open University, Milton Keynes UK MK7, United Kingdom, n.smith@open.a c.uk*

[4]*Open University, Milton Keynes UK MK7, United Kingdom, k.g.waugh@open.a c.uk*

In this paper, we present an investigation into the development of a framework for the automatic grading (marking) of student submitted course work. We discuss this framework, its structure and its subsystems. Our context is the consideration of the case where undergraduate Computing students submit coursework that consists of two components: a design (using the UML methodology) and implementation (using the Java programming language). The focus of our framework is upon the consistency between the implementation and design. We discuss its context and development and highlight how we can infer structure from the student submission and use this to inform the assessment process. We define consistency from the viewpoint of a design and its implementation are two different representations of the same artefact. The design (in diagrammatic format) is viewed as prescribing the structure and function contained within the implementation, whilst the implementation (source code) is viewed as implementing the design whilst adhering to its specified structure and function. We consider consistency to be important as it enables the student to demonstrate adherence to the development life-cycle. In making it explicit that we are grading a submission for consistency we wish to provide feedback that will engender within the student an engineering approach to the development of a software product.

Keywords

Computer Science Education; Automated Assessment

1. Introduction

In this paper, we present an investigation into the development of a framework for the automatic grading (marking) of student submitted course work. Our context is the consideration of the case where the student submission consists of two components: a design (using the UML methodology) and source code (using the Java programming language). The focus of our framework is upon the consistency between the student code and design. We consider consistency to be important as it enables the student to demonstrate adherence to the development life-cycle. In grading a submission for consistency we wish to provide feedback that will engender within the student an engineering approach to the development of a software product. The course that students are studying and consequently the coursework upon which we will apply our developed framework is a B.Sc.(hons) Computing programme. In particular, the module that we are considering applying our technique to is that delivered to year 2 undergraduate students in the subject area of object oriented development.

Our interest in this area has arisen from work in the automated assessment of free-form, diagrams submitted by students as a component of their assignment. Work in this field focuses on both fully automated [1, 2] and semi-automated [5, 6] assessment systems. In both contexts, student-submitted diagrams are often imprecise and contain missing or extraneous data. Often such data promulgates into the student code and it is against this background that we are developing our automated assessment framework. Additionally, work in the automated assessment of student source code [3, 4] led to our consideration of the interface of the design and code areas and the implications for the automated assessment process. Consequently, our framework considers the automation of the assessment of student coursework for consistency between the code and design.

The remainder of this paper presents the context of our framework followed by discussing how we can infer design and code structure from the student submission. We then present a model for how this inferred structure can be used to aid the automated assessment process before describing how we currently intend to apply this framework to our undergraduate scheme.

2. Automated Marking - Context

The context of the approach we have taken to the automated assessment of the student submission is illustrated in figure 1 below. The student submission consists of two separate deliverables: a design and an implementation.

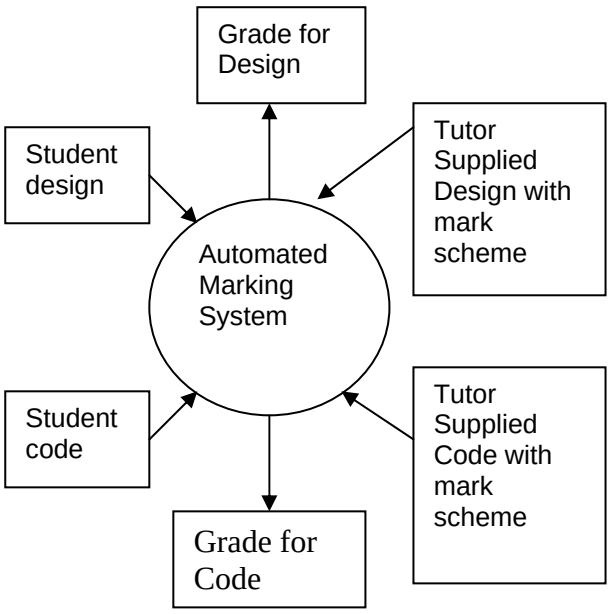

Figure 1: Initial Context of an Automated Marking System

If we treat the two student submissions as disjunctive non-related deliverables it would be possible to divide the automated marking system into two distinct components, one focusing on the design and one on the implementation. This approach is illustrated in figure 2.
However, such an approach does not lend itself to focusing upon the interface between the code and the design. When considering the automation of the assessment process for consistency at the interface there needs to be a link between the structure of the student code and that of the accompanying design. However, a consequence of a disjunctive approach is that a mark scheme that focuses upon consistency between the student design and student code may not be supplied by the tutor. This particularly may be the case when,

for example, the submission date is different for each deliverable to allow for formative feedback to be given on the design before the student embarks upon the implementation stage or when the design and implementation assignments are contained within two separately delivered modules (integrative assignment). However, it remains the case that an assignment comprising of an integrated design and implementation needs the student to produce more than just a design and an implementation. Consequently, our proposed framework looks at enhancing the model illustrated in figure 2 with a view to providing support for the automated assessment process when the context is grading for consistency between the two separate deliverables of a design and an implementation

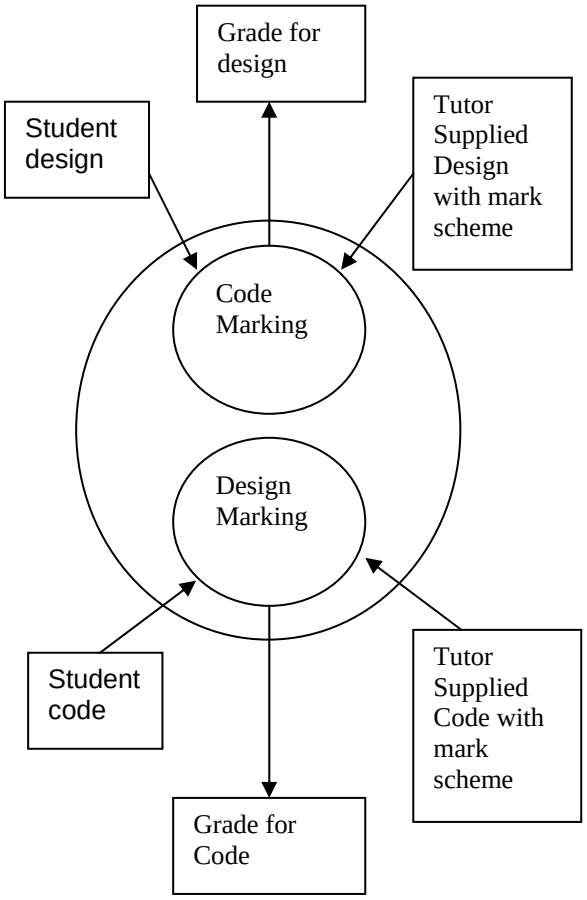

Figure 2: A system that marks the design and the code disjunctively

2.1 Inferred Structures

When the focus of the assignment is that of consistency between the source code submitted and its accompanying design there are several marking models that emerge for the automated tool. Focusing upon the student submission, with an appropriate tool, it would be possible to imagine forward engineering the student's design to produce an idealised structure for the submitted student code. It is also possible to imagine an appropriate tool that would reverse engineer the student code and produce an idealised structure for the student design. Consequently an automated marking tool has the possibility of generating two further enhancements to that illustrated in Figure 1. This is illustrated in Figures 3 and 4.

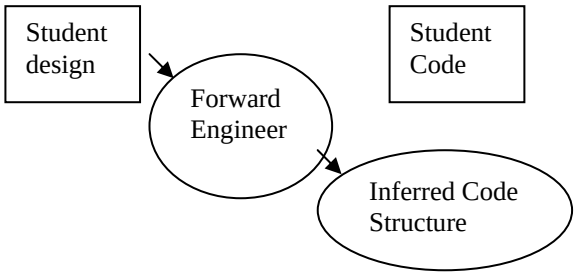

Figure 3: Forward Engineer the Design to produce the inferred code structure

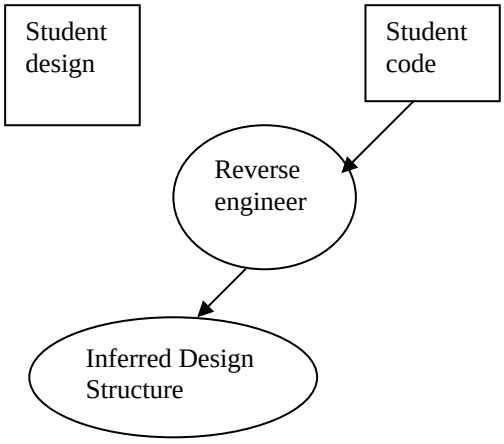

Figure 4: Reverse engineer the code to produce the inferred design structure

2.2 Inferred Structures and the Assessment Process

Having created the inferred design and the inferred code it would be possible to use them as input into the automated marking system. It should be noted that the inferred models do not contain any marking allocation as they have been derived from the student submission. An automated marking system could use these inferred structures to both confirm consistencies and identify inconsistencies within the student submission. Such an approach leads to three possible models. Each model offers a different perspective upon the student submission and consequently provides an automated marking system with the potential of utilising each resultant model to analyse and determine a grade for the student submission.
The first step is to compare the submitted design and the inferred design structure. We refer to this as a design-centric model. Discrepancies identified in the comparison could be used to identify possible inconsistencies between the student code and the accompanying student design. Hence, one model that focuses upon consistency between the student code and the student design would be for the automated marking system to take as its input the student design and the inferred design structure. This is illustrated in Figure 5.

The second model is to compare the student code with the inferred code structure. We refer to this as a code-centric model. Discrepancies identified in the comparison could be used to identify possible inconsistencies between the student code and the inferred code structure. This is illustrated in figure 6.

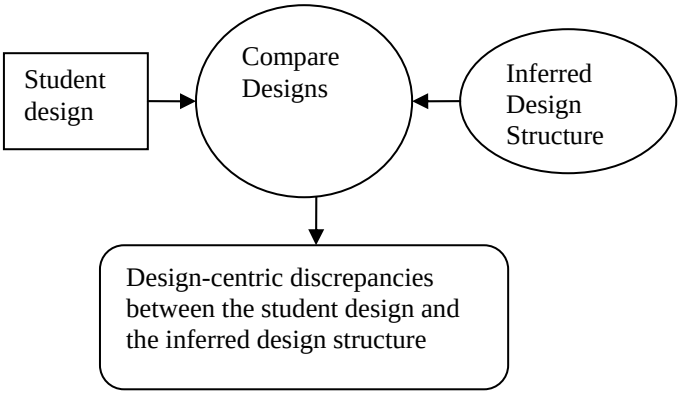

Figure 5: A model that focuses upon comparing the student design with the inferred design structure

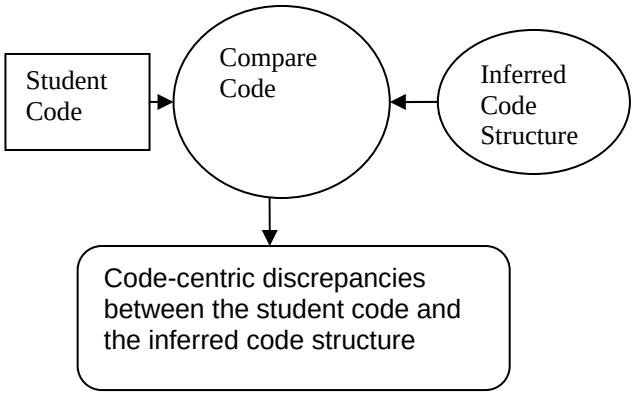

Figure 6: A model that focuses upon comparing the student code with the inferred structure.

Ideally the results from adopting these two approaches would be identical. However, this may not be the case. The imprecise nature of the student submission for both the code and the design could lead to ambiguities in inconsistencies identified. For example, erroneous or missing data in the student design will be reflected in the derived inferred code structure. This erroneous data may not be reflected in the student code. Similarly, erroneous data contained in the student code will be reflected in the inferred design structure. This erroneous data may not be reflected in the student design. It is therefore, possible to envisage a third model that triangulates between the two models identified (figure 7).

Such triangulation would enhance the model by providing the possibility of two benefits. The first is that of confirmation of consistencies identified between the student code and the student design. For example, a component in the student submission identified as being consistent when comparing the student design with the inferred design that is also identified as being consistent when comparing the student code with the inferred code leads to a high degree of confidence in concluding that the component has been designed and implemented consistently by the student.

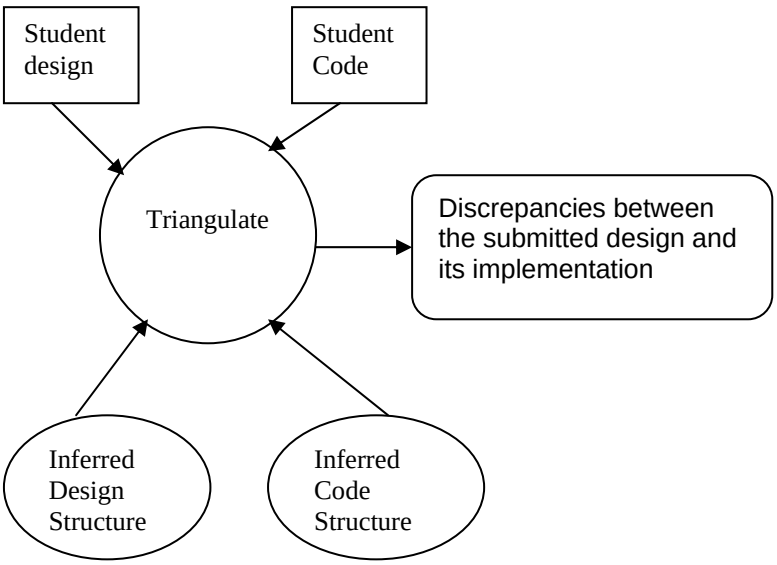

Figure 7: Triangulate the Assessment of the student submission with both the inferred code structure and inferred design structure

The second benefit offered by triangulation is that of confirmation of inconsistencies identified between the student design and the student code. For example, a component identified as being inconsistent in the student submission when comparing the student design with the inferred design that is also identified as being inconsistent when comparing the student code with the inferred code leads to a high degree of confidence in concluding that the component has been designed and implemented inconsistently by the student.

There remains one further case to consider under the triangulation model. It is conceivable that inconsistencies identified when comparing the student code with the inferred code are not identified when the student design is compared with the inferred design. It is possible to imagine, with an appropriate tool, that the triangulation model would facilitate some resolving of this type of ambiguity.

3 Experimentation

We are currently conducting experimentation in the automated assessment of student submitted course work using the techniques and models illustrated above. We are developing our system using data collected from work submitted for assessment by undergraduate computing students. We have refined our model to include the use of a tutor-supplied mark scheme that attributes marks for consistency between the source code and the design. Such a marking scheme would guide the automated marking system in how marks are to be awarded/deducted for consistencies/inconsistencies identified when comparing the student submission with the inferred models. This is illustrated in figure 8.

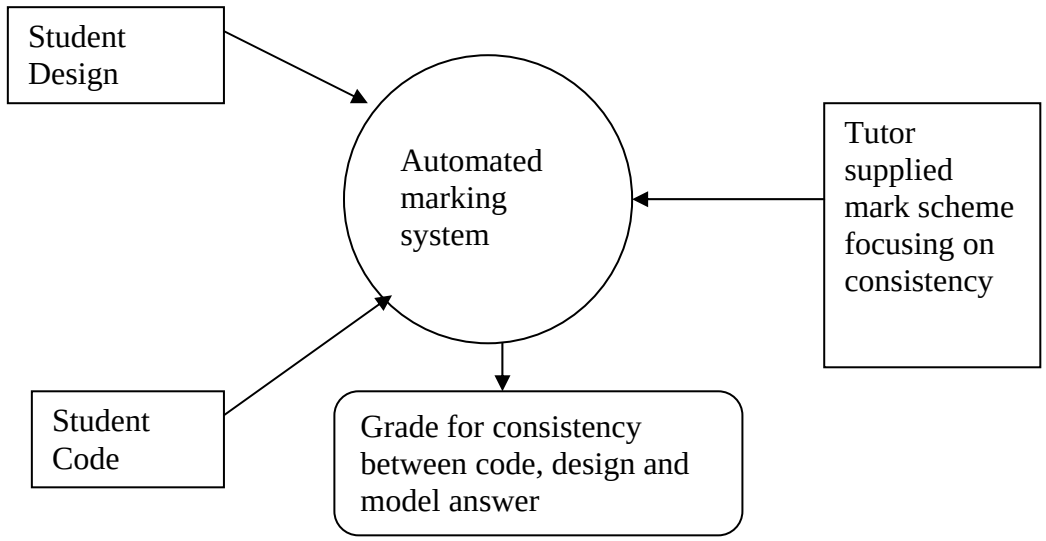

Figure 8: A model that requires the tutor to supply a marking scheme that focuses upon consistency

However, we have found that within such a model there remains a need to be able to compare the student submitted code with the student submitted design. We are adopting the above approach of using reverse and forward engineering techniques to facilitate traversal between the student code and student design to produce the inferred code structure and the inferred design structure. We are developing a system that uses the tutor supplied mark scheme as the mechanism for allocating a grade. In this context our automated marking system analyses the student design and compares this with the inferred design structure generated from the student code and the tutor supplied marking scheme. This is illustrated in figure 9.

We are currently at the data-collection stage of this project. We have designed the integrative assignment and distributed it to the student cohort. We have produced the marking scheme and identified the UML design and Java implementation tools. Once this data has been collected we intend to instantiate our framework and conduct further experimentation. This will consist of using one of the forward and reverse engineering tools commercially available, for example those features found in Borland's JBuilder Enterprise, and using the work on the assessment of diagrams in [1] to implement the model in figure 9.

4. Conclusion

We have presented and discussed our framework for the development of an automated assessment tool that focuses upon consistency between the source code and the design. An integral component to our approach is the adoption of reverse and forward engineering techniques to produce an inferred structure from the student submission. We have discussed how such inferred structures can be used to support the automated assessment process. We have discussed and illustrated the need for an assessment tool to triangulate between the inferred structures and a tutor-supplied marking scheme. We have described the experimentation and development that we are currently undertaking to utilise these models in the development of our assessment framework.

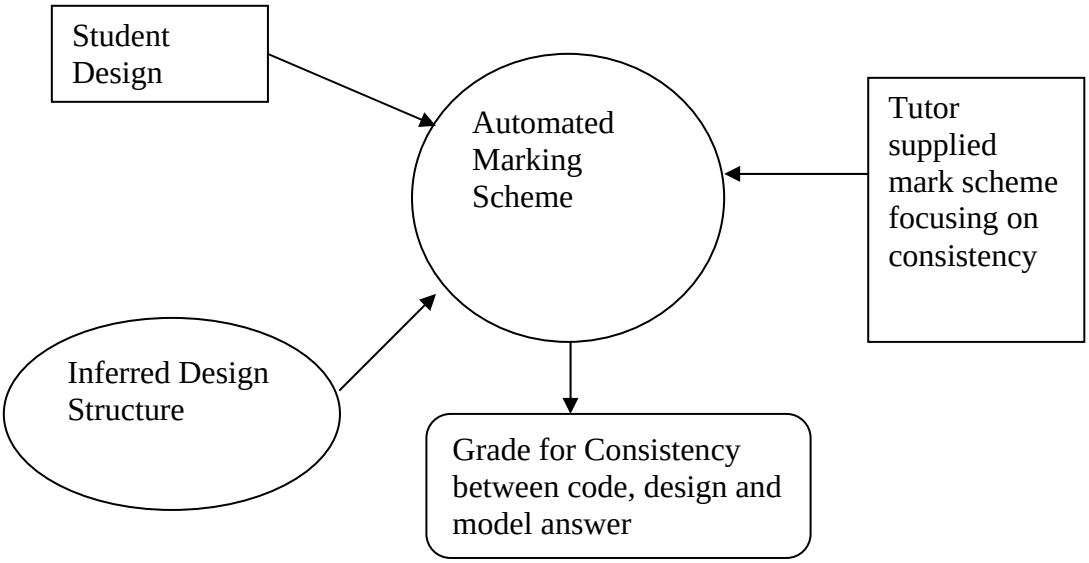

Figure 9: A model that marks the student submitted design by using input from the student design and a tutor-supplied mark scheme focusing on consistency

5. Future Work

It is anticipated that the resulting automated marking system will need to traverse easily between source code and design. Further work needs to take place in ensuring that the data structure and format adopted by the tool to represent the design would be the same as that used to represent the code. This is because the models require traversal and comparison between structures contained within the student code and those contained within the student design. Additionally, we intend to instantiate our framework and consequently develop an automated marking tool. We intend to exercise the resultant system against the student data that we are in the process of collating and subsequently publish our results. In this context our results will constitute a comparison between the grades produced by our assessment system and those generated by the (human) marking team.

References

1 Thomas, P., Waugh K. and Smith N., (2005) Experiments in the Automated Marking of ER-Diagrams. In *Proceedings of 10th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2005)* (Lisbon, Portugal, June 27-29, 2005).

2 Smith N., Thomas, P. and Waugh K.(2004) Interpreting Imprecise Diagrams. In *Proceedings of the Third International Conference in Theory and Applications of Diagrams.* March 22-24, Cambridge, UK. Springer Lecture Notes in Computer Science, eds: Alan Blackwell, Kim Marriott, Atsushi Shimomnja, 2980, 239-241. ISBN 3-540-21268-X.

3 English, J. (2004) Automated Assessment of GUI Programs using JEWL . In *Proceedings of the 9th annual SIGCSE conference on Innovation and Technology in Computer Science Education (ITiCSE '04) (Leeds, United Kingdom, June 2004).*

4 Tsintsifas, A. (2002) A Framework for the Computer Based Assessment of Diagram Based Coursework. *Ph.D. Thesis University of Nottingham, School of Computer Science and Information Technology.* March 2002.

5 Batmaz F. and Hinde C.(2006) A Diagram Drawing Tool for Semi-Automatic Assessment of Conceptual Diagrams. In Proceeding s of the 10th International Conference on Computer Assisted Assessment,(Loughborough, July 2006)
6 Tselonis C., Sargeant J and McGee M. (2005) Diagram Matching for Human-Computer Collaborative Assessment. In *Proceeding s of the 9th International Conference on Computer Assisted Assessment*, (Loughborough University July 2005).
7 Thomas, P.G., Waugh, K and Smith, N. (2007) Computer Assisted Assessment of Diagrams. In the *Proceedings of ITiCSE*, 25-29 June 2007, Dundee Scotland
8 Thomas, P.G., Waugh, K, Smith, N. (2007) Learning and automatically assessing graph-based diagrams. In the Proceedings of 14th International Conference of the Association for Learning Technology (*ALT-C)*, September 2007, Nottingham, UK
9 Waugh, K.G, Thomas, P.G, Smith, N. Teaching and learning applications related to the automated interpretation of ERDs. In Proceedings of the Fifth Workshop on Teaching, Learning and Assessment of Databases (TLAD) , July 2007, Glasgow, UK