# Will Johnny/Joanie Make a Good Software Engineer?: Are Course Grades Showing the Whole Picture?

Jane Huffman Hayes[§], Alex Dekhtyar[§], Ashlee Holbrook[§], Olga Dekhtyar[¶], Senthil Sundaram[§]
[§] Dept. of Computer Science,
*{hayes,dekhtyar}@cs.uky.edu,{ashlee,skart2}@uky.edu*
University of Kentucky
[¶] Dept. of Statistics,
*olga@ms.uky.edu*
University of Kentucky

**Abstract**

*Predicting future success of students as software engineers is an open research area. We posit that current grading means do not capture all the information that may predict whether students will become good software engineers. We use one such piece of information, traceability of project artifacts, to illustrate our argument. Traceability has been shown to be an indicator of software project quality in industry. We present the results of a case study of a University of Waterloo graduate-level software engineering course where traceability was examined as well as course grades (such as mid-term, project grade, etc.). We found no correlation between the presence of good traceability and any of the course grades, lending support to our argument.*

## 1. Introduction

When a student graduates and interviews for a position within industry, potential employers are provided a résumé, grades, references, and the interview itself as measures to judge how well the student will perform. In the area of software engineering, potential employers may review a number of indicators of potential success such as grades of relevant courses, and grades of individual tests or assignments within certain courses. While student grades have long served as an indicator of future success, we argue that important aspects of student ability in software engineering are NOT being captured as part of the grading process. To illustrate our argument, we examine the ability of software engineering students to build traceable artifacts in course projects. Traceability is defined here as the *degree to which individual elements within the artifact can be connected with matching elements of other artifacts.* Traceability of generated artifacts embodies the ability of the student to complete the software life cycle and could serve as an indicator of their future success as a software engineer.

One could argue that if we are not producing good software engineers as a result of our many efforts in software engineering education and training, we are not succeeding. Yet, research addressing the problem of ensuring that the students we teach end up being productive software engineers is scarce (see Related Work section). There are two main ways to assess the potential of a software engineering student: direct and indirect (see Figure 1). Direct means include interviewing the employer of a student after the student has been hired and has been working for some time and asking the employer to assess the skills of the student. Indirect means include two categories: course grades and derived measures. Course grades can further be divided as practical/hands-on measures or knowledge-based measures. An example of a knowledge-based course grade would be the score on a mid-term or final. A practical course grade would be the grade for a software engineering project or artifact.

A derived measure is one that has been developed using properties of artifacts developed by the students as part of their coursework. Evaluation of these artifacts may or may not be part of the course grade. In this paper, we examine one such derived measure, *traceability of a project*, and look to see whether it correlates with the typical course grades being collected in software engineering courses. If it does not, there is an indication that the grading process is not necessarily capturing all the information that a future employer might need.

In recent years, researchers studying industry practice have concluded that traceability is among the most important qualities of software projects. For example, Egyed states that "*traces are the 'blood vessels' of [software] models*" [10]; Dömges and Pohl claim that "*requirements traceability is a prerequisite for effective system maintenance and consistent change integration*" and that "*neglecting traceability … leads to a decrease in system quality, causes revisions, and thus, increase in project costs and time*" [6,9], and Ramesh et al. claim that traceability is a way of "*showing compliance with requirements, maintaining system design rationale, showing when the system is complete and establishing change control and maintenance mechanisms*" [19]. It is a widely held belief in industry that traceability "*is needed for the successful completion of a project and that without it, their organization's success would be in jeopardy"* [19].Traceability is a requirement for large mission-critical software projects within such organizations as U.S. Department of Defense and NASA [21,19].

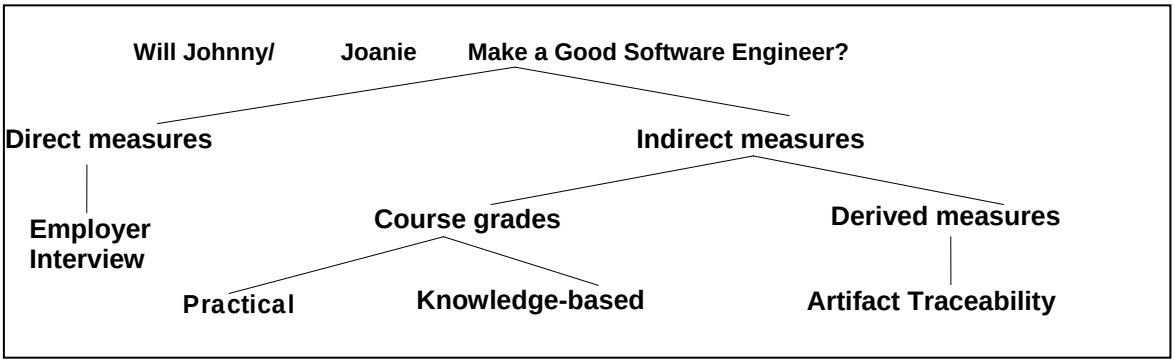


**Figure 1. Measures to assist in Software Engineer success prediction.**

The development of software project artifacts in such a way that they are easily traceable has a very important tie to a student's potential success as a software engineer. Arguably, many courses of the Computer Science curriculum teach students how to code well. It is the role of the software engineering courses to teach future software engineers how to successfully develop other artifacts necessary for the project life cycle. One of the key features that distinguishes well-written artifacts is *traceability*. The traceability information for the project artifacts is usually stored in the *Requirements Traceability Matrix* (RTM). In developing the software engineering artifacts (such as software requirements specifications, design specifications, UML diagrams, etc.) to eventually yield a developed product, the RTM is the roadmap or the proof of the path that was taken to the solution. The traceability of project artifacts indicates how easy (or how hard) it is to build the RTM for the project. It is therefore desirable that in software engineering classes students learn how to write traceable artifacts. Two questions need to be addressed – do students create traceable artifacts in their projects and is the traceability of artifacts reflected in the student grades?

In this paper, we describe a case study which supports our conjectures that course grades do not reflect artifact traceability. We have studied 22 group projects produced by students in a graduate-level software engineering course. Using a requirements tracing procedure we have established earlier in [12], we have measured the traceability of the projects and analyzed the relationship of the derived measures with the student grades. We have discovered no significant relationship between various student grades and different traceability measures we have used.

The paper is organized as follows. Section 2 presents related work in Software Engineering education. Section 3 describes our case study, including the research hypothesis, case study design, methods, analysis results, etc. Section 4 presents conclusions and future work.

## 2. Related Work in Software Engineering Education

While several papers have been published on how to evaluate individual efforts within group projects [3,5,13], examining students' future success as software engineers is an open research area. Initial studies have shown that grades from programming courses may indicate whether a student has mastery of programming in a particular language or discipline, but are not applicable to predictions of future success [8]. Such a measure, however, is very different than predicting future success as a software engineer. In order to be a successful software engineer, a student must be able to work on a project throughout the software life cycle, specifying correct requirements, translating those requirements into design, and then coding and testing the product.

Many agree that grading serves a key role in the educational process. Walker [22] notes that student evaluation serves two purposes: (a) to provide feedback to students on progress, and (b) to assign grades to students. Numerous authors have outlined grading criteria for computer science and software engineering courses [15,17,22,14]. Measures such as Attitude Toward Software Engineering (ATSE) have also been examined as ways to judge software development expertise [7].

Several studies have been performed on predicting success in a Computer Science course or major, particularly in early CS courses [18,23,16,20]. Alexander et al. performed a case study on predicting future success in a college computer science curriculum based on high school experiences and grades. While they found that better grades overall were preferable, they did not find a strong correlation between particular grades and later successful completion of a Computer Science major. Just as there is a fundamental difference between high school work and college work, there are critical factors of success as a software engineer in industry that current college grading schemes do not fully capture [1]. Chmura additionally found no correlation between high school grades and future success in Computer Science/Software Engineering coursework [4].

## 3. The Case Study

### 3.1. Case Study Context

While the importance of traceability is widely-recognized, creation, maintenance and validation of RTMs in industry is still largely performed manually and is very labor-intensive [12]. Our studies have shown that tools, employing traditional Information Retrieval (IR) techniques [2] for building candidate RTMs, can outperform human analysts [11] and can produce with user feedback[2], fairly accurate candidate RTMs. Since the methods we considered in [12,11] have been validated, we can apply our methods to measure traceability. Our view is that the more traceable the project artifacts are, the easier it should be for an automated tracing method to construct an accurate RTM. In particular, we can apply the automated tracing methods to measure the traceability of student project artifacts.

The objective of our case study is to examine how well typical assessment methods in software engineering courses predict the potential success of the student in the future. Note that we cannot draw any general conclusions from our case study as there was no random assignment of subjects to objects and it was not a controlled experiment. Also, results are presented as descriptive statistics that can potentially serve as indicators. We are using the project grades for a University of Waterloo software engineering course as our baseline. We were constrained by not having access to all the project artifacts or to any demographic information about the students (as their total privacy was maintained). This also precluded us from knowing what percentage of the work was performed by what student, or what tasks each student performed.

### 3.2. Case Study Planning and Validation

For our study, we used the student projects of the University of Waterloo graduate level software engineering class (January 2005) as our experimental subjects, and specifically used the artifacts and grades as the objects of study. The course curriculum was typical of the graduate courses in software engineering, with traceability getting only cursory mention. Measurements were taken by University of Waterloo faculty as the course progressed. These included: mid-term grade, project grade, final examination grade, and course grade. The projects were performed by groups of three or four students, and the course policy was to award each student in a group the same grade for the project. All other grades were individual for each student. We performed our study of these measures after the course had completed. In addition, we generated some derived measures related to traceability that will be described below.

A total of one hundred and thirty three (133) students were enrolled in the course and a total of thirty five (35) groups were organized. Twenty eight (28) groups consisted of four students while seven (7) groups consisted of three students. We have obtained the requirements and use case documents for the groups as well as the requirements traceability information for the two documents. The full RTM was available for only twenty two (22) groups with a total of eighty-five participating (85) students, which were used in the case study. In Table 1, we compare the available grades information for the groups we have used in the study and the groups that were left out of the study. The data in the table indicates that the two groups *did not differ significantly* In terms of grades. In Table 2, we provide a summary description of the project artifacts.

**Table 1. Comparison between the study participants (P) and non-participants (NP).**

| | **Project Grade** | | | **Midterm Grade** | | | **Final Exam Grade** | | | **Course Grade** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | St Dev | t-val p-val | Mean | St Dev | t-val p-val | Mean | St Dev | t-val p-val | Mean | St Dev | t-val p-val |
| **P** | 36.75 | 2.08 | **-.14** | 7.24 | 1.39 | **-1.42** | 36.05 | 7.58 | **-.10** | 80.04 | 9.46 | **-.39** |
| **NP** | 36.81 | 2.28 | **0.89** | 7.59 | 1.37 | **0.16** | 36.18 | 6.58 | **0.92** | 80.65 | 8.39 | **0.71** |

**Table 2. Sizes of Project Artifacts.**

| Number of | Min | Mean | Median | Max | Std. Dev |
|---|---|---|---|---|---|
| Functional Requirements | 17 | 46.18 | 47 | 80 | 16.19 |
| Use Cases | 5 | 17.13 | 17.5 | 30 | 7.90 |
| RTM links | 19 | 55.63 | 48 | 143 | 29.10 |

### 3.3. Measuring the Traceability: Procedures, Measures, Hypotheses

To a large degree, *traceability* identifies the ease of reconstructing the RTM for the project. In [12], we have described RETRO (REquirements TRacing On-target), a software tool for automated construction of RTMs. We use one of RETRO's methods, combined with the simulated analyst feedback procedure, to construct candidate RTMs which are measured for traceability. The construction of a candidate RTM in RETRO proceeds as follows. The high- and low-level documents, broken into individual elements, are parsed and an information retrieval method is run to construct a list of candidate links for each high-level element. This list may contain errors of two types: (a) errors of commission – a false link is included in the list, and (b) errors of omission - a true link is not found in the list. In general, a human analyst working with RETRO must go over the list and fix all errors of commission, after which (s)he must determine where errors of omission were made and fix them as well. RETRO employs user feedback processing to adjust the candidate link lists as the analyst is making decisions and communicating them to the software. User feedback is used by RETRO to search for more elements like the ones the analyst identified as true links, and then discard the elements like the ones the analyst identified as false positives. In [12], we have seen significant improvement in the number of

errors of commission and some improvement in the number of errors of omission, from candidate link list to candidate link list, when perfect analyst feedback was simulated.

In our case study we have used *vector space retrieval method* using *term frequency-inverse document frequency (tf-idf)*[1] term weighting schema [2] to generate an initial candidate RTM. After this, analyst feedback was simulated for eight iterations. At each iteration, for each functional requirement, two top *previously unvisited* links were checked against the *real* RTM and the "yes-link/no-link" decisions were communicated back to RETRO. These decisions were used to produce a new candidate RTM. *The candidate RTM produced after iteration eight was used to measure the traceability of the project.*

The accuracy of each candidate RTM can be measured in both absolute and relative terms. In absolute terms, we can measure the accuracy in terms of the number of *strikes* (errors of commission) and *misses (*errors of omission) found in the candidate RTM. In relative terms, we can use *precision* and *recall.* Precision is the percentage of the retrieved links that are true. Recall is the percentage of true links that are retrieved. Precision and recall can be combined into a single measure, called *f-measure,* the harmonic mean of the two. If one parameter is valued more than the other, a *skewed f-measure* with parameter $b>0$ is used. If $b < 1$, precision is preferred, if $b>1$, recall is preferred. In our study we used the skewed harmonic mean with $b=2$, which is a standard value for the situation when recall is about twice as important as precision..

Recall and precision of a candidate RTM measure its accuracy and, thus, can be viewed as measures of traceability. In our study, we also used a family of more direct measures, estimating the effort needed to create the final RTM from the one produced by our process. As mentioned above, "fixing" a candidate RTM involves correcting errors of commission and errors of omission. We expect that fixing an error of omission should be a more complicated task than fixing an error of commission. The candidate RTM provides pairs of elements (links) to inspect. Errors of commission are links observed in the candidate RTM which are not correct matches. To fix them, the analyst simply needs to analyze the presented link. However, to fix an error of omission, the analyst must: (a) recognize that such an error is present in the RTM, i.e., detect that the high-level element is not completely satisfied in the candidate RTM, (b) search for potential matches in the low-level document, and (c) make "yes-link/no-link" determination for each potential match detected. In general, we expect that the amount of time spent fixing an error of omission will be longer than the amount of time spent fixing an error of commission.

For this case study, we have modeled the effort to "fix" a candidate RTM in terms of number of error of commission "equivalents" needed to turn the candidate RTM into a perfect RTM. Here, each error of commission is counted once, and each error of omission is counted $k$ times, where $k$ is the ratio of the effort needed to fix errors of each type. In the case study, we have looked at four such ratios, covering a reasonably large span of possibilities: $k = 1,4,8,16$. When $k=1$, we assume that one error of omission "costs" exactly one error of commission, when $k=4$, we assume that one error of omission costs four (4) errors of commission, etc. Thus, if $s$ is the number of *strikes* and $m$ is the number of *misses* in the candidate RTM, and $k$ is the above-mentioned ratio, then the *traceability effort* measure $te\_k$ is defined as follows:

$$te\_k = k*m + s.$$

Thus, for each project, we have collected the *precision, recall, f-measure, f2-measure,* and $te\_k$ measures for $k=1,4,8,16$ in addition to the project grade and other coursework grade information available to us from the dataset. We have run two sets of analyses. When looking for

[1] Vector-space retrieval methods represent artifact elements as vectors of keyword weights. *Tf-idf* method computes the weight of an individual keyword in an element as a product of *term frequency(tf)* – normalized frequency of occurrence of a term in the element and *inverse document frequency (idf)* – the logarithm of the ratio of the total number of elements to the number of elements containing the term in question. Given two vectors, constructed this way, their similarity is computed as the cosine of the angle between them.

correlation between project grades and traceability measures, we have used the 22 projects as our sample. When looking for correlation between other coursework grades and traceability measures, we used the sample of 85 students. Each student had his/her individual grades report, while the project traceability information was common for all students from the same group. Our goal was to see which correlations, if any, are found between these two sets of characteristics.

The *null hypothesis* for the study was that there is a *positive correlation* between observable student performance indicators (grades) and some of the traceability measures considered. The *alternative hypothesis* was that no traceability measure shows positive correlation with any of the grades. Possible experimental threats to validity include: lack of data on individual contribution to group projects, imperfections in the grading process, lack of data on students at other universities, and lack of data on diversity of the students involved in the experiment.

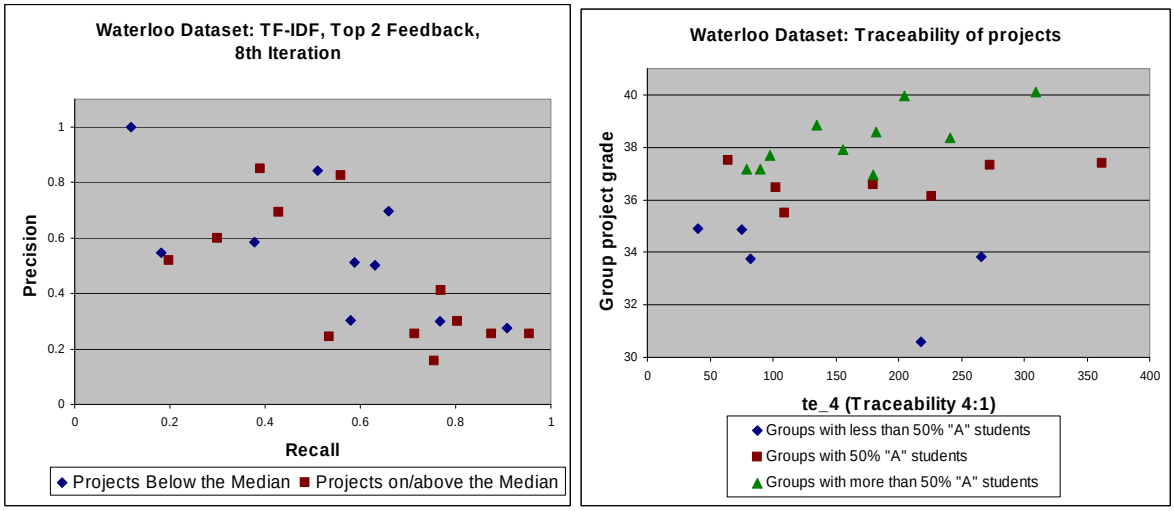


**Figure 2. Precision vs. Recall and Group Project Grade vs. Traceability.**

### 3.4. Case Study Analysis

The graphs shown in Figure 2 are included to give the reader an idea of the Waterloo dataset. The first graph shown plots the precision-recall pairs for each project. The precision and recall were measured after the 8th iteration of RETRO's feedback processing loop. For illustrative purposes, we use different markers to distinguish between the projects in the top half of the class (12 projects, the median project grade of 37.17 is shared by two groups) and the projects in the bottom half of the class (10 projects). The second graph plots the project grade for each project vs. its traceability score *te_4*. We distinguish between three categories of projects: those completed by groups with less than half "A" students, those completed by groups with exactly 50% of "A" students in the group, and those completed by groups where the majority of students received "A" in the class. As can be observed from the graph, project grade should be in high correlation with the percentage of "A" students in the group.

Tables 3 and 4 show the key results of our case study. In Table 3, we show the Pearson correlation coefficients and the significance values for the one-tailed test for our eight selected traceability measures and the project grade (analysis performed on the dataset of 22 group entries). We considered the correlation to be significant at level 0.05. As seen from the table, *only f-measure shows a significant correlation* with the group project grade, however, this correlation is ***negative!*** In Table 4, we show the Pearson correlation and significance values for the relationships between the eight selected traceability measures and the three individual grades earned by students: midterm, final exam and course score (analysis performed on the dataset of

85 student entries). As seen from the table, the only significant correlation observed is, again, the ***negative*** correlation between course grade and *f-measure*.

**Table 3. Traceability Measures vs. Project Grade**

| | ***te_1*** | ***te_4*** | ***te_8*** | ***te_16*** | ***recall*** | ***precision*** | ***f-meas.*** | ***f2-meas.*** |
|---|---|---|---|---|---|---|---|---|
| **Pearson Corr.** | 0.332 | 0.186 | 0.072 | 0.007 | -0.002 | -0.266 | **-0.497** | -0.292 |
| **Significance** | 0.0655 | 0.203 | 0.3755 | 0.4885 | 0.4975 | 0.116 | **0.0095** | 0.0935 |

**Table 4. Traceability Measures vs. Other Course Grades**

| | **Midterm Grade** | | **Final Exam Grade** | | **Course Grade** | |
|---|---|---|---|---|---|---|
| **Measure** | **Corr.** | **Sig.** | **Corr.** | **Sig.** | **Corr.** | **Sig.** |
| ***te_1*** | 0.065 | 0.2785 | 0.041 | 0.3535 | 0.119 | 0.139 |
| ***te_4*** | 0.147 | 0.09 | 0.096 | 0.191 | 0.143 | 0.096 |
| ***te_8*** | 0.154 | 0.0795 | 0.101 | 0.1785 | 0.122 | 0.132 |
| ***te_16*** | 0.149 | 0.086 | 0.098 | 0.1855 | 0.105 | 0.17 |
| ***recall*** | -0.080 | 0.2325 | -0.069 | 0.2635 | -0.063 | 0.282 |
| ***precision*** | 0.064 | 0.279 | 0.038 | 0.3655 | -0.018 | 0.4355 |
| ***f-measure*** | -0.078 | 0.2405 | -0.097 | 0.189 | **-0.192** | **0.039** |
| ***f2-measure*** | -0.124 | 0.1285 | -0.115 | 0.1465 | -0.169 | 0.061 |

### 3.5. Discussion

Our range of traceability measures is quite broad, encompassing standard IR measures, like precision and recall, and their harmonic means, as well as direct measures to assess the traceability effort which capture distinctly different assumptions about the trade-offs between errors of omission and errors of commission. We do not know which of the four *te_k* measures considered is **the** valid one (or closest to **the** valid one), but we believe that the four *te_k* measures considered capture enough possibilities to make at least one of them a realistic approximation of traceability effort. The fact that **none of them** are in statistically significant relationships with any of the grades suggests to us that the *observable course performance indicators did not capture the notion of project artifact traceability* in our case study. The only two statistically significant relationships detected were between the *f-measure* and the project and course grades, but these relationships were both ***negative***, meaning that contrary to our null hypothesis, higher value of *f-measure* tended to lead to lower grades.

The data available to us does not allow us to expand our conclusions beyond the scope of the case study. Due to lack of data we are additionally unable to address individual student effort applied to group projects and other facets of this project that could provide futher insight. However, in our opinion, our study has uncovered an important issue – an apparent mismatch between observable student grades in software engineering coursework and the qualities considered important in the software engineering profession.

## 4. Conclusions and Future Work

We found that none of the current grades in the University of Waterloo software engineering course (which are very typical of the grades in other software engineering courses around the world) embraced or captured the traceability quality. We feel that the ability to develop traceable projects is an important skill necessary to succeed as a software engineer and should be captured via the grading process. A much larger study, using students from various universities working on diverse projects and following those students during their transition into the workforce, should be undertaken before broad conclusions can be reached. Our work raises a number of additional questions to be investigated. First, we want to study *te_k* measures *in-vivo* to determine which values of *k* are typically exhibited by industry analysts in their tracing work. Other questions include “are indirect measures as a group the best way to predict success?” and “does the quality

of a requirement specification best predict the student's future success?" Practical indirect measures specifically and many different types of derived measures should be examined.

## Acknowledgments

Our work is funded by NASA under grant NAG5-11732. We thank Stephanie Ferguson and Ken McGill. We thank Professor Dan Berry, Davor Svetinovic, and the University of Waterloo for the student projects, as well as the University of Waterloo software engineering students. We thank Hakim Sultanov for his help.

## References

[1] Alexander, S., Clark, M., Loose, et.al. 2003. Case studies in admissions to and early performance in computer science degrees. In *Working Group Reports From ITiCSE on innovation and Technology in Computer Science* D. Finkel, Ed. ITiCSE-WGR '03. ACM Press, New York, NY, 137-147.

[2] Baeza-Yates, R., B. Ribeiro-Neto. Modern Information Retrieval, *Addison-Wesley*, 1999.

[3] Brown, T., Reeves, T., and Scott, T. 2003. Assigning and grading computer programs. *J. Comput. Small Coll.* 19, 1 (Oct. 2003), 167-168.

[4] Chmura, G. A. 1998. What abilities are necessary for success in computer science. *SIGCSE Bull.* 30, 4 (Dec. 1998), 55-58.

[5] Clive C.H. Rosen. "Individual Assessment of Group Projects in Software Engineering: A Facilitated Peer Assessment Approach," *Proc. 9th Conference on Software Engineering Education* (*CSEE*), p.68 1996.

[6] Conklin, J.: "Design Rationale and Maintainability"; *Proc. 22nd Hawaii International Conference on System Science*, Los Alamitos: *IEEE Computer Society Press*, Vol. 2 1989), pp. 533-539.

[7] Klappholz, D. Bernstein, L. Port, D. "Assessing Attitude Towards, Knowledge of, and Ability to Apply, Software Development Process," *Proc. 16th Conference on Software Engineering Education and Training* (*CSEE&T 2003*), p. 268 2003.

[8] "Do grades make the grade for program assessment?" *ABET Quarterly News Source*, (Fall/Winter):8--9, 2003.

[9] Dömges, R. and Pohl, K.. "Adapting Traceability to Project-SP." *Comm. of the ACM*, 41(12):54–62, Dec. 1998.

[10] Egyed, A. "A Scenario-Driven Approach to Traceability." *Proc. of the 23rd international Conference on Software Engineering*, 123-132, 2001.

[11] Hayes, J.H., Dekhtyar, A., and Osborne, J. Improving Requirements Tracing via Information Retrieval, in *Proceedings, 11th International Requirements Engineering Conference (RE 2003)*, September 2003, Monterey Bay, CA.

[12] Hayes, J.H., Dekhtyar, A., Sundaram,S.K, and Howard, S. Helping Analysts Trace Requirements: An Objective Look (2004), in *Proceedings, 12th International Requirements Engineering Conference (RE 2004)*, pp. 249-261, September 2004, Kyoto, Japan.

[13] Hayes, J.H., Lethbridge, T and Port, D. "Evaluating Individual Contribution Toward Group Software Engineering Projects," in *Proc. of the International Conference on Software Engineering* (*ICSE*), Portland, Oregon, May 2003, pp. 622-627.

[14] Hazzan, O. 2003. Computer science students' conception of the relationship between reward (grade) and cooperation. In *Proc.of the 8th Annual Conference on innovation and Technology in Computer Science Education* (Thessaloniki, Greece, June 30 - July 02, 2003). D. Finkel, Ed. ITiCSE '03. ACM Press, New York, NY, 178-182.

[15] Howatt, J. W. 1994. On criteria for grading student programs. *SIGCSE Bull.* 26, 3 (Sep. 1994), 3-7.

[16] Hostetler, T. R. 1983. Predicting student success in an introductory programming course. *SIGCSE Bull.* 15, 3 (Sep. 1983), 40-43.

[17] Miller, N. E. and Peterson, C. G. 1980. A method for evaluating student written computer programs in an undergraduate computer science programming language course. *SIGCSE Bull.* 12, 4 (Dec. 1980), 9-17.

[18] Newsted, P. R. 1975. Grade and ability predictions in an introductory programming course. *SIGCSE Bull.* 7, 2 (Jun. 1975), 87-91. DOI= http://doi.acm.org/10.1145/382205.382897

[19] Ramesh, B T. Powers, C. Stubbs, M. Edwards. "Implementing requirements traceability: a case study," re, p. 89, *2nd IEEE International Symposium on Requirements Engineering*, 1995.

[20] Rountree, N., Rountree, J., and Robins, A. 2002. Predictors of success and failure in a CS1 course. *SIGCSE Bull.* 34, 4 (Dec. 2002), 121-124.

[21] U.S. Department of Defense, "Military Standard: Defense Systems Software Development, DOD-DTD-2167A," February 1988.

[22] Walker, H. M. 2000. Notes on grading. *SIGCSE Bull.* 32, 2 (Jun. 2000), 18-19.

[23] Wilson, B. C. and Shrock, S. 2001. Contributing to success in an introductory computer science course: a study of twelve factors. In *Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education* (Charlotte, North Carolina, United States). SIGCSE '01. ACM Press, New York, NY, 184-188.