Personalized Assessment as a Means to Mitigate Plagiarism

Sathiamoorthy Manoharan

***Abstract*—Although every educational institution has a code of academic honesty, they still encounter incidents of plagiarism. These are difficult and time-consuming to detect and deal with. This paper explores the use of personalized assessments with the goal of reducing incidents of plagiarism, proposing a personalized assessment software framework through which each student receives a unique problem set. The framework not only auto-generates the problem set but also auto-marks the solutions when submitted. The experience of using this framework is discussed, from the perspective of both students and staff, particularly with respect to its ability to mitigate plagiarism. A comparison of personalized and traditional assignments in the same class confirms that the former had far fewer observed plagiarism incidents. Although personalized assessment may not be cost-effective in all courses (such as language courses), it still can be effective in areas such as mathematics, engineering, science, and computing. The paper concludes that personalized assessment is a promising approach to counter plagiarism.**

***Index Terms*—Academic dishonesty, automated grading, learning environment, student experience, student assessment, plagiarism**

I. Introduction

Large classes require innovative approaches to both teaching and learning. One issue in large classes is plagiarism. Almost every academic institution has a code of honour and enforces some disciplinary action when that code is violated, but nevertheless plagiarism is still a problem to be dealt with.

This paper examines the effectiveness of using personalized assignments to counter plagiarism. To be useful in practice, automation is essential; assignments need to be automatically generated, and the submitted solutions to the assignments need to be auto-marked. Without auto-generation and auto-marking, personalized assignments entail a large amount of manual work.

The literature includes papers that deal with personalized assignments, but most of these focus on tailoring the assignments to help progressive learning – they do not look at personalization from the plagiarism perspective. This paper addresses this gap, posing the following interrelated research questions on personalized assignments to mitigate plagiarism:

1) Architecture: is there a reasonably generic software architecture that can facilitate personalized assignments to mitigate plagiarism?
2) Plagiarism mitigation: can personalization help mitigate plagiarism?
3) Effectiveness: under what circumstances are personalized assignments effective?

These questions have been partially answered in some related work [1], [2], [3]. The work described in this paper is built upon those studies to provide (a) a more generic programmable software architecture, and (b) a detailed study of the effect of personalization on plagiarism mitigation supported by empirical results. This was done by building a software framework that enables automated generation of personalized assignments and automated marking of submitted solutions. The goal was that each student would receive a unique assignment, different to any other student's assignment, that would necessitate a unique solution. Simply copying a solution from another student would therefore serve no purpose. This framework was evaluated in large classes; the results of this are presented here.

The rest of this paper is organized as follows: Section II discusses automated assessments and plagiarism, drawing from related work. Section III presents the design of a service-oriented software framework to distribute personalized assignments, and compares the proposed framework to two other existing frameworks. Section IV evaluates the framework in the light of its ability to mitigate plagiarism from the perspective of students and staff who used the system. Section V shares the experience of using the system, and its advantages and disadvantages. The final section summarizes and concludes the paper.

II. Background and Related Work

A. Automated Assessments

One of the time-consuming aspect of teaching large classes is how assessments are delivered and graded [4]. The efficiency of assessments and their associated feedback can be increased through automation, helping instructors as well as students. From an instructor's perspective, automation helps in handling large classes, requiring substantially less manual processes than traditional methods [5]. From a student's perspective, automation delivers the results of the assessment and feedback fast. In addition, depending on the level of sophistication, automation can help identify the areas where the student is weak, and strengthen those areas through personalized and targeted assessment [5], [6], [7]. Almost all of the automated assessments are delivered online, allowing the students to work from anywhere [1], [2], [8].

Components of an automated assessment include:

S. Manoharan is with the Department of Computer Science, University of Auckland, Auckland, New Zealand e-mail: s.manoharan@auckland.ac.nz.

Manuscript received January 13, 2016; revised June 10, 2016; revised August 19, 2016; accepted August 23, 2016.

a) Generation of assignments: Assignments can be "static" – in the sense that every student gets the same assignment – or they can be dynamically generated. For example, Problets [2] and OASIS [1] dynamically generate course work from a large database of questions. Problets is used for generating programming exercises while OASIS is used for generating quizzes. Both systems use parameterized exercises so that repeated attempts generate different (pseudo-random) values for the parameters.

b) Personalization of assignments: This is technically a part of dynamic assignment generation, where student-specific details are used in the generation process. Personalization is valuable for students with special needs. These systems can adapt to the student's needs, generating progressively harder problems as the student progresses with her learning. For example, Li and Chen [9] outline a web-based personalized framework where decision trees are used to select exercises at the right level of difficulty for the target student. Similarly, Zare [6] presents a personalized system for mobile learning.

c) Assignment dropbox: Traditionally, students drop paper copies of the assignment solutions into boxes. Most automated systems have an electronic drop-off system that accepts electronic submissions, and issue a receipt when the submission is successful. Some can accept multiple submissions until the due date. Some allow rules to be specified for the submission (such as file types and file names). See Voce [10] for a discussion on online submission systems and policies around these systems.

d) Automated marking: Automated marking is not only cost-effective for the instructors, but also provides quick feedback to the students. With automated marking in place, automated assignment generation can be used to produce practice assignments prior to distributing graded assignments. OASIS [1] and CodeRunner [8] are examples of systems that provide automated marking. CodeRunner [8], for instance, is specifically targeted for programming assignments where small sections of code are written by students. Its marking automation is based on test cases for the code, and the instructor adds several test cases; when a test case passes, the student receives the marks allocated for this test case.

e) Mark distribution: The marking process provides a mark as well as feedback on each student's submission. The marks and feedback are given instantaneously in a number of systems (as long as the systems are configured so). The results are typically published on the educational institution's learning management system, and often students are emailed their results.

B. Plagiarism

Another time-consuming aspect of large classes is detecting and dealing with plagiarism. Almost all academic institutions have a code of academic honesty [11], [12], [13], but it is estimated that about 70% of students admit to some cheating [12]. Lang [14] argues that some of the factors that induce plagiarism come from the learning environment itself, with there being a high possibility that the following environments may encourage plagiarism:

- performance-oriented environments,
- environments where motivation is not intrinsic, and
- environments that use high-value infrequent assessments rather than low-value frequent assessments.

Detecting plagiarism is an active area of research and a profitable area of commercialization [15], [16], [17], [18], [19], [20], [21]. Plagiarism detection tools should be robust enough to mitigate potential cheating of the tools themselves [22].

While there are a number of innovative techniques to speed up the detection of possible plagiarism, dealing with detected plagiarism cases is still time consuming. When possible plagiarism is signalled, instructors need to conduct interviews (occasionally multiple interviews), gather evidence, and fill out administrative forms. Dealing with plagiarism is therefore an expensive affair. Assuming around two hours of interviews and evidence gathering per student, and assuming that 10% of a large class of, say, 400 students were caught cheating, an instructor may have to spend about 80 hours overall in dealing with plagiarism.

Some academic institutions, such as the University of Virginia, delegate the authority to try identified plagiarism cases to a student body [23]. However, in many other institutions, staff run the disciplinary committees that try plagiarism cases. Either way, these trials take a lot of time, and are not normally run by people with a legal background to judge the case.

Another, seemingly more efficient, approach to countering plagiarism is to reduce the factors that induce it [24], [25]. Carpenter et. al. [26] obtained, through student surveys, students' perspective of reducing plagiarism. They point out that fair assessments, academic care, open-book tests, and group work encourage students not to cheat. Born [3] lists further strategies to minimize plagiarism, including personalized assessments. It is this strategy of exploiting personalized assignments that this paper explores through the construction and use of a software framework that supports the strategy.

III. Software Framework

This section discusses the development of a software framework based on Born's [3] observation: personalization minimizes plagiarism. The section first reviews Problets [2] and OASIS [1], two related frameworks, and argues for a more generic personalized system. Problets is used in introductory programming courses while OASIS is used for delivering quizzes.

In Problets [2], programming exercises are chosen from a large database of templated questions. The templates allow randomizing components such as variable names, function names, types and constant values of a program fragment. Fig. 1 shows a short example of a template, and a program generated from this template. The exercises are based on such generated programs and may question the observable characteristics of a program (such as its output).

A similar approach is taken by OASIS [1] where questions are chosen from a large database. The questions are parameterized so that at the time of the delivery of the question, these parameters are replaced with appropriate pseudo-random values. Fig. 2 illustrates a sample question in which the resistance values R1, R2, and R3 are pseudo-randomly generated.

```
{< T0 >< F0 > (){
< T1#integer#>< V1 >;
< T1#integer#>< V2 =< R2#integer; 3 <= R2 <= 8;#>;
...
}}
```

```
void main {
    int val;
    int count = 5;
    ...
}
```

Fig. 1. A sample Problet template and a corresponding C program

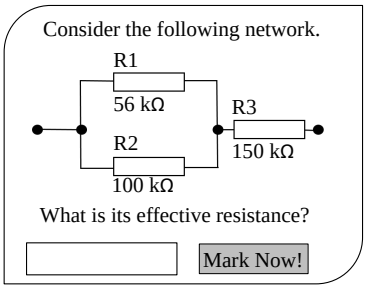

Fig. 2. A sample OASIS question

Both Problets and OASIS use parameterized exercises so that repeated attempts would generate different (pseudo-random) values for the parameters. Such randomization would ensure that no student would get the same exercise, making it impossible to blindly copy answers.

The software framework described in this section takes this a step further. Each assignment is programmatically generated and thus the assignments can be as complex as required. This allows the framework to be used in higher-level classes as well as introductory classes. The framework uses the student's identity to generate problem sets, allowing a problem set to be, if required, idempotent – that is, repeated attempts by the same student would generate the same problem set.

The remainder of this section describes the design of this framework, including the rationale for some of the design choices.

A. Service-Oriented Architecture

Generation and distribution of assignments can naturally be thought of as a service. Consequently, the design of assignment generation and distribution follows a service-oriented architecture (SOA) [27]. A service-oriented architecture is defined as a "set of components which can be invoked, and whose interface descriptions can be published and discovered" [28]. The service interface defines the operations of the service as well as the object types and formats used in conjunction with these operations [29], [30].

B. Service Operations

The service needs to have two roles: a student role and a staff role. A student will be able to obtain assignments and their corresponding answer sheets. Staff should be able to obtain, for a given student, a copy of the student's assignment and the answer sheet pre-filled with the correct answers, and be able to grade a student's submission.

The assignment service therefore needs the following student-facing service operations:

1) Generate a named assignment, and
2) Generate a blank answer sheet for a named assignment.

It also needs the following staff-facing operations:

1) Generate a named assignment for a given student,
2) Generate the answer sheet for a named assignment for a given student, and
3) Grade a given student's submission.

The reason staff require copies of a student's assignment and answer sheet is to offer help when required. It may also be desirable for the staff's copies of the assignment and answer sheet to be annotated with answers.

Additionally, it is important to have a staff-facing service operation that bulk-grades students' submissions and gives them their results. This bulk-grading service operation can provide just the mark, or along with the mark give details such as correct answers and feedback. The former is useful for giving students provisional marks, while the latter is useful for the final marks. Provisional (non-final, in that these do not count toward the final grade) marks are discussed further in Section IV.

Final results also need to be exported in a format that can be imported into the educational institution's learning management system.

C. Assignment Personalization and Sign-on

In order to personalize the assignments, a student needs to identify herself before using the service. This is typically achieved by using a sign-on system, such as the Shibboleth single sign-on framework [31]. Once signed on, the student's identity can be transparently picked up and used by the assignment service for personalization. In addition, the sign-on mechanism allows identification of roles (e.g., student or staff).

D. Plugin-Enabled Architecture

One of the important considerations of a generic assignment generation service is customization. Staff should be able to write their own assignment and provide a grading scheme for it. This is achieved in the system by using plugins [32].

Each assignment is an abstract entity with the following contracts:

1) Can generate the assignment specification
2) Can generate an answer sheet
3) Can grade a submission

Staff need to write their own concrete assignments conforming to the above set of contracts. These concrete entities extend (or derive from) the abstract assignment entity and implement the contracts. Note that the development of the assignment requires some programming knowledge, and in the domain

of computational and mathematical sciences and engineering, this is not seen to be an issue.

Once developed, the concrete assignments are optionally peer-reviewed (including functionality and security reviews) and deployed to a plugin location within the service. The service dynamically loads these assignments from the plugin location on demand. The service framework therefore does *not* change as new assignments are developed and deployed.

Assignment: Cryptanalysis

The first part of this assignment familiarizes you with Cryptanalysis by requiring you to decrypt a simple cipher. In the second part, you will decrypt a little more advanced cipher.

Part 1. What is shown below is an intercepted cipher text. What is the plain text corresponding to this, if the encryption is based on a Caesar cipher? [3 marks]

```
Kbdl hsbccfe b mjof gps cbmbodf, evdlfe
bt uif nbjotbjm kfslfe upxbse ijn, uifo
dpoujovfe ijt tupsz, ibsemz njttjoh b
cfbu.
```

Part 2

In this part, we have another intercepted communication. However, this time, it is thought that the communication is encrypted using the Vigenère cipher. What is shown below is the intercepted cipher text. We observe that the characters of the English alphabet are encrypted, and other characters are left alone.

```
Cjq mev rgd cevn cxjro djq meqvkzb evn
ufvvmn fdzeusnk jyb dq eze. Vvgs rwrail
kpp hsdof fjaibf fci jyy icizo Cdvfmvnm
nxwyf eoezsps jyb yxqm xpo darwxbkf. [...]
```

2a. What is the length of the Vigenère cipher's key? Provide the smallest possible numeric value. [3 marks]

2b. What is the plain text corresponding to the cipher text? [4 marks]

Fig. 3. A sample personalized assignment on Cryptanalysis

Figs. 3 and 4 show two of the automated personalized assignments. The encrypted texts in Fig. 3 and the authentication data in Fig. 4 are personalized for each student.

While the answer sheets are typically textual in nature, the assignment specifications may point to non-textual (binary) data. Four of the assignment specifications used in the classes in fact use binary data: one uses personalized image files, another uses personalized SQLite [33] databases, and the other two use personalized network packet capture (PCAP) files. In case of such personalized binary data, the assignment sheet could be static and shared (i.e., non-personalized) but with hyperlinks to the personalized binary data file. A student would then download her own binary data file, process it, and finally submit the answer sheet based on her copy of the binary data. See Fig 5, which illustrates an assignment that involves such personalized binary data. Here the link *this image* points to a personalized (binary) image.

The framework provides the service contracts to distribute assignments and answer sheets, mark assignments, and distribute grading results to students. A concrete assignment, its answer sheet, and grading scheme are developed by the instructor and packaged as a plugin that is deployed to the service framework. The framework allows access to the user ID (such as the single-sign on ID of the student) which can be used to vary the assignment in a suitable way. Typically, the ID is used as part of the seed of a pseudo-random number generator, so that data pertaining to an assignment is pseudo-randomly generated.

For each assignment, the instructor provides a list of the associated staff, and these staff will have all the privileges of a student as well as the access to any student's assignment and its solution. The staff are also allowed to run the auto-marking facility.

Assignment: HTTP Authentication

This assignment aims to familiarize you with some aspects of HTTP's Basic and Digest authentication mechanisms.

Part 1. In this part, you will be working with HTTP's Basic authentication. A man-in-the-middle attack found the following HTTP Authorization header in transit.

```
Authorization: Basic YXBkbzp1cGJz
```

1a. What is the user name as seen in the above Authorization header? [1 mark]

1b. And, what is the password seen in this Authorization header? [1 mark]

Part 2. In this part, you will be working with HTTP's Digest authentication. Assume that accessing a resource "/CS/Home.png" on a web server results in the following (partial) response.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="Mordor",
nonce="03e2abb8a924e966bee59d41cef32851",
opaque="4043168947418128"
```

2a. Assuming that the user name is 6177077 and the password is 7707716, the HTTP Authorization header line sent by the client (e.g., browser) following the above notification for authentication will be of the following form.

```
Authorization: Digest username="6177077",
response="", realm="Mordor",
nonce="03e2abb8a924e966bee59d41cef32851",
uri="/CS/Home.png", opaque="4043168947418128"
```

What should be the value of the "response" field which currently is empty? Please do NOT include the quotation marks (i.e., ") in your answer. You must use lowercase hexadecimal digits in your answer, if applicable. [2 marks]

2b. In order to safeguard against server compromises, the server stores a hash value that includes the password (and other things) and uses this hash value for authentication. In this particular example here, what is the hash value you expect the server to keep in its repository? You must use lowercase hexadecimal digits in your answer, if applicable. [2 marks]

2c. A man-in-the-middle attack found the following HTTP Authorization header in transit.

```
Authorization: Digest username="6177077",
response="624d3e05b6c345ff27028cf3f65a6966",
uri="/CS/Home.png", realm="Mordor",
nonce="03e2abb8a924e966bee59d41cef32851",
opaque="4043168947418128"
```

What is the password that gives rise to the above HTTP Authorization header? The password is thought to be a common dictionary word. [3 marks]

Fig. 4. A sample personalized assignment on HTTP authentication

Assignment: Steganography

In this assignment, you are required to extract the text message hidden in *this image*. The LSB (least-significant bit) of each colour component (R, G, and B) of the pixels of the image potentially contains a bit of the message. The task is to pick these bits to form the hidden message.

The text message hidden in the image employs the Pascal-format. It uses a 32-bit integer to keep the length of the text, and then uses as many bytes as specified by the length to keep the characters of the hidden text.

The message is hidden using LSB Steganography and the pixels for hiding the message are chosen sequentially from Pixel (0,0). The pixels are accessed in a row-major fashion (i.e. accessed row by row rather than column by column).

You will need to first extract the 32 bits that tell the length of the text, and once this is known, you extract the other $8N$ bits where N is the text length previously extracted. You then assemble those $8N$ bits into N bytes to form the text message. Once you have extracted the message, please write it down in the answer sheet.

Fig. 5. A sample assignment on Steganography [34] that uses a personalized binary image. The hyperlink points to the image.

An assignment will need to be of the form $y = f(x_1, x_2, \ldots, x_n)$ where x_1, x_2, etc. represent data that can be pseudo-randomly generated or gathered. For instance, instead of asking a static question "what is the XOR of 0xBE and 0xEF", a dynamic personalized variant such as "what is the XOR of $\$x_1$ and $\$x_2$" can be asked where the values of $\$x_1$ and $\$x_2$ are auto-generated based on the student ID and filled in by the underlying framework. Clearly, this does not cover all possible forms of an assignment, and this limitation and its applicability are discussed later in Section V-C.

IV. Evaluation

This section evaluates the software framework and its use aligned with the research questions posed in Section I and based on the field use of the framework.

A. Architecture

In comparison with Problets [2] and OASIS [1], the software framework presented in this paper is highly programmable, which allows for more sophisticated assignments than that is possible with parameter-based randomization. For instance, consider the Steganography [34] assignment in Fig. 5. The image linked to in this assignment requires:

1) pseudo-randomly selecting, based on a student's ID, an image from a database of images,
2) pseudo-randomly selecting, based on a student's ID, a text message from a database of texts, and
3) using a Steganography algorithm (such as LSB replacement [34]) to embed the selected text message into the selected image.

Such a process cannot be achieved by simple parameter-based randomization and requires a reasonable amount of programming (especially for step (3) above).

Instructors have the ability to make the assignments as complex and varied as they want, aided by the programmability the framework offers. The generic programmability allows the framework to be used in various courses at higher levels. The downside, however, is that the programming effort increases with the complexity of the assignment.

The development and use of the three frameworks, Problets, OASIS, and the one described here, answer in the positive the first research question as to whether a reasonably generic architecture can facilitate personalized assignments.

B. Plagiarism Mitigation

The personalization approach – essentially delivering a different exercise to every student – helps reduce plagiarism: each student gets an assignment or quiz that is different to that of any other student in the class, and thus blindly copying another student's solution is unlikely to earn marks. The documentation on Problets makes a similar observation. This gives a positive answer to the second research question on the likelihood of personalization mitigating plagiarism. This section provides further empirical evidence to support this answer, using the experience from in-class trials and surveys.

The personalized assessment system was first tested in a large second-year class on "Computer Communications", whose enrolment was a little over 360 students. Traditionally, the practical assessment in the class consisted of three manually-marked assignments. One of these was retained, but the other two were replaced with seven smaller automated personalized assignments (three of these are shown in Figs. 3 to 5) which were delivered more frequently and carried fewer marks compared to the manually-marked large assignment. This follows from the earlier note on Lang [14]'s observation: low-value frequent assessments tend to reduce plagiarism.

A correlation check between the traditional manually-marked assignment and the automated personalized assignments show that both types of assessments are positively correlated, with the correlation coefficient being 0.87.

In order to evaluate the effect of personalization on plagiarism, an online survey was conducted of the whole class (of a little over 360 students). The response rate was around 38%.

TABLE I
Student evaluation results of personalized assignments (2015). SD: strongly disagree; D: disagree; N: neutral; A: agree; SA: strongly agree.

Question	SD (%)	D (%)	N (%)	A (%)	SA (%)
Personalization encouraged me to work independently	0	4.6	22.7	34.1	38.6
It would be a good idea to have personalized assignments in other courses	0	4.6	27.3	22.7	45.5
Overall, I like the idea of personalized assignments	0	2.3	31.8	20.5	45.5

The summary of the survey results for assignment personalization is listed in Table I. The results show that a large proportion of the participants are in favour of personalized assignments, and over 70% agree that personalized assignments

encouraged them to work independently. The free-format text feedback from some of those students who remained neutral or disagreed indicated that these students work independently irrespective of whether an assignment is personalized or not.

The survey was repeated the following year in the same course, now with around 370 students. Three new questions were in this new survey, one explicitly asking the students their view on reducing the level of cheating using personalization. One of the questions also asked about their view on low-value, frequent assignments. Table II summarizes the results of the survey where the response rate was around 41%.

TABLE II
STUDENT EVALUATION RESULTS OF PERSONALIZED ASSIGNMENTS (2016). SD: STRONGLY DISAGREE; D: DISAGREE; N: NEUTRAL; A: AGREE; SA: STRONGLY AGREE.

Question	SD (%)	D (%)	N (%)	A (%)	SA (%)
Personalization encouraged me to work independently	0	0	20.3	40.6	39.1
Personalized assignments ensure everyone needs to work equally hard to achieve good scores	1.4	2.9	23.2	42.3	30.4
Personalized assignments reduce the incidents of plagiarism and the level of cheating	0	5.8	15.9	33.3	43.4
I prefer low-value, high-frequency assignments to high-value, low-frequency assignments	0	7.3	17.4	29	46.4
It would be a good idea to have personalized assignments in other courses	0	2.9	23.2	36.2	37.7
Overall, I like the idea of personalized assignments	0	3	22.7	34.8	39.4

The results of the second survey are quite similar to the first survey, with a clear majority of the students stating that personalization encouraged them to work independently. Over 75% of the students thought personalization reduces the incidents of plagiarism and the level of cheating.

In addition, this second survey included four questions related to student-observed plagiarism. The first question was "With respect to these personalized assignments, do you know of instances where someone violated the academic honesty policy? For example, do you know of someone who solved the assignment of someone else; or someone who supplied their own code/program/app to solve the assignment?". To those who answered yes, there was a follow-on question: "If you answered yes to the above question, how many different such incidents do you know of (only with respect to the personalized assignments)". Similarly there was another yes/no question for the standard (i.e., non-personalized) assignment asking if the students knew of plagiarism incidents, and a follow-on question (for those who answered yes) asking about the number of incidents they knew of. Some 20.2% of the students said they knew of plagiarism cases with personalized assignments; 36.8% of the students said they knew of plagiarism incidents with the standard assignment.

Table III shows the incident counts the students reported in the survey. The survey responses indicate that the students observed more plagiarism incidents in the case of the standard assignment.

TABLE III
STUDENT-OBSERVED PLAGIARISM INCIDENTS (2016).

Incidents	Personalized assignments (%)	Standard assignment (%)
1–4	85.7	23.7
5–10	14.3	14.1
11–20	0	22.5
21–40	0	39.7
41–100	0	0
more than 100	0	0

There were three cases of confirmed plagiarism incidents relating to the seven personalized assignments. Here, students had anonymously posted their assignment questions to online forums, seeking answers[1].

The single standard assignment, on the other hand, had 28 cases of confirmed plagiarism incidents.

The system was also used in a large third-year computer science class with just over 310 students. This class traditionally had two large programming assignments. One of the programming assignments was replaced with a set of small personalized assignments on software penetration testing. The corresponding large assignment in the previous year had over 10% of detected plagiarism cases (using MOSS [19]). The personalized assignments were guaranteed to have unique answers for each student, so by definition, there is no MOSS-detectable plagiarism. While this is not a direct comparison, it shows that changing the assessment approach can reduce cases of plagiarism, and personalization is a promising approach to this end.

C. Effectiveness

Personalization will not be effective without using an automated marking facility. Since every student's assignment is different, manually marking the assignments would be prohibitively expensive. The software framework therefore incorporates an auto-marker.

Table IV shows a summary of survey results on this auto-marker. The survey questions were administered alongside the survey reported in Table I and use the same student base.

TABLE IV
STUDENT EVALUATION RESULTS OF AUTO-MARKING (2015). SD: STRONGLY DISAGREE; D: DISAGREE; N: NEUTRAL; A: AGREE; SA: STRONGLY AGREE.

Question	SD (%)	D (%)	N (%)	A (%)	SA (%)
Provisional marks from the Auto-marker were useful to check my answers	9.1	2.3	13.6	29.6	45.5
The format of the feedback generated by the Auto-marker was acceptable	15.9	9.1	18.2	31.8	25

One of the issues with auto-marking is that the answer format required can be quite stringent. One approach to

[1]Given that the assignments were unique, it was easy to find and track the offenders.

manage this is to have low-value assignments and have small sub-questions in each assignment where the expected format of the answer is well-specified. In this case, if a student makes a simple careless formatting error in the answer to one of these questions, the effect on the overall mark is likely to be quite small. Besides, distributing provisional marks allows the students to correct formatting errors.

Provisional (non-final) marks are just numeric mark values and do not include the correct answers or any other feedback. They help students to correct not only their logical errors but also formatting errors and re-submit their solutions. The survey shows that the students appreciated having provisional marks. Provisional marks were only provided once, so that the students cannot repeatedly use the facility as an oracle to arrive at solutions. The final marks included correct answers as well as some feedback. The survey shows that a number of students would like to see the format of the final marks and the feedback improved.

Personalization can only be effective for assignments that can be formulated as a function of some data that can be pseudo-randomly generated or chosen. Many science and engineering courses have learning outcomes that fit this paradigm. However, there needs to be substantial time invested in finding how these learning outcomes can be translated into testable assignments.

To this end, feedback on the effectiveness of the system was sought from six Science faculty staff who reviewed and used the system. Results of a short (non-anonymous) feedback from these six staff are shown in Table V. The numbers in the table are respondent counts for a given rating.

TABLE V
STAFF FEEDBACK ON USING THE FRAMEWORK.
RATING SCALE – 1: VERY POOR; 5: VERY GOOD.

Question/Rating	1	2	3	4	5	Average rating
Ease of use			2	2	2	4.0
Time/cost saving (compared to traditional assignments)			2	1	3	4.2
Ability to reduce plagiarism incidents					6	5.0
Overall usefulness				1	5	4.8

Overall, the staff were positive about using the system. Ease of use scored the lowest average rating, mainly due to the programming effort required to set up an assignment. Staff felt that they needed two to seven days to create an assignment, with the average of around five days. Most of this time is spent on planning and preparation required to re-shape traditional assignments. Yet the staff appreciated such efforts were one-time only, for the assignments can be re-used for a while. They also noted that it was more productive to spend time developing personalized assignments than to spend time dealing with a large number of plagiarism incidents.

So far, five courses have successfully used the system, two in the second year and three in the third year, in computer science and engineering areas.

V. DISCUSSION

This section covers a number of discussion points that arose from the experience using the personalized assessment framework within a large class.

A. Architecture

The personalized assignment framework requires instructors to write some code. In mathematics, science, and engineering, where the framework finds the most applicability, this is not seen as a limiting factor. The framework currently provides a library of common tasks related to assignment generation so as to minimize the amount of code an instructor needs to write. In addition, the assignments themselves become libraries in some sense, because they can be re-used time and again.

B. Plagiarism Mitigation

The assignments are generated programmatically, and the students could equally write programs to solve the assignments and share the programs even though any such sharing is explicitly prohibited under the honour code.

The value of writing programs to solve most of these assignments is very low, since most of these assignments are solvable much more quickly by hand and/or using online tools. For example, there are online tools to solve the Cryptanalysis questions depicted in Fig. 3, as well as the HTTP authentication questions depicted in Fig. 4. The learning objectives of the assignments were to use manual methods and online tools to understand the solution process. This was made clear to the students. Each assignment was only worth 2% of the overall course mark, and consequently a solution involving extensive programming was not worth the students' while.

Not only did the personalized assignments encourage students to work independently, but the low-value and relative ease of the assignments discouraged them from plagiarism. The author believes that the right direction to combat plagiarism is to set assessments that:

1) are personalized
2) have low value
3) are easy to solve
4) are delivered frequently

The last three of these points were already noted and supported by Lang [14]. Low-value and easy-to-solve assignments encourage and motivate weak students. The high frequency of assignment delivery means that the topics covered in the week can be practised in an assignment in the following week. Weak students particularly appreciate this because they are quickly able to apply the concepts they learned and consolidate their understanding.

It is quite possible that a student can hire someone (i.e., a ghostwriter) to do their assignments. This is a case that can only be tackled by an in-house examination rather than a take-home assignment. Personalization of assignments only helps to mitigate blind copying of solutions from someone else.

C. Effectiveness

Personalized assignments work very well in a function evaluation scenario: evaluating the results of a function with a number of parameters. Each parameter could be randomly

chosen for each student. This scenario finds applicability in mathematics, science, and engineering.

Personalized assignments are costly if the marking is to be manual (e.g., marking critiques, essays, etc.).

VI. Summary and Conclusions

This paper addressed how personalized assessments could mitigate plagiarism, an issue that commonly plagues large classes.

Personalized assessment requires automation to be manageable: assignments need to be automatically generated and graded. The paper established that it is possible to construct a reasonably generic software framework to support such automation. The framework reported in the paper uses a service-oriented architecture to generate, distribute and grade assignments, and uses plugins to facilitate adding new assignments.

The system was tested in several large second- and third-year university classes. The evaluation results indicate that the students liked the system. Over 70% of the students agreed that personalized assignments encouraged them to work independently and reduce the incidents of plagiarism. The experience of using the system also indicates that the personalized assignments have far fewer plagiarism incidents compared to traditional assignments. Personalized assignments had three incidents over seven assignments, while a traditional assignment had 28 incidents just for a single assignment. This suggests that employing personalized assignments is a promising approach to mitigate plagiarism.

While personalized assessments cannot be cost-effective in some large courses (e.g., language courses) where personalization may require manual processes, they can still be quite effective in some other large courses in areas such as mathematics, engineering, science, and computing. A lot of thought and effort is needed to form good personalized assignments that meet the learning outcomes of a course. However, once created, these assignments can be used for a long time.

References

[1] C. Smaill, "The implementation and evaluation of OASIS: a web-based learning and assessment tool for large classes," *IEEE Transactions on Education*, vol. 48, no. 4, pp. 658–663, Nov 2005.

[2] A. N. Kumar, "Using problets for problem-solving exercises in introductory C++/Java/C# courses," in *2013 IEEE Frontiers in Education Conference (FIE)*, Oct 2013, pp. 9–10.

[3] A. D. Born, "How to reduce plagiarism," *Journal of Information Systems Education*, vol. 14, no. 3, pp. 223–224, 2003.

[4] A. Margaryan, M. Bianco, and A. Littlejohn, "Instructional quality of massive open online courses (MOOCs)," *Computers & Education*, vol. 80, no. 0, pp. 77 – 83, 2015.

[5] R. Kvadsheim, M. Mehlen, and H. Kittang, "Web-based automatic feedback on assignments in statistics: How can it help students learn statistics and universities reduce costs?" *International Journal of Engineering Education*, vol. 26, no. 3, pp. 642–654, 2010.

[6] S. Zare, "Personalization in mobile learning for people with special needs," in *Universal Access in Human-Computer Interaction. Applications and Services*, ser. Lecture Notes in Computer Science, C. Stephanidis, Ed. Springer Berlin Heidelberg, 2011, vol. 6768, pp. 662–669.

[7] L. Cheniti-Belcadhi, N. Henze, and R. Braham, "Implementation of a personalized assessment web service," in *Sixth International Conference on Advanced Learning Technologies*, July 2006, pp. 586–590.

[8] R. Lobb and J. Harlow, "Coderunner: A tool for assessing computer programming skills," *ACM Inroads*, vol. 7, no. 1, pp. 47–51, Feb. 2016.

[9] L. Li and G. Chen, "A coursework support system for offering challenges and assistance by analyzing students' web portfolios," *Educational Technology & Society*, vol. 12, no. 2, pp. 205–221, 2009.

[10] J. Voce, "Reviewing institutional policies for electronic management of assessment," *Higher Education*, vol. 69, no. 6, pp. 915–929, 2015.

[11] P. Melgoza and J. Smith, "Revitalizing an existing honor code program," *Innovative Higher Education*, vol. 32, no. 4, pp. 209–219, 2008.

[12] M. Broeckelman-Post, "Faculty and student classroom influences on academic dishonesty," *IEEE Transactions on Education*, vol. 51, no. 2, pp. 206–211, May 2008.

[13] Tracey Bretag , Ed., *Handbook of Academic Integrity*. Springer, 2016.

[14] J. M. Lang, *Cheating Lessons: Learning from Academic Dishonesty*. Harvard University Press, 2013.

[15] I. Koss and R. Ford, "Authorship is continuous: Managing code plagiarism," *IEEE Security Privacy*, vol. 11, no. 2, pp. 72–74, March 2013.

[16] C. Kaner and R. Fiedler, "A cautionary note on checking software engineering papers for plagiarism," *IEEE Transactions on Education*, vol. 51, no. 2, pp. 184–188, May 2008.

[17] A. Schmidt *et al.*, "A concept for plagiarism detection based on compressed bitmaps," in *Proceedings of the Sixth International Conference on Advances in Databases, Knowledge, and Data Applications*, April 2014, pp. 30–34.

[18] T. Ohmann and I. Rahal, "Efficient clustering-based source code plagiarism detection using PIY," *Knowledge and Information Systems*, vol. 43, no. 2, pp. 445–472, 2015.

[19] K. Bowyer and L. Hall, "Experience using "MOSS" to detect cheating on programming assignments," in *29th Annual Frontiers in Education Conference*, vol. 3, Nov 1999, pp. 13B3/18–13B3/22 vol.3.

[20] S. Schleimer, D. Wilderson, and A. Aiken, "Winnowing: Local algorithms for document fingerprinting," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, June 2003.

[21] B. Gipp, *Citation-based Plagiarism Detection – Detecting Disguised and Cross-language Plagiarism using Citation Pattern Analysis*. Springer Vieweg Research, 2014.

[22] C. Collberg, G. Myles, and M. Stepp, "Cheating cheating detectors," Department of Computer Science, University of Arizona, Tech. Rep. TR04-05, March 2004.

[23] The Honor Committee, *The Honor Newsletter*. The University of Virginia, February 2015, vol. 5.

[24] N. C. Heckler, "Mitigating plagiarism in large introductory courses in higher education," Ph.D. dissertation, The University of Alabama, 2012.

[25] N. C. Heckler, D. R. Forde, and C. H. Bryan, "Using writing assignment designs to mitigate plagiarism," *Teaching Sociology*, vol. 41, no. 1, pp. 94–105, 2013.

[26] D. Carpenter, T. Harding, C. Finelli, S. M. Montgomery, and H. J. Passow, "Engineering students' perceptions of and attitudes towards cheating," *Journal of Engineering Education*, vol. 95, no. 3, pp. 181–194, 2006.

[27] T. Erl *et al.*, *SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST*. Prentice Hall Press, 2012.

[28] H. Haas and A. Brown, *Web Services Glossary*, February 2004, W3C recommendation.

[29] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[30] J. Webber, *REST in Practice: Hypermedia and Systems Architecture*. O'Reilly, 2010.

[31] K. Takaaki, S. Hiroaki, D. Noritoshi, and M. Ken, "Design and implementation of web forward proxy with Shibboleth authentication," in *IEEE/IPSJ 11th International Symposium on Applications and the Internet*, July 2011, pp. 321–326.

[32] R. Chatley, S. Eisenbach, and J. Magee, "Modelling a framework for plugins," in *Workshop on Specification and verification of component-based systems*, September 2003.

[33] J. A. Kreibich, *Using SQLite*. O'Reilly Media, 2010.

[34] N. Johnson and S. Jajodia, "Exploring steganography: Seeing the unseen," *IEEE Computer*, vol. 31, no. 2, pp. 26–34, February 1998.

Sathiamoorthy Manoharan is a senior lecturer in the Department of Computer Science, University of Auckland. He received the Ph.D. degree in computer science from the University of Edinburgh, Scotland.