ISSN 1744-1986

The Open University

Technical Report N^{o} 2007/06

A Developmental Framework for Computer-based Automated Assessment

Alan Hayes
Pete Thomas
Neil Smith
Kevin Waugh

19th April 2007

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk

A Developmental Framework for Computer-based Automated Assessment

Alan Hayes, Pete Thomas, Neil Smith, Kevin Waugh

Abstract

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 framework has been developed in the context of the student submission consisting 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 discuss its context and development and highlight how we can infer structure from the student submission and use this to inform the assessment process.

1.0 Introduction

This document is concerned with discussing the development of a computer based assessment system that focuses upon the automated marking of student submitted coursework. We present an overview of several high-level models for how such a system could be developed. No attempt is made to consider the internal operational detail of the models presented. The focus, instead, is to consider the inputs that such models might require and to identify and discuss the operational challenges that each model presents. The aim of the document therefore is to provide a framework of several models that address automated assessment and an identification of research areas that need to be addressed in order to facilitate their implementation. 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 context 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 impact that such errors in the design have upon the implementation process. Consequently, our framework considers the automation

of the assessment of student coursework for consistency between the code and design.

The student submission consists of two components: a design (using the UML methodology) and source code (using the java programming language). A marking scheme is provided by the tutor for both submissions. This is illustrated in fig 1 below.

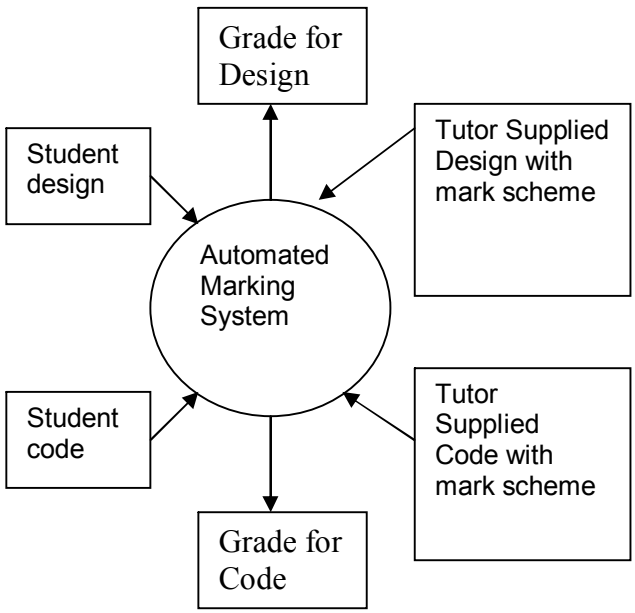

Figure 1: Initial Context of an Automated Marking System

2.0 Developing the Framework

This section provides an evolutionary context of the development of our framework. We start by considering a disjunctive approach to the assessment of the design and code before considering the implications of integrating them together to consider issues surrounding consistency between the design and its implementation. We illustrate how forward and reverse engineering techniques might be used to infer a structure for the student submission and discuss how this inferred structure can be used in the assessment process. We move on to consider the role of a tutor-supplied mark scheme and consider an assessment model that triangulates between the mark-scheme and the inferred structure.

2.1 The Disjunctive Approach

If we treat the two student submissions from figure 1 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. It is worth noting that a consequence of this disjunctive approach is that a mark scheme that focuses upon consistency between the student design and student code is not necessarily supplied by the tutor. The approach is illustrated in figures 2 and 3.

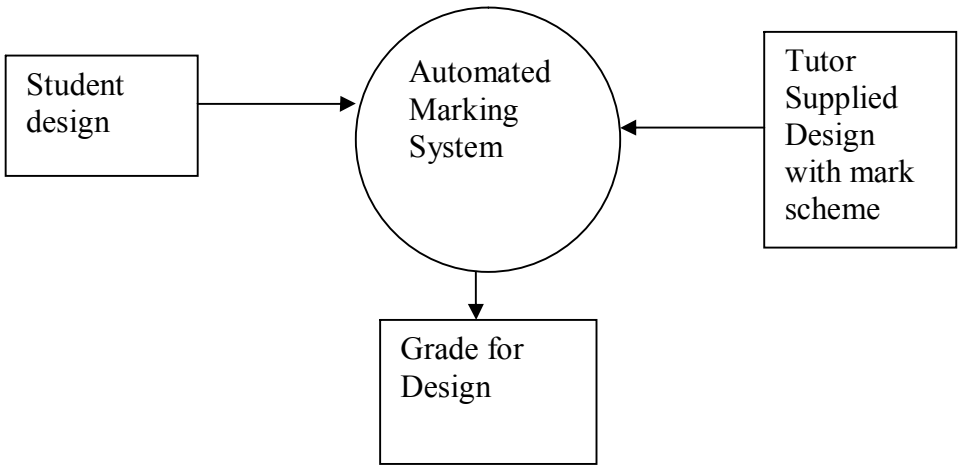

Figure 2: A system that focuses upon the Design

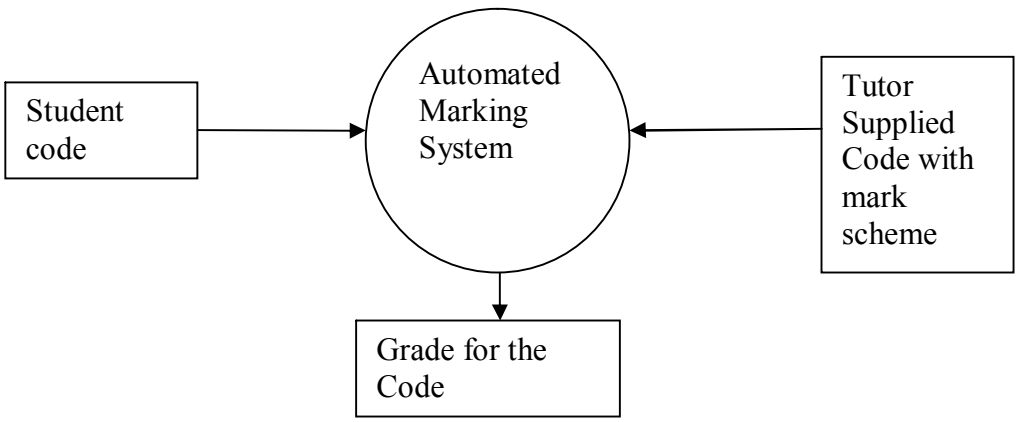

Figure 3: A system that focuses upon the code

2.2 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. We refer to this as the *inferred code structure*. 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. We refer to this as the *inferred design structure*. Consequently an automated marking tool has the possibility of generating two further enhancements to that illustrated in Figure 1. This is illustrated in Figures 4 and 5.

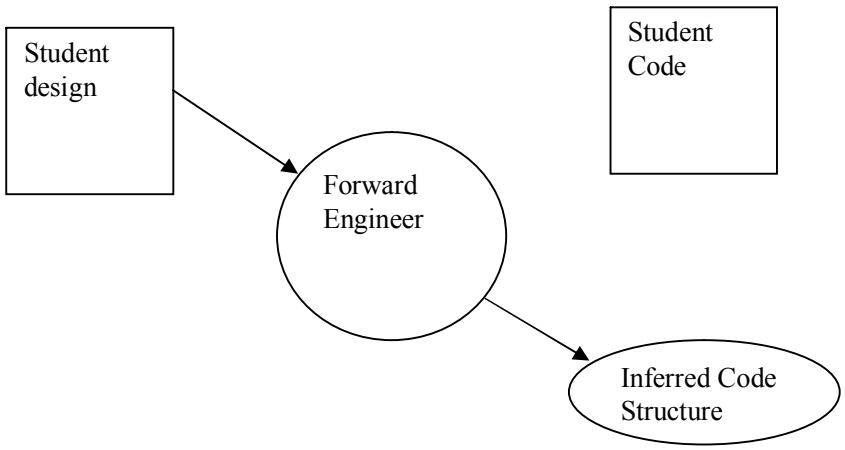

Figure 4: Forward Engineer the Design to produce the inferred code structure

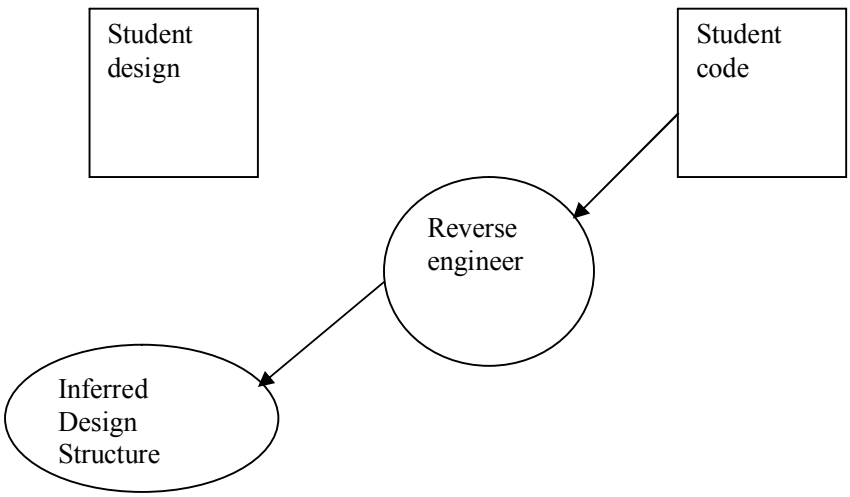

Figure 5: Reverse engineer the code to produce the inferred design structure

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. However, it is feasible that an automated system could use them to both confirm consistencies and identify inconsistencies in the student submission. Such an approach leads to three possible models. The first model would be to focus upon the student submitted design and compare this with the inferred design structure that has been generated from the student code. 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 6.

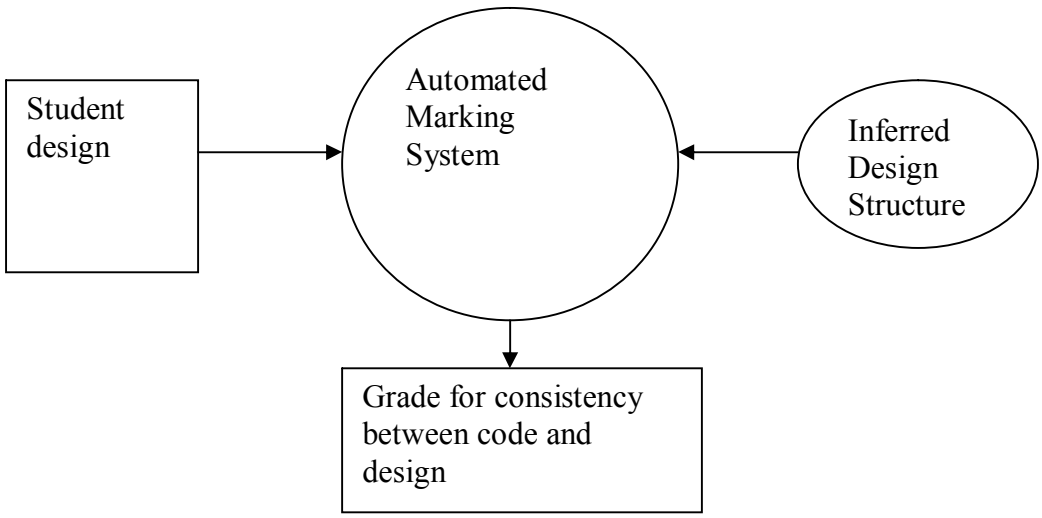

Figure 6: A model that focuses upon comparing the student design with the inferred design structure

The second model involves focusing upon the student code and comparing this with the inferred code structure that has been generated from the student design. 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 7.

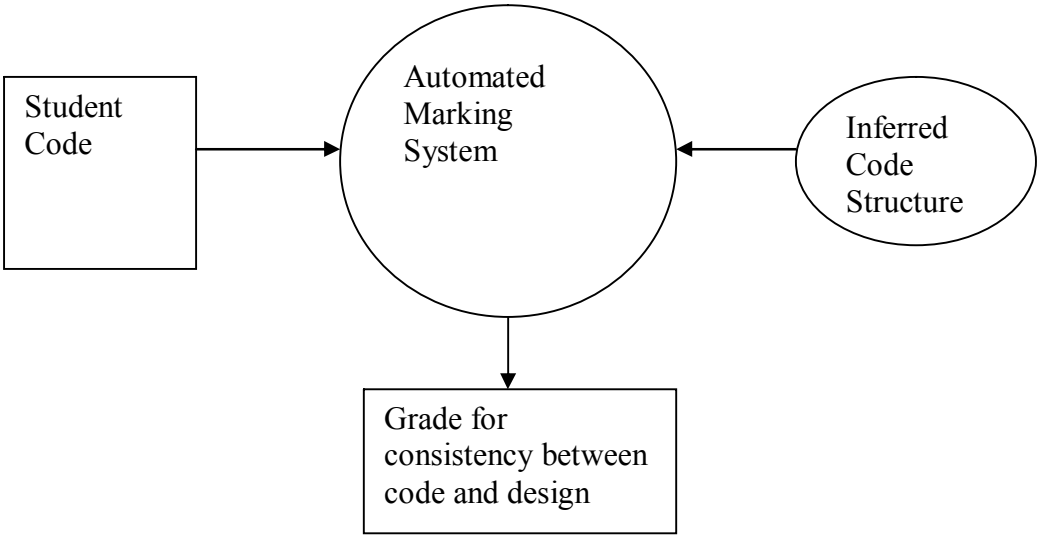

Figure 7: A model that focuses upon comparing the student code with the inferred structure.

2.3 Triangulation between Inferred Structures

Ideally the results from adopting the model in figure 6 would be the same as those delivered by the model contained in figure 7. However, this may not necessarily 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 model that triangulates between the two models identified above in figures 6 and 7. Such triangulation would enhance the model by offering the possibility of offering two enhancements. The first is that of confirmation of consistencies identified between the student code and the student design. For example, an automated marking system that confirms a correct component in the student submission when comparing the student design with the inferred design structure has a high degree of confidence in that component being correct if it is also identified as being present and consistent when comparing the student code with the inferred code structure.

The second enhancement offered by triangulation is that of confirmation of inconsistencies identified between the student design and the student code. For example, an automated marking system that identifies an inconsistent component in the student submission when comparing the student design with the inferred design structure has a high degree of confidence in that component being incorrect if it is also identified as being inconsistent when comparing the student code with the inferred code structure.

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 structure are not identified when the student design is compared with the inferred design structure. It is possible to imagine, with an appropriate tool, that the triangulation model would facilitate some resolving of this type of ambiguity. The triangulation model is illustrated in figure 8 below.

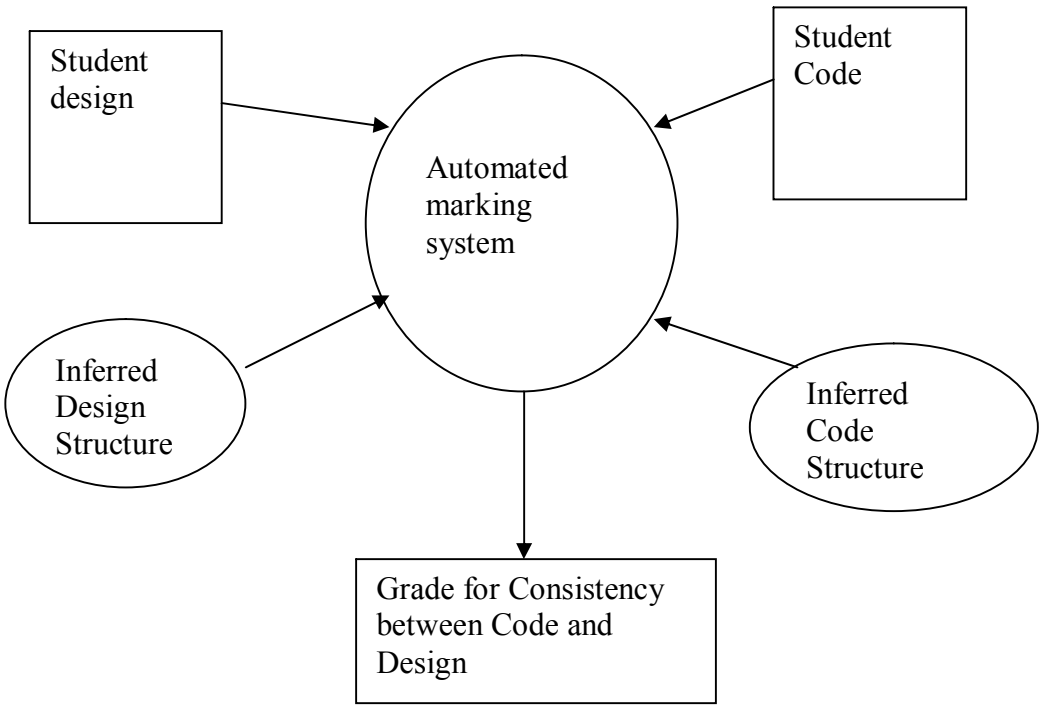

Figure 8: Triangulate the Assessment of the student submission with both the inferred code structure and inferred design structure

2.4 Triangulation with Marking Schemes

There are two problems associated with the models above. The first is that the inferred models do not contain a marking scheme as they have been generated from the student submission. Figures 6, 7 and 8 identify how inconsistencies between the student design and student code could potentially be identified but they do not take into account the tutor supplied design with mark scheme nor the tutor supplied code with mark scheme. Furthermore, figure 1 indicates that the model developed thus far does not contain a tutor-supplied mark scheme that focuses upon consistency between the student code and the student design.

The second problem is that although the automated marking system has been developed to check for consistency between the student code and student design we have not checked that the submission meets its requirement specification. In practice the student submission could deviate significantly from what was asked and the tool, as currently presented, would return a high mark based upon consistency between the two submissions. Both problems are solved by incorporating tutor-supplied marking schemes into the generation of the inferred structures.

In considering the first problem, that of a lack of grading criteria that considers consistency issues, we need to include a mechanism for allocating a grade based upon these criteria. One possible solution to this could be to analyse the marking schemes supplied by the tutor (as indicated in figure 1) and use them to infer a marking scheme based upon the discrepancies detected when comparing the student design with the inferred design (or the student code with the inferred code). Consequently it would be possible to imagine an enhancement to the forward and reverse engineering models of figure 4 and 5 to include taking full cognisance of the tutor supplied marking schemes. Figure 9 illustrates the generation of a marking scheme for the student code by forward engineering the student design to produce the inferred code structure and comparing this with the tutor supplied code and mark scheme to produce an inferred code mark scheme. Similarly figure 10 illustrates the generation of a marking scheme for the student design by reverse engineering the student code to produce the inferred design structure and comparing this with a tutor supplied code and mark scheme to produce an inferred design mark scheme.

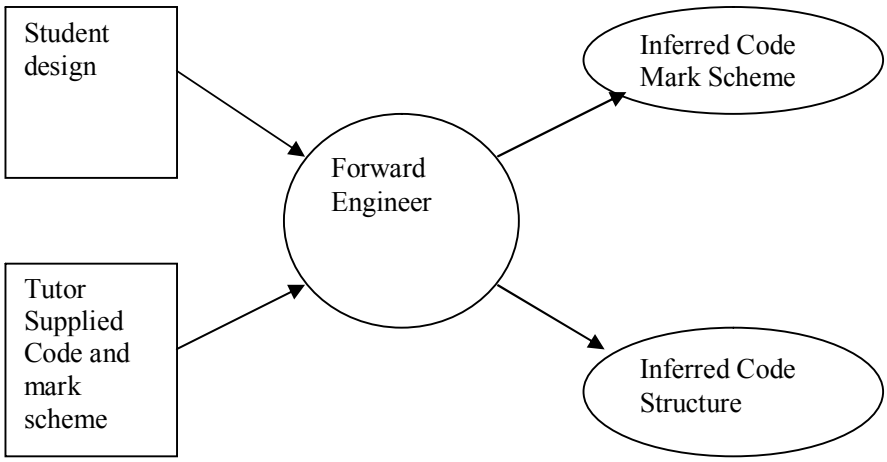

Figure 9: Utilise the tutor supplied code and mark scheme to produce a mark scheme for the inferred code structure

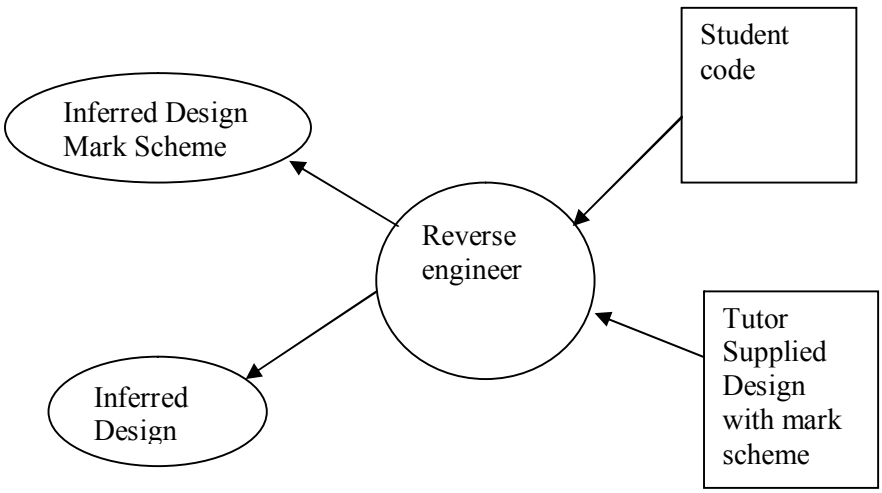

Figure 10: Utilise the tutor supplied design with mark scheme to produce a mark scheme for the inferred design structure

Consequently the models illustrated in figures 6,7 and 8 could potentially be enhanced to allocate a marking component to the consistencies and discrepancies identified by using the inferred marking schemes produced from models illustrated in figures 9 and 10. This is illustrated in figures 11, 12 and 13.

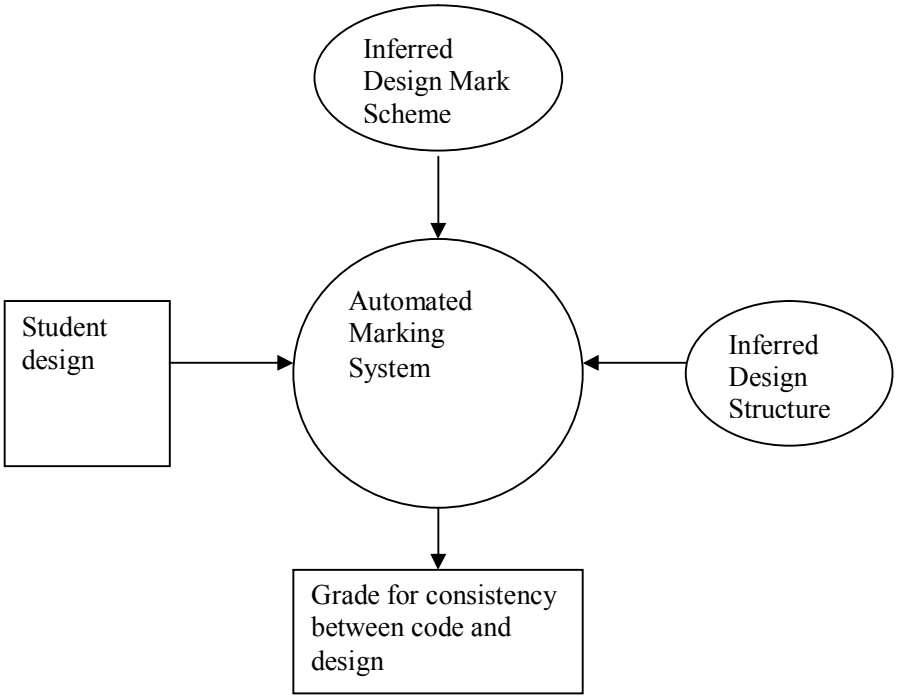

Figure 11: An automated system that marks a student design for consistency with the student code using an inferred design structure and an inferred design mark scheme

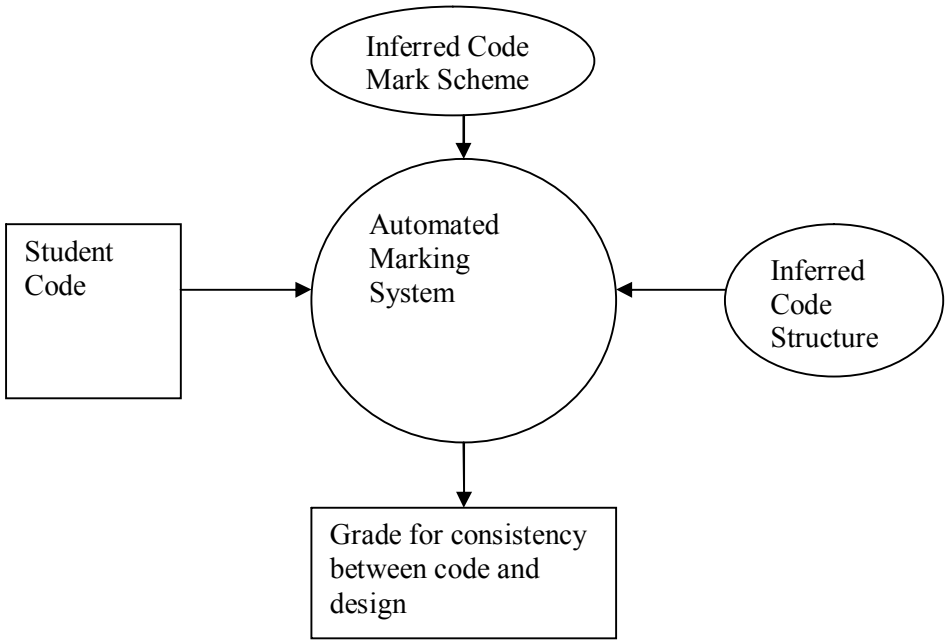

Figure 12: An Automated Marking System that marks student code for consistency with the student design using an inferred code structure and an inferred code mark scheme

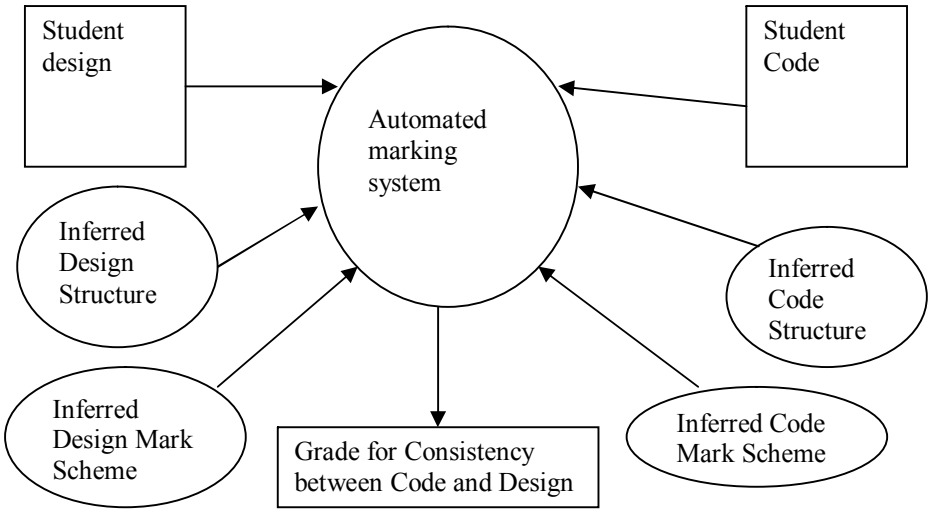

Figure 13: Triangulate the Assessment of the Student Submission with both the inferred code structure and the inferred design structure

In utilising the tutor-supplied solutions in the generation of the automated mark schemes for the inferred structures we have also addressed the second problem identified: that of consistency with the requirements specification. The idealised marking scheme has been generated using the tutor-supplied solution which, by assumption, represents conformance with the requirements.

An alternative model to automatically producing an inferred mark scheme would be to require the tutor to supply a marking scheme that focussed upon consistency between the 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 14.

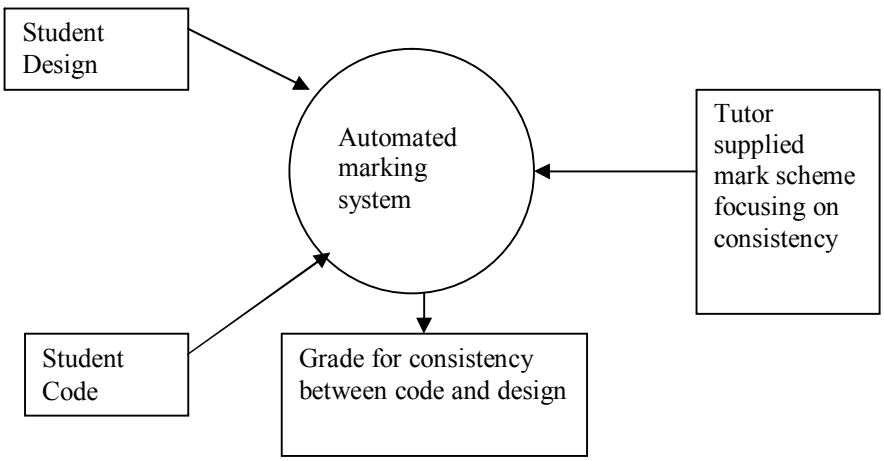

Figure 14: A model that requires the tutor to supply a marking scheme that focuses upon consistency

However, within such a model there remains a need to be able to compare the student submitted code with the student submitted design. Adopting an approach similar to the above we can imagine how it would be possible to use 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. In this case however, instead of deriving an inferred mark scheme as indicated in figures 9 and 10 we could utilise the tutor supplied mark scheme focusing upon consistency as the mechanism for allocating a grade. In this context an automated marking system would analyse the student design and compare this with the inferred design structure generated from the student code and the tutor supplied marking scheme. This is illustrated in figure15.

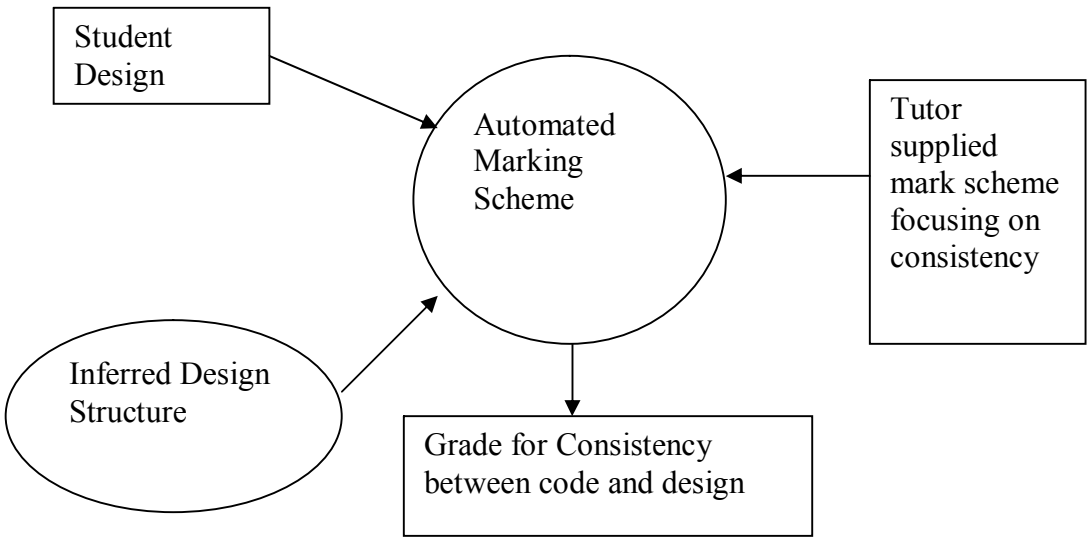

Figure 15: A model that marks the student submitted design by using input from the student design and a tutor-supplied mark scheme focusing on consistency

Alternatively, the automated marking system would analyse the student code and compare this with the inferred code structure generated from the student design and the tutor supplied marking scheme. This is illustrated in figure 16.

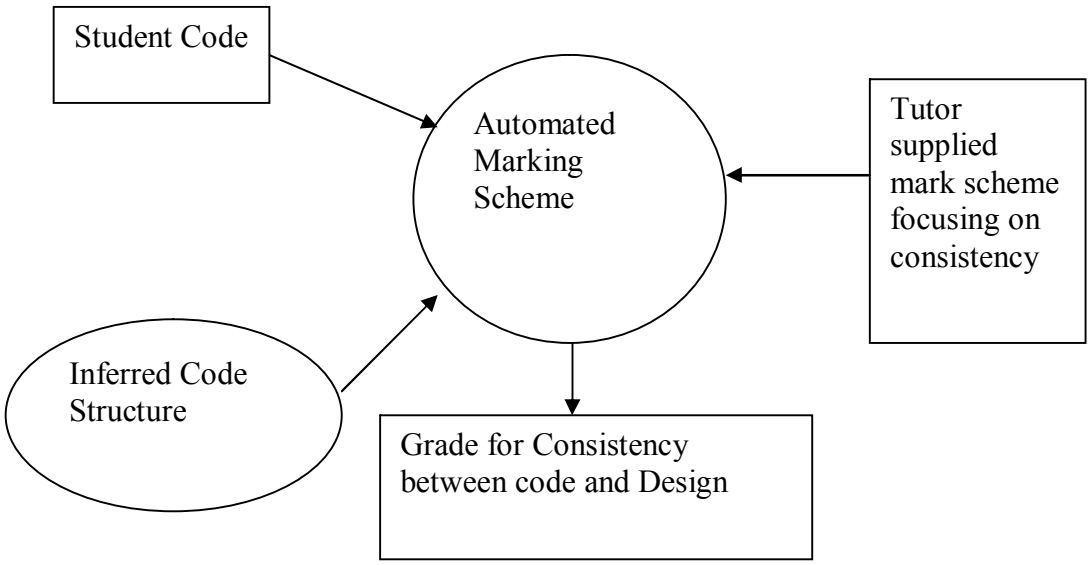

Figure 16: A model that marks the student code by using input from the inferred code structure and a tutor supplied mark scheme focusing on consistency

2.5 Possible Uses for Plagiarism Detection

One possible bi-product of the assessment models illustrated above is that of plagiarism detection. In particular, figures 6 and 7 illustrate how it would be possible, with an appropriate tool, to compare the student design with the inferred design structure and the student code with the inferred code structure. In performing this comparison, it is possible that, for example, significant deviation between the student code and the inferred code structure could be detected. This could be potentially interpreted as implying that one or both components in the student submission (i.e. the student design or the student code) has possibly been plagiarised. A similar conjecture could also be constructed should a similar significant deviation between the student design and the inferred design be detected. Further work would need to be undertaken regarding what constitutes a significant deviation. It should also be stressed that such a system could not be seen as proving that plagiarism has taken place but that there is significant discrepancy in the submission to warrant further investigation. One method of further investigation would be to manually inspect the submission and if appropriate invite the student to attend a viva.

An alternative plagiarism detection model would be to take advantage of the fact that the inferred design structure, generated by an appropriate tool, will be in the same format for each student submission. Comparing a student's inferred design with those from all other students offers the potential to detect potential plagiarism. A student whose inferred design structure is sufficiently similar to another's could be an indication of potential plagiarism. In this case plagiarism has potentially taken place in the student code as it is the processing of the student code that produces the inferred design structure. Further work would need to be undertaken to consider what is meant by sufficiently similar. Again, it should also be stressed that such a system could not be seen as proving that plagiarism has taken place but that there is significant similarity in the submission to warrant further investigation. One method of further investigation would be to manually inspect the submissions and if appropriate invite the students concerned to attend a viva. It is also possible to imagine an alternative but similar approach that focuses upon comparing the inferred code structures for each student. In this case sufficient similarity would highlight potential plagiarism has taken place in the student design.

3.0 Concluding Comments

The above discussion presents several research challenges. Figure 2 is concerned with comparing the design submitted by the student with the solution presented by the tutor. The imprecise nature of student diagrams and the marking thereof is a rich area of interest. Figure 3 is concerned with the automated marking of student code. This too is a rich area with a focus upon both the static structure of the student submission in addition to its dynamic behaviour. Figures 4 and 5 are concerned with techniques of reverse and forward engineering. Figures 6, 7 and 8 represent different models for how an automated marking tool can identify and corroborate consistencies and inconsistencies identified in the student submission. The automated marking system will need to traverse easily between source code and design. Ideally, the data structure and format adopted 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. Figures 9 and 10 extend the concepts of reverse and forward engineering further as they require the generation of a marking scheme based upon both the student submission and the tutor supplied solution. Inconsistencies between the student submission and the tutor solution encompass the same types of issues as identified in figures 2 and 3. Figures 11, 12 and 13

represent different models for how an automated marking scheme can generate a grade for consistency between the student code and student design when there is no mark scheme for this supplied by the tutor. Figures 14, 15 and 16 represent different models for consistency between the student code and student design when the tutor has supplied a consistency mark scheme.

4.0 A Summary of Research Challenges Identified

The table below lists some of the research challenges that arise out of the development of the proposed framework.

Model	Research Challenges
Generic to all	• How do you represent a mark scheme • How do you apply a mark scheme to the student submission • How do you represent the code, the design and the mark scheme in order to compare and traverse between each component.
Figure 2	• How do you identify the key components in the student design • How do you handle erroneous or missing data
Figure 3	• Separation of run-time behaviour from static code structure and the marking thereof • What metrics do you apply to determine a 'good' static code structure • How do you test the run-time behaviour of the student code and how do you handle partial completion/run-time errors
Figure 4	• In what format do you require the student design • How do you take a student design and forward engineer it to produce the inferred code structure • What data structures do you need to model the inferred code structure with a view to comparing it to the student code
Figure 5	• How do you take the student code and reverse engineer it to produce the inferred design structure • What data structures do you need to model the inferred design structure with a view to comparing it to the student design.

Model	Research Challenges
Figure 6	• How do traverse between/compare the student design and the inferred design structure • How do you represent/handle inconsistencies identified between the student design and the inferred design structure • How do you represent/handle consistencies identified between the student design and the inferred design structure • How do you/should you report these back to the student in a meaningful manner
Figure 7	• How do you traverse between/compare the student code and the inferred code structure • How do you represent/handle inconsistencies identified between the student code and the inferred code structure • How do you represent/handle consistencies identified between the student code and the inferred code structure • How do you/do you report these back to the student in a meaningful manner
Figure 8	• Once the consistencies/inconsistencies have been identified through a comparison of the student design with the inferred design structure and the student code with the inferred code structure how do you triangulate between the two views. • In triangulating between the two views what do you do if they are contradictory.
Figure 9	• When constructing the inferred code structure from the student design, is it possible to use the tutor supplied code and mark scheme to produce a mark scheme for the inferred code structure. • How do you separate inferred static behaviour from inferred dynamic behaviour when producing a mark scheme for the inferred code structure
Figure 10	• When constructing the inferred design structure from the student code, is it possible to use the tutor supplied design and mark scheme to produce a mark scheme for the inferred design structure.

Model	Research Challenges
Figure 11	• How do you identify the key components in the student design • How do you handle erroneous or missing data • How do you traverse between the student design, the inferred design structure and the inferred design mark scheme • How do you determine a mark for the student design based upon the inferred design structure and the inferred design marking scheme
Figure 12	• How do you identify the key components in the student code • How do you distinguish between the marking of static and dynamic behaviour utilising an inferred code mark scheme • How do you traverse between the student code, the inferred code structure and the inferred code mark scheme • How do you determine a mark for the student design based upon the inferred design structure and the inferred design marking scheme
Figure 13	• Once the consistencies/inconsistencies have been identified through a comparison of the student design with the inferred design structure and the student code with the inferred code structure how do you triangulate between the two views against the backdrop of those issues identified in figures 11 and 12. • In triangulating between the two views what do you do if they are contradictory
Figure 14	• How do you represent a marking scheme that focuses upon consistency between the student code and the student design. • What should the format of this marking scheme be
Figure 15	• How do you allocate a grade to the student design based upon the inferred design structure and the tutor supplied mark scheme focusing upon consistency. • How do you identify the components contained in the student design • What do you do with erroneous components • What do you do when components are missing from the student submission

Model	Research Challenges
Figure 16	• How do you allocate a grade to the student code based upon the inferred code structure and the tutor supplied mark scheme focusing upon consistency. • What do you do with erroneous code contained in the student code • What do you do with missing code that should be contained in the student code • How do you manage the distinction between the analysis of static code and its dynamic behaviour.

5.0 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)

Annexe

Glossary of Terms

Inferred Code	What the structure of the student submitted source code should be as determined by reverse engineering the accompanying student design
Inferred Design	What the structure of the student submitted design should be as determined by forward engineering the accompanying student source code
Student Code	The program submitted by the student for electronic assessment
Student Design	The design submitted by the student for electronic assessment
Inferred Design Mark Scheme	A marking scheme used to assess the student design that has been generated from analysing the inferred design with the tutor supplied design with mark scheme.
Inferred Code Mark Scheme	A marking scheme used to assess the student code that has been generated from analysing the inferred code with the tutor supplied design with mark scheme
Tutor Supplied Code with Mark Scheme	Source code produced by the tutor that represents a model solution to the program with an accompanying mark scheme indicating how marks have been attributed to individual sections of the model solution
Tutor Supplied Design with Mark Scheme	Design produced by the tutor that represents a model solution with an accompanying mark scheme indicating how marks have been attributed to individual sections of the model solution.
Forward Engineer	The process of taking the student design and producing an idealised structure for the accompanying student code
Reverse Engineer	The process of taking the student code and producing an idealised structure for the accompanying student design
Tutor Supplied Mark Scheme Focusing Upon Consistency	A marking scheme supplied by the tutor that indicates how marks are to be awarded for consistency between the student code and the student design