The Learning and Productivity Benefits to Student Programmers from Real-World Development Environments

Justin C. W. Debuse
jdebuse@usc.edu.au

Meredith Lawley
mlawley1@usc.edu.au

Faculty of Business,
University of the Sunshine Coast
Maroochydore DC, Australia

Abstract

Existing research and practice in software development environments shows no clear consensus on the most appropriate development tools to use; these may range from simple text editors through teaching-oriented examples to full commercial integrated development environments (IDEs). This study addresses this gap by examining student perceptions of two development environments at opposite ends of the complexity spectrum. The results, gathered over several years using students at a range of experience levels, suggest that complex commercial IDEs are appropriate for programming education, even for entry-level students. Indeed, they offer a range of features that may improve the understanding and productivity of students. However, given the greater simplicity of simple text editors and potential for students to become overly dependent upon the support mechanisms provided by IDEs, teaching IDEs in combination with simple text editors appears to offer an ideal combination to maximize learning opportunities and student employability.

Keywords: integrated development environment, IDE, programming, learning, teaching

1. INTRODUCTION

A key challenge for ICT educators is to teach underlying concepts, such as structured analysis and data modelling (Tastle & Russell, 2003), so that students have transferable skills and deep understanding. However, the employment market demands specific skills such as ASP (Colomb, Death, Brown, & Clarkson, 2001) or Java (Liu, Liu, Lu, & Koong, 2003), and a compromise must therefore be found between technology-specific details and fundamental principles. Programming courses must strike this balance not only for the language but also the development environment. For example, the popular Java language can be taught using a range of environments, from a command line interface and text editor through a simple teaching-oriented integrated development environment (IDE) to a complex commercial IDE. The selected environment must fulfil a number of different and potentially conflicting criteria: employment market demand, learning support and ease of use.

The demand by employers appears highest for text editors (Russell, 2005a), although users show preference for using IDEs (Computerworld,

2005; Russell, 2005a). Learning support is high in teaching-oriented systems such as BlueJ, which have been found to assist student understanding of the object-oriented paradigm (Van Haaster & Hagan, 2004) and allow unusual topic orderings to be used (Murray, Heines, Moore, Trono, Kolling, Schaller, & Wagner, 2003). However, commercial IDEs such as JBuilder can also offer considerable teaching and learning support (Liang, 2005).

The ease of use of a development environment is likely to be affected by its complexity. Indeed, the complex nature of commercial IDEs has been used to justify using teaching-oriented alternatives (Kölling, Quig, Patterson, & Rosenberg, 2003), and may explain the high popularity of text editors for education (Russell, 2005a) and low usage of CASE tools for application development at both undergraduate and postgraduate levels (Chinn, Lloyd, & Kyper, 2005). However, there appears to be no clear consensus on whether IDE usability should be criticised (Kline, Seffah, Javahery, Donayee, & Rilling, 2002; Murray et al., 2003; Reis & Cartwright, 2004; Seffah & Rilling, 2001) or praised (Dujmovic & Nagashima; Murray et al., 2003), and students have not shown a preference for specific development environments (Russell, 2005a). Programming textbooks show similar dissent; examples exist that use commercial IDEs such as JBuilder (Liang, 2004), teaching-oriented IDEs (Barnes & Kolling, 2008), text editors (Farrell, 2003), or allow educators to choose between a text editor and IDE (Liang, 2009).

The impact of development environments upon student learning and understanding from a formative perspective is one of the least studied areas of IDE research (Gross & Powers, 2005). Existing studies, particularly those which measure student performance directly (Kordaki, 2010; Vogts, Calitz, & Greyling, 2008), have examined the educational suitability of IDEs to only a limited level of granularity; for example, Kordaki (2010) examines different development environments across broad areas such as the quality of students' code, rather than the features of the environments in detail. Further, although educational IDEs appear to yield improvements in student understanding (Rigby & Thompson, 2005; Van Haaster & Hagan, 2004; Xinogalos, Satratzemi, & Dagdilelis, 2006) and programming performance (Kordaki, 2010; Vogts et al., 2008), they require room in the syllabus to be found for students to convert to real world environments, which is unlikely to prove easy (Xinogalos et al., 2006).

This study therefore attempts to extend existing work to a finer level of granularity, in order to clarify the selection of development environments for programming education by determining whether the learning support and ease of use of an environment for which significant employment market demand exists are sufficiently strong for it to be successfully used without going through the intermediate step of using a teaching-oriented IDE. The environment used is Borland's JBuilder/Together, one of the object-oriented analysis and design market leaders (Blechar, 2004) and now incorporated into the popular Eclipse environment. The students examined are from a single regional university and thus likely to have greater requirements for learning support and ease of use than their metropolitan equivalents. Moreover, the students are examined at three stages in their programming education to determine the performance of the environment across a range of experience levels.

2. METHOD

The commercial IDE examined within this study was JBuilder from Borland; this was supplied within the teaching laboratories and used to deliver lectures and tutorials. Programming students were studied from 2005 to 2008; during 2007 JBuilder was incorporated into the Eclipse system, which offered very similar functionality. Students were surveyed across the three groups described below, to allow differences between programmers across a range of experience levels to be investigated.

Group 1: Introductory Java Programming

The first group of students took an introductory course in Java programming, held during semester 2 each year from 2005 to 2008. Students' attitudes to the IDE were surveyed using an instrument adapted from (Hede, 2005); the 2008 version is presented within Appendix 2. The first section established their prior knowledge of programming and IDEs, using questions adapted from (Russell, 2005b). The section included items determining whether JBuilder and Java were the most commonly used development environment and language, to confirm that students met the requirements of the study.

The second section investigated the complexity of JBuilder, since this drawback of commercial IDEs has been used to justify using teaching-oriented IDEs for education (Kölling et al., 2003). The statements in Table 1, labelled JBA, were used to assess how this affected students, both when they began learning to use JBuilder and once they had become proficient in using it; the available responses ranged from five (strongly agree) through three (neutral) to one (strongly disagree). Statement JBA1 is similar to the learnability scale from the Software Usability Measurement Inventory (SUMI) (Kirakowski & Corbett, 1993), described by (Kline et al., 2002). However, Kline et al. (2002) also cite a minimum sample size of 50 subjects to be confident of the results (Nunnally & Bernstein, 1994). This exceeds the current student numbers for the courses examined within this paper, and so SUMI scales were not used. Statements JBA3 to JBA6 were instead added to the questionnaire to cover the missing SUMI scales of affect, helpfulness, efficiency and control respectively.

The instrument also included questions, labelled JBB, to measure how aspects of the IDE improved or impaired understanding of the course concepts and productivity in producing its required deliverables. Five point Likert scales were again used, with one set (labelled A) determining the effect on understanding and a second (labelled B) measuring the effect on productivity; however, unlike the previous scales, the values ranged from 5 (strong improvement) through 3 (no effect) to 1 (strong impairment), together with 0 (if they have never used the feature or respond that this is not applicable). The IDE aspects investigated are shown in Table 2; some were adapted from (Dujmovic & Nagashima; Russell, 2005a; Storey, Michaud, Mindel, Sanseverino, Damian, Myers, German, & Hargreaves, 2003), and features absent from the university Java programming courses were excluded. The instrument has some overlap with Russell's (2005b) survey, although it examines the IDE aspects at a greater level of detail.

Preliminary results from this study suggested that students may become over-reliant upon the support mechanisms offered by JBuilder. The course examined was therefore revised after its 2005 intake to use a text editor (Programmers Notepad) initially, followed by JBuilder, rather than using JBuilder throughout. The second survey and its successors thus contained additional questions: PNA, which applied the complexity statements in Table 1 to the text editor rather than JBuilder; PNB, which investigated similar IDE aspects to those listed within Table 2, but aimed at the text editor rather than JBuilder (with a corresponding reduction in the number of aspects due to the more limited functionality of the text editor); and JBPN, adapted from (Russell, 2005b), which determines which environment students would have preferred to use to learn programming, together with which environment they would rather currently program with (the JBPN questions were administered in a separate survey during 2006 but incorporated into the main survey from 2007 onwards).

Group 2: Intermediate Java Programming

The second group of students took an intermediate level follow-on from the introductory Java programming course taken by group 1, held during semester 1 from 2006 to 2008. The course taken by group 1 was a prerequisite for the course taken by this group; a number of students from group 1 would therefore subsequently join group 2. For example, 68% of the students who took the intermediate course during 2006 had previously taken the introductory course during 2005. The group 1 instrument was applied for the group 2 students with minor modifications corresponding to their differing course enrolments.

Group 3: Architecture & Systems Integration

The third group of students took a capstone architecture and systems integration course in semester 2 2005, where programming skills were applied to systems integration tasks, using JavaScript and the Notepad text editor. The survey was only administered in 2005, and used an adaptation of the group 1 instrument which was modified to reflect different course enrolments and the use of Notepad in place of Programmers Notepad; further, section PNB (IDE aspects) was omitted due to the limited functionality of Notepad.

Analysis of Results

Missing values were identified as such when the data was entered and excluded from calculations on a pairwise basis; this means that the response for a student was only excluded from a calculation if data required by that calculation

was missing. Responses of zero for the IDE aspect statements were also treated as missing, as this value represented that the statement was not applicable or that the respondent had never used the feature. If a respondent indicated that they had not used a development environment but then proceeded to respond to items regarding the environment then these responses were excluded and treated as missing. Similarly, for each aspect statement there are two questions, covering understanding and productivity; if a response to either of these questions indicated that the aspect was never used or was not applicable then both were treated as missing data.

Hypothesis Testing

The results of the survey within this study were used for hypothesis testing, using a similar approach to that described in (Debuse, Lawley, & Shibl, 2007, 2008; Stevens & Jamieson, 2002). For the complexity assessment statements (labelled JBA and PNA), two hypotheses were formed; the first was that respondents agreed with the statement and the second was that respondents disagreed. Such an approach was used in place of a single hypothesis since responses could indicate agreement, neutrality or disagreement; thus, a hypothesis based on agreement may fail to hold, but this does not necessarily indicate disagreement. Specifically, for the first hypothesis to hold, the response must be greater than three; this equates to a response above 'Neutral', which may be high enough to equate to 'Tend to Agree' or 'Strongly Agree'. For the second hypothesis to hold, the response must be three or less; this equates to a response of 'Neutral', 'Tend to Disagree' or 'Strongly Disagree'. The 95% confidence interval for the mean response value was computed, and its lower and upper bounds were used to test the first and second hypotheses respectively. For example, consider lower and upper bounds for the 95% confidence interval for the mean response to statement JBA1 of 3.4 and 5.1 respectively. Such values would cause the first hypothesis for JBA1 to be accepted, since 3.4 is greater than three, and the second to be rejected, since 5.1 is greater than three. Such a result would lead to the conclusion that respondents agreed with JBA1.

Similar hypotheses were formed and tested for the aspect statements (labelled JBB and PNB). For each aspect, two hypotheses were again formed; the first was that it had improved respondents' understanding of the course concepts and the second was that it had impaired them. A third and fourth hypothesis were similarly formed for each aspect; these concerned its improvement or impairment respectively to respondents' productivity. The lower end of the 95% confidence interval of the mean response to the understanding scale had to exceed three for the first hypothesis to hold; for the second, the upper end had to be three or less. Similarly, lower and upper ends of the 95% confidence interval of the mean response to the productivity scale were calculated. If the lower exceeded three then the first hypothesis held; an upper value of three or less caused the second hypothesis to hold.

3. RESULTS

The total responses received across all surveys totalled 167; the breakdown of these, together with key demographic information, hypotheses and preferred development environments are presented in the following sections.

Demographics

Table 3 shows that all groups represent junior programmers, with at most one to three years' experience. Group 1 are the most junior, with responses being mainly less than one year rather than the one to three years for groups 2 and 3. All groups apart from 3 report JBuilder as the IDE used; the majority of groups used JBuilder the most. The JBuilder environment is thus very familiar to the students, and all groups had the most programming experience in Java.

The demographics thus suggest that the students examined meet the requirements of this study, namely junior programmers at differing points in their programming education, with experience in Java and JBuilder.

Hypotheses

The left half of Table 4 shows the hypotheses that held for each group; hypothesis 1 (H1) holding is denoted by 1, and hypothesis 2 (H2) holding is denoted by 2. An empty cell shows that neither hypothesis held; nor does NA show that the hypothesis was tested for the specified year.

The right half of Table 4 summarises the total number of times that hypotheses 1 and 2 held

across all groups, along with the percentage of non-NA groups for which hypotheses 1 and 2 held; these summaries are also given for group 1 across all years together with group 2 across all years. Grey denotes rows for which hypothesis 1 holds for every group examined, and bold denotes rows for which hypothesis 1 holds for no group examined.

The Notepad results for group 3 are not included in the table as they were only recorded for a single group. For these PNA statements, hypothesis 1 held for statements PNA1 and PNA2; hypothesis 2 held for PNA4.

Table 4 suggests that, for some JBuilder aspects (denoted by grey rows, starting at aspect JBB1A), every group examined found them to yield understanding and/or productivity benefits. Of these aspects, the following yielded both understanding and productivity benefits:

- Automatic bracket/brace matching
- Automatic code formatting
- Automatic completion of words in programs
- Display of parameter lists
- Automatic code colouring
- Automatic syntax error reporting
- Code audit warnings
- Breakpoint / line by line execution in debugging
- Variable value viewing / modification in debugging

The remainder of the aspects for which every group examined reported benefits yielded productivity but not understanding improvements:

- Automatic creation of program code
- Automatic generation of Javadoc comments
- Display of line numbers

There was no universal agreement on any other area of JBuilder or Programmers Notepad, but one area of Programmers Notepad (PNB4A – the benefit of case conversion within Programmers Notepad to understanding) was not perceived to give understanding/productivity benefits within any group. Further, respondents disagreed with one of the Programmers Notepad utility / ease of use statements (PNA4 – Programmers Notepad gives me assistance in its use) in one group, although they agreed with this within another.

The results can also be analysed in terms of totals over all group 1 students compared to totals over all group 2 students. In addition to the grey cells noted above (which will have 100% hypothesis 1 coverage for both these groups), these groups have 100% hypothesis 1 agreement for the following statements relating to JBuilder:

- Automatic program code creation improves understanding (group 1 and group 2)
- Code creation wizards improve productivity (group 1 only)
- Sync edit tool, which allows all instances of a variable name to be changed by editing a single instance of the name, improves understanding and productivity (group 1 only)
- Line number display improves understanding (group 1 and group 2)
- Automatic Javadoc creation improves understanding (group 1 and group 2)
- Javadoc integration improves understanding and productivity (group 2 only)
- The automatic link between Java and UML improves understanding (group 1 only)

Further, group 1 has 100% agreement with hypothesis 1 for the following statements relating to Programmers Notepad:

- Learning to use Programmers Notepad is straightforward
- Programmers Notepad automatic code colouring improves productivity
- Programmers Notepad display of line numbers improves understanding and productivity

Items for which no agreement was shown over the group 1 and/or group 2 groups, in addition to PNB4A described above, are:

- I feel I am in control of JBuilder when I use it (group 2 only)
- The automatic Java/UML link improves understanding and productivity (group 2 only)
- Once you have learned to use Programmers Notepad then producing Java software with it is straightforward (group 2 only)
- Using Programmers Notepad is enjoyable (group 2 only)

- Programmers Notepad gives me assistance in its use (group 2 only)
- The amount of time and effort required to perform tasks in PN is low (group 2 only)
- Programmers Notepad bookmarks improve understanding and productivity (group 2 only)
- Programmers Notepad display of line numbers improves productivity (group 2 only)

Summarising the differences between groups 1 and 2, it appears that the two have similar views regarding JBuilder, with differences in terms of 100% agreement only occurring for a small number of items. Differences are more extreme for Programmers Notepad, with only group 1 having some items which were agreed with across all years, and only group 2 having some items for which agreement was not found for any year. The group 1 students therefore appear much more positively disposed towards Programmers Notepad than group 2.

The most experienced programmers (group 3) were more positively disposed towards text editors than group 2, finding Notepad easy to use and produce Java software with, although unsurprisingly they did not find it supportive. However, they also found JBuilder to be easy to produce software with, and found it enjoyable and supportive to use. Further, the majority of the features of JBuilder proved to be useful to both their understanding and productivity.

Development Environment Preferences

Table 5 shows that there is no consensus across all groups in terms of the preferred environment to learn programming, although all but one prefer a combination of JBuilder and Programmers Notepad. All groups preferred to use JBuilder to do programming now (one of these was multi-modal).

4. DISCUSSION

The results suggest that students perceive considerable benefits from a real-world integrated development environment (IDE) such as JBuilder, which represents their preferred option for programming; however, a combination of text editor and IDE appear to be preferable for learning purposes. Students' responses overall are very positive for almost all areas examined within this study; the only negative responses were for case conversion and support within the text editor. All three groups of students appeared not to believe that any of the surveyed JBuilder IDE aspects impaired their understanding of course concepts or productivity. Indeed, the majority of the JBuilder aspects examined were found to improve productivity and/or understanding by all groups, and every item was present in at least one group.

A text editor appears particularly appealing to the group 1 students, particularly in terms of its reduced complexity; the more experienced group 2 students appear to be less positively inclined towards it, although the most experienced group (3) appeared to view such systems more favourably. However, the groups have similar views regarding JBuilder, and a number of its features in areas such as debugging and simple code writing support appear to yield understanding and productivity benefits across all groups and years. Further, features such as more sophisticated code writing support appear to have universal benefit, but only in terms of productivity; this is unsurprising given the potential for such support to deny students the opportunity to learn how to create code.

The most experienced programmers (group 3) were more positively disposed towards text editors than group 2, finding Notepad easy to use and produce Java software with, although unsurprisingly they did not find it supportive. However, they also found JBuilder to be easy to produce software with, and found it enjoyable and supportive to use.

The preference for simplicity by entry level students is unsurprising given the documented unsuitability of professional IDEs for teaching given their complexity (Reis & Cartwright, 2004). Complicated aspects of the Java language may also prove distracting (Reis & Cartwright, 2004); this may explain why many students in this find automated code creation to improve their understanding, since at a conceptual level the language complexities may impair learning.

The results contain a number of points of interest. Firstly, despite the study being held at a regional university, at which student quality is unlikely to be higher than at metropolitan centres, the respondents did not find the JBuilder IDE complex; indeed, both the novice

and experienced programmers found it straightforward to produce software with, and the novices found JBuilder easy to learn. This may be partially explained by the approach used to teach the programming courses at the university, with JBuilder being covered extensively throughout lectures, tutorials and the course text.

The overall status of JBuilder as the preferred environment for students to use currently matches existing research (Russell, 2005a). The results also overlap with those testing a visual programming language, where most students perceived improvements in their understanding and found the environment helpful (Collins & Fung, 2002). The results of Kline et al. (2002), who found that experienced programmers viewed their IDE as helpful and efficient, also support this study. However, unlike this study their programmers did not find the IDE easy to learn. Further, the preferred option identified within this study of combining a text editor and IDE for learning is unsurprising given the lack of consensus in existing research on whether text editors or IDEs would be preferred for training purposes (Russell, 2005a).

It is surprising that the majority of the IDE aspects examined were found to improve understanding and/or productivity, with debugging support being particularly useful; this contradicts existing research suggesting that integrated debugging is the least useful feature for both learning and production programming (Russell, 2005a). Other popular features such as automated code completion and Javadoc integration also proved unpopular (Russell, 2005a), although the popularity of areas such as bracket matching and syntax highlighting is supported by existing research (Russell, 2005a). Further, these results are supported by studies indicating that over 85% of user requirements are satisfied by current IDEs, with JBuilder offering the best performance (Dujmovic & Nagashima), and that the JBuilder debugging support is useful for teaching (Liang, 2005; Murray et al., 2003). Similarly, the BlueJ development environment has also improved students' understanding of object-oriented concepts (Van Haaster & Hagan, 2004); correspondingly, the educational IDE objectKarel yielded improvements in students' perceptions of their understanding (Xinogalos et al., 2006), and students using the LECGO for C educational IDE programmed more successfully than using a non-teaching environment or pencil and paper (Kordaki, 2010). Students' performance using the SimplifIDE educational plug IDE improved the programming performance of students compared to a professional IDE; their understanding, measured by assessment grades, was only superior using the educational IDE for weaker students (Vogts et al., 2008). The Gild educational plug in for Eclipse, when compared to Eclipse, appears to improve students' perceptions of their understanding but not their programming performance or productivity (Rigby & Thompson, 2005). Improvements in students' perceptions of their understanding have also been attributed to the Eclipse IDE (Hanks, 2006).

A study examining actual usage data for the Eclipse IDE across 41 Java software developers using the Mylar Monitor plug-in (Murphy, Kersten, & Findlater, 2006) gave strong support for the automatic program word completion that was found to be so important to understanding and productivity; the developers used such completion as often as popular editing commands such as copy and paste. The importance of debugging identified within this study was also supported (Murphy et al., 2006). Further, the sync edit tool, which was particularly popular with entry-level programmers within this study and allows all instances of a variable name to be changed by editing a single instance of the name, was part of the most popular refactoring command (rename), which was used by all respondents (Murphy et al., 2006).

The study has a number of limitations. Firstly, although students are surveyed at three different points in their education, the longer term effects of the IDE are not examined. Secondly, the most experienced group of students do not use JBuilder within their course; however, over half of them have used JBuilder, although many of these will be relying on memories of past courses. Thirdly, the study is restricted to a single organisation and single example of each tool. This approach, though used in a number of existing studies (Collins & Fung, 2002; Kordaki, 2010; Xinogalos et al., 2006), restricts the extent to which the results can be generalised, since specific details such as courses, tools and student demographics may contribute to the results, particularly as IDEs are presented very positively to students within the programming courses of this study; this does however offer the advantage of limiting potential confounding effects from areas such as

instructor or syllabus variations. Fourthly, any development environment that is successfully used by students will have a positive effect on their understanding, and the sequential usage of the two environments examined means that they will impact upon students at different learning stages. Finally, the study examines only students' perceptions rather than actual usage data. Although student perceptions of software usability and its effects on their own productivity would be unlikely to give inaccurate responses, understanding has the potential to be more problematic. This is because students' perceptions of their own understanding may not correlate well with their actual levels; also, the surveys query students' understanding of course concepts without giving details of the specific learning outcomes and course objectives to which such concepts relate, which gives the potential for weaker students to not realise that they have missed certain concepts; the objectives will also not be the same across all of the courses. However, the overall approach is not unusual, with a number of existing studies measuring student perceptions of understanding (Collins & Fung, 2002; Hanks, 2006; Rigby & Thompson, 2005; Xinogalos et al., 2006). Further, although weaker students have demonstrated a tendency to overrate themselves compared to educators, no consistent over or underrating has been found (Boud & Falchikov, 1989); indeed, a weak positive correlation has been found between student self assessment and educator assessment (Falchikov & Boud, 1989), and a review of existing work suggests that in the majority of studies the number of cases where student and staff marks agreed outnumbered those where they disagreed (Boud & Falchikov, 1989). Moreover, the number of development environment features for which understanding is examined is too large to feasibly investigate directly.

5. CONCLUSIONS

This study has highlighted a number of areas of importance for software development education. It appears that university students can learn introductory programming using a complex commercial IDE, without requiring the intermediate step of using an educational environment. Moreover, most of the IDE aspects improve their understanding and/or productivity. However, some of these mechanisms can deny students the opportunity to learn key programming skills that environments with limited support require. This suggests that the use of a text editor in addition to a complex IDE would be an ideal combination to maximize learning and future employment opportunities. However, institutional constraints such as the availability of IT service department support clearly need to be taken into account if such approaches are to be adopted.

Future research may determine how students' perceptions of the utility of IDE features correlate with their actual usage data, and where the perceived benefits translate into actual performance enhancements.

6. REFERENCES

Barnes, D. J., & Kolling, M. (2008). Objects First with Java (4th ed.). Prentice Hall / Pearson.

Blechar, M. (2004). Market Details for OOA&D Tools, Update for 2005. *Gartner Research*.

Boud, D., & Falchikov, N. (1989). Quantitative studies of student self-assessment in higher education: a critical analysis of findings. *Higher education, 18*(5), 529-549.

Chinn, S. J., Lloyd, S. J., & Kyper, E. (2005). Contemporary Usage of CASE Tools in U. S. Colleges and Universities. *Journal of Information Systems Education, 16*(4), 429-436.

Collins, T. D., & Fung, P. (2002). A visual programming approach for teaching cognitive modelling. *Computers & Education, 39*(1), 1-18.

Colomb, R., Death, B., Brown, A., & Clarkson, A. (2001). Trends in Computing Jobs - 2001. Retrieved 13 May, 2003, from www.itee.uq.edu.au/~colomb/Jobs-Anal-2001.html

Computerworld (2005). Computerworld Development Survey gives nod to C#. Retrieved 30 August, 2005, from http://www.computerworld.com/developmenttopics/development/story/0,10801,100542,00.html

Debuse, J., Lawley, M., & Shibl, R. (2007). The Implementation of an Automated Assessment Feedback and Quality Assurance

System for ICT Courses. *Journal of Information Systems Education, 18*(4), 491-502.

Debuse, J., Lawley, M., & Shibl, R. (2008). Educators' perceptions of automated feedback systems. *Australasian Journal of Educational Technology, 24*(4), 374-386.

Dujmovic, J., & Nagashima, H. Evaluation of IDE's for Java Enterprise Applications. Retrieved 15 August, 2005, from http://www.seas.com/downloadUNReg/IDE6p.pdf

Falchikov, N., & Boud, D. (1989). Student self-assessment in higher education: A meta-analysis. *Review of Educational Research, 59*(4), 395-430.

Farrell, J. (2003). Java Programming (2nd ed.). Course Technology, Boston, Massachusetts.

Gross, P., & Powers, K. (2005). *Evaluating assessments of novice programming environments.* Paper presented at the First international workshop on Computing education research.

Hanks, B. (2006). Using Eclipse in the classroom. *Journal of Computing Sciences in Colleges, 21*(3), 118-127.

Hede, A. (2005). Personal Communication.

Kirakowski, J., & Corbett, M. (1993). SUMI: The Software Measurement Inventory. *British Journal of Educational Technology, 24*(5), 210-212.

Kline, R., Seffah, A., Javahery, H., Donayee, M., & Rilling, J. (2002, September 3-6). *Quantifying Developer Experiences via Heuristic and Psychometric Evaluation.* Paper presented at the IEEE Symposia on Human Centric Computing Languages and Environments, Arlington, VA.

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, 13*(4).

Kordaki, M. (2010). A drawing and multi-representational computer environment for beginners' learning of programming using C: Design and pilot formative evaluation. *Computers & Education, 54*(1), 69-87.

Liang, Y. (2004). Introduction to Java Programming with JBuilder (3rd ed.). Prentice Hall.

Liang, Y. (2005). Learning Java Effectively with JBuilder. In *Introduction to Java Programming* (7th ed.).

Liang, Y. (2009). Introduction to Java Programming, Comprehensive (7th ed.). Prentice Hall.

Liu, X., Liu, L., Lu, J., & Koong, K. (2003). An Examination of Job Skills Posted on Internet Databases: Implications for Information Systems Degree Programs. *Journal of Education for Business, 78*(4), 191-196.

Murphy, G. C., Kersten, M., & Findlater, L. (2006). How Are Java Software Developers Using the Eclipse IDE? *IEEE Software, 23*(4), 76-83.

Murray, K., Heines, J., Moore, T., Trono, J., Kolling, M., Schaller, N., & Wagner, P. (2003). *Panel on Experiences with IDEs and Java Teaching: What Works and What Doesn't.* Paper presented at the ACM SIG CSE 8th International Conference on Innovation and Technology in Computer Science Education, Thessaloniki, Greece.

Nunnally, J., & Bernstein, I. (1994). Psychometric Theory (3rd ed.). McGraw-Hill, New York.

Reis, C., & Cartwright, R. (2004). *Taming a professional IDE for the classroom.* Paper presented at the SIGCSE technical symposium on Computer Science Education.

Rigby, P. C., & Thompson, S. (2005). *Study of novice programmers using Eclipse and Gild.* Paper presented at the OOPSLA workshop on Eclipse technology eXchange.

Russell, J. (2005a). *Do the benefits of learning Java using an IDE outweigh the in-depth understanding gained by learning with a text editor only?* MSc Thesis, University of Liverpool.

Russell, J. (2005b). MSc Student Survey: Do the benefits of using an IDE to learn Java outweigh the understanding gained by learning with a text editor only. Retrieved 30 August (from Google cache dated 27 February), 2005, from http://www.jeremyrussell.co.uk/studentsurveyquestion.jsp

Seffah, A., & Rilling, J. (2001). *Investigating the Relationship between Usability and Conceptual Gaps for Human-Centric CASE Tools.* Paper presented at the IEEE Symposium on Human-Centric Computing Languages and Environments, Stresa, Italy.

Stevens, K., & Jamieson, R. (2002). The Introduction and Assessment of Three Teaching Tools (WebCT, MindTrail, EVE) into a Post Graduate Course. *Journal of Information Technology Education, 1*(4), 233-252.

Storey, M., Michaud, J., Mindel, M., Sanseverino, M., Damian, D., Myers, D., German, D., & Hargreaves, E. (2003). *Improving the Usability of Eclipse for Novice Programmers.* Paper presented at the Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), Anaheim, California, USA.

Tastle, W., & Russell, J. (2003). Analysis and Design: Assessing Actual and Desired Course Content. *Journal of Information Systems Education, 14*(1), 77-90.

Van Haaster, K., & Hagan, D. (2004, June). *Teaching and Learning with BlueJ: an Evaluation of a Pedagogical Tool.* Paper presented at the Information Science + Information Technology Education Joint Conference, Rockhampton, QLD, Australia.

Vogts, D., Calitz, A., & Greyling, J. (2008). *Comparison of the effects of professional and pedagogical program development environments on novice programmers.* Paper presented at the Annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries.

Xinogalos, S., Satratzemi, M., & Dagdilelis, V. (2006). An introduction to object-oriented programming with a didactic microworld: objectKarel. *Computers & Education, 47*(2), 148-171.

Appendix 1: Tables

Table 1. Complexity assessment statements

Statement	Description
JBA1	Learning how to use JBuilder is straightforward.
JBA2	Once you have learned how to use JBuilder then producing Java software with it is straightforward.
JBA3	Using JBuilder is enjoyable.
JBA4	JBuilder gives me assistance in its use.
JBA5	The amount of time and effort required to perform tasks using JBuilder is low.
JBA6	I feel I am in control of JBuilder when I use it.

Table 2. IDE Aspects

Aspect	Description
JBB1	Automatic code formatting.
JBB2	Automatic completion of words within programs.
JBB3	Parameter list display
JBB4	Automatic creation of code such as missing curly brackets.
JBB5	Code creation wizards for tasks such as class creation.
JBB6	An editing mode that allows all instances of a variable name to be changed by editing a single instance of the name.
JBB7	Integrated help system.
JBB8	Automatic code colouring.
JBB9	Line numbering.
JBB10	Automatic syntax error reporting.
JBB11	Code audit warnings.
JBB12	Deprecation warnings
JBB13	Debugging support through breakpoints and line-by-line execution.
JBB14	Debugging support through viewing and modifying variable values.
JBB15	Automatic bracket matching.
JBB16	Automatic generation of Javadoc comments.
JBB17	Javadoc integration through automatic creation and view of HTML associated with Javadoc comments
JBB18	Automatic two-way links between UML diagrams and their associated program code.

Table 3. Mode responses to demographics (percentage giving mode response in brackets)

Year	2005		2006			2007		2008	
Group	1	3	2	1	1[a]	2	1	2	1
N	10	16	10	31	24	19	22	8	27
How much programming experience do you currently have (years)?	<1 (50%)	1-3 (69%)	1-3 (60%)	1-3 (35.5%)	<1 (33.3%)	1-3 (52.6)	<1 (50%)	<1 (37.5%), 1-3 (37.5%) ie bimodal	<1 (48.1%)
Which Integrated Development Environments (IDEs) have you used?	JBuilder (90%)	A text editor (75%)	JBuilder (100%)	JBuilder (93.5%)	JBuilder (100%)	JBuilder (94.7%)	JBuilder (90.9%)	JBuilder (87.5%)	JBuilder (96.3%)
Which Integrated Development Environment (IDE) do you use the most?	JBuilder (80%)	JBuilder (50%)	JBuilder (100%)	JBuilder (77.4%)	JBuilder (87.5%)	JBuilder (94.7%)	JBuilder (63.6%)	JBuilder (37.5%), Eclipse (37.5%) ie bimodal	Eclipse (51.9%)
Which programming language do you have the most experience in?	Java (70%)	Java (75%)	Java (100%)	Java (80.6%)	Java (75%)	Java (78.9%)	Java (72.7%)	Java (75%)	Java (77.8%)

[a]This survey of the preferred environment was run separately to the rest of the survey during 2006

Table 4. Results of hypotheses (grey denotes rows for which hypothesis 1 holds for every group examined and bold denotes rows for which hypothesis 1 holds for no group examined)

Hypotheses holding for each aspect (1 & 2 denote hypothesis number; empty cells and NA denote no hypothesis holding and no testing respectively)										Summary data for hypotheses holding for each aspect						
Year	2005		2006			2007		2008								
Group	1	3	2	1	1[a]	2	1	2	1	H1total	H2 total	H1%[b]	H1 total (1)[c]	H1 total (2)[c]	H1% (1)[c]	H1% (2)[c]
JBA1	1			1	NA	1			1	4	0	50	3	1	75	33.33
JBA2	1	1		1	NA	1			1	5	0	62.5	3	1	75	33.33
JBA3		1		1	NA	1			1	4	0	50	2	1	50	33.33
JBA4		1		1	NA	1	1	1	1	6	0	75	3	2	75	66.67
JBA5					NA	1			1	2	0	25	1	1	25	33.33
JBA6		1			NA				1	2	0	25	1	0	25	0
JBB1A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB1B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB2A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB2B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB3A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB3B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB4A	1		1	1	NA	1	1	1	1	7	0	87.5	4	3	100	100
JBB4B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB5A			1	1	NA	1	1		1	5	0	62.5	3	2	75	66.67
JBB5B	1	1	1	1	NA	1	1		1	7	0	87.5	4	2	100	66.67
JBB6A	1	1	1	1	NA	1	1		1	7	0	87.5	4	2	100	66.67
JBB6B	1	1	1	1	NA	1	1		1	7	0	87.5	4	2	100	66.67
JBB7A		1		1	NA	1	1		1	5	0	62.5	3	1	75	33.33
JBB7B		1		1	NA	1	1		1	5	0	62.5	3	1	75	33.33
JBB8A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB8B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB9A	1		1	1	NA	1	1	1	1	7	0	87.5	4	3	100	100
JBB9B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB10A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB10B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB11A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB11B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB12A		1		1	NA	1	1	1	1	6	0	75	3	2	75	66.67
JBB12B		1		1	NA	1	1	1	1	6	0	75	3	2	75	66.67
JBB13A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB13B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB14A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB14B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB15A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB15B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB16A	1		1	1	NA	1	1	1	1	7	0	87.5	4	3	100	100
JBB16B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB17A			1	1	NA	1	1	1	1	6	0	75	3	3	75	100
JBB17B		1	1	1	NA	1		1	1	6	0	75	2	3	50	100
JBB18A	1			1	NA		NA		NA	2	0	33.33	2	0	100	0
JBB18B		1		1	NA		NA		NA	2	0	33.33	1	0	50	0
PNA1	NA	NA	NA	1	NA	1	1		1	4	0	80	3	1	100	50
PNA2	NA	NA	NA	1	NA				1	2	0	40	2	0	66.67	0
PNA3	NA	NA	NA		NA				1	1	0	20	1	0	33.33	0
PNA4	NA	NA	NA		NA	2			1	1	1	20	1	0	33.33	0
PNA5	NA	NA	NA		NA				1	1	0	20	1	0	33.33	0
PNA6	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB1A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB1B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB2A	NA	NA	NA	1	NA					1	0	20	1	0	33.33	0
PNB2B	NA	NA	NA	1	NA				1	2	0	40	2	0	66.67	0
PNB3A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB3B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB4A	**NA**	**NA**	**NA**		**NA**					**0**	**0**	**0**	**0**	**0**	**0**	**0**
PNB4B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB5A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB5B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB6A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB6B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB7A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB7B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB8A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB8B	NA	NA	NA	1	NA	1	1		1	4	0	80	3	1	100	50
PNB9A	NA	NA	NA	1	NA		1		1	3	0	60	3	0	100	0
PNB9B	NA	NA	NA	1	NA	1	1		1	4	0	80	3	1	100	50

[a] This survey of the preferred environment was run separately to the rest of the survey during 2006. [b] The percentage of non-NA groups for which hypothesis 1 holds. [c] Groups 1 and 2 are denoted (1) and (2) respectively.

Table 5. Mode responses to system preference questions (percentage giving mode response in brackets)

Year	2006	2007		2008	
Group	**1**	**2**	**1**	**2**	**1**
If you had free choice, which development environment would you prefer to have used to learn programming?	Both JBuilder and Programmers Notepad (45.8%)	JBuilder only (42.1%)	Both JBuilder and Programmers Notepad (18.2%)	Both JBuilder and Programmers Notepad (25%)	Both JBuilder and Programmers Notepad (29.6%)
If you had free choice, which development environment would you prefer to use to do programming now?	JBuilder only (66.7%)	JBuilder only (52.6%)	JBuilder only (22.7%)	JBuilder only (12.5%), Both JBuilder and Programmers Notepad (12.5%), Textmate (12.5%)	JBuilder only (37%)

Appendix 2: Group 1 Survey Instrument (2008)

Note: the labeling used in this survey has been modified within the paper to improve readability; for example, B1 corresponds to JBA1 within the paper

SURVEY ON INTEGRATED DEVELOPMENT ENVIRONMENTS

A - ICT 221/521 STUDENT INFORMATION

A1 Which degree programme are you enrolled in? *(Please tick one box only)*

- BICT ❑1
- BBus (Information Systems) ❑2
- Grad Dip (IS) ❑3
- Other (please specify below) ❑4

.....

A2 Are you enrolled in ICT321/621 Architecture & Systems Integration? *(Please tick one box only)*

- Yes, I am enrolled in ICT321/621 this semester ❑1
- No, but I have already taken ICT321/621 ❑2
- No, but I have been given credit for ICT321/621 through prior learning ❑3
- No, but I will take ICT321/621 in the future ❑4
- No, and I will not take ICT321/621 in the future ❑5

A3 Have you taken INF311/611 Advanced Business Programming? *(Please tick one box only)*

- Yes ❑1
- No, but I have already taken ICT311/611 ❑2
- No, and I have not taken ICT311/611 ❑3

A4 How much programming experience do you currently have? *(Please tick one box only)*

- None ❑0
- Less than one year ❑1
- Between one and three years ❑2
- Between four and six years ❑4
- Between seven and ten years ❑5
- More than ten years ❑6

A5 Which Integrated Development Environments (IDEs) have you used? *(Please tick all boxes that apply)*

AnyJ	❑1	NetBeans	❑9
BlueJ	❑2	OptimalJ	❑10
JBuilder	❑3	Oracle JDeveloper	❑11
DrJava	❑4	Programmers Notepad	❑12
Eclipse	❑5	Visual Cafe	❑13
IdeaJ	❑6	Websphere Studio	❑14
JCreator	❑7	A text editor	❑15
JGrasp	❑8	Other (please specify below)	❑16

.....

A6 Which Integrated Development Environment (IDE) do you use the most? *(Please tick one box only)*

AnyJ	❑1	NetBeans	❑9
BlueJ	❑2	OptimalJ	❑10
JBuilder	❑3	Oracle JDeveloper	❑11
DrJava	❑4	Programmers Notepad	❑12
Eclipse	❑5	Visual Cafe	❑13
IdeaJ	❑6	Websphere Studio	❑14
JCreator	❑7	A text editor	❑15
JGrasp	❑8	Other (please specify below)	❑16

.....

A7 Which programming language do you have the most experience in? *(Please tick one box only)*

- Java ❑1
- Other (please specify below) ❑2

.....

B – JBUILDER UTILITY AND EASE OF USE

The following set of questions asks about the usefulness and usability of JBuilder used within ICT221/ICT521. *Please indicate your agreement or disagreement with each statement by ticking the appropriate response on the 1-to-5 point scale:*		1 = Strongly Disagree	2 = Tend to Disagree	3 = Neutral	4 = Tend to Agree	5 = Strongly Agree
B1	Learning how to use JBuilder is straightforward	❑1	❑2	❑3	❑4	❑5
B2	Once you have learned how to use JBuilder then producing Java software with it is straightforward	❑1	❑2	❑3	❑4	❑5
B3	Using JBuilder is enjoyable	❑1	❑2	❑3	❑4	❑5
B4	JBuilder gives me assistance in its use	❑1	❑2	❑3	❑4	❑5
B5	The amount of time and effort required to perform tasks using JBuilder is low	❑1	❑2	❑3	❑4	❑5
B6	I feel I am in control of JBuilder when I use it	❑1	❑2	❑3	❑4	❑5

Please continue on the following page

C – UNDERSTANDING / PRODUCTIVITY MEASURES

The following is a list of aspects of JBuilder used within ICT221/521 with two different sets of responses required.

In the 1st set of responses – (a) Rate how each aspect of JBuilder used in ICT221/521 improves or impairs your understanding of the course concepts: *(please indicate your response on the 1-to-5 point scale of improvement/impairment, or give a response of 0 if you have never used the feature)*

In the 2nd set of responses – (b) Rate how each aspect of JBuilder used in ICT22/521 improves or impairs your productivity in producing software: *(please indicate your response on the 1-to-5 point scale of improvement/impairment, or give a response of 0 if you have never used the feature)*

		Improves/Impairs Understanding (a)						*Improves/Impairs Productivity (b)*					
		0 = Never used this feature	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*	*0 = Never used this feature*	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*
C1	JBuilder's automatic code formatting (this indents code such as contents of methods or loops)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C2	JBuilder's automatic completion of words in programs (for example, entering "System." presents a list of all items in the System class, and "out" can be selected from this)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C3	JBuilder's display of parameter lists (when a method name is entered, JBuilder shows a list of the parameters that it requires)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C4	JBuilder's automatic creation of program code (this performs tasks such as adding a missing closing curly bracket after enter is pressed)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C5	JBuilder's code creation wizards (these perform tasks such as creating classes or methods)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C6	JBuilder's Refactor…Rename tool (where changing a variable name results in all other copies of that name being changed as well)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C7	JBuilder's integrated help system (this allows help to be viewed for example by highlighting an item and pressing F1 or going to the Help menu)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C8	JBuilder's automatic code colouring (this makes comments green, reserved words red, etc)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5

Please continue on the following page

Please continue to rate your improvements/impairments in understanding and productivity by ticking the appropriate responses to each statement on the 1-to-5 point scales:		*Improves/Impairs Understanding (a)*						*Improves/Impairs Productivity (b)*					
		0 = Not applicable	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*	*0 = Not applicable*	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*
C9	JBuilder's display of line numbers within program code	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C10	JBuilder's automatic syntax error reporting (this reports errors such as missing semicolons)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C11	JBuilder's code audit warnings (this warns of problems such as variables that are not used)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C12	JBuilder's deprecation warnings (where warnings are given when parts of Java that should not be used any more are included in programs)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C13	JBuilder's support for debugging operations through breakpoints and line-by-line execution	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C14	JBuilder's support for variable values to be viewed and modified when in debugging mode	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C15	JBuilder's automatic matching of brackets and braces (when a regular or curly bracket is selected, the matching bracket is highlighted)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C16	JBuilder's automatic generation of Javadoc comments (entering '/**' and return creates the comment with tags for parameters etc included)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C17	JBuilder's Javadoc integration (when you are viewing source code you can click on the tab marked 'Javadoc' to see how all the Javadoc comments look when turned into HTML)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5

Please continue on the following page

D – JBUILDER COMMENTS

D1 *Please describe the best aspect(s) of the JBuilder integrated development environment used within ICT221/521:*

.....

.....

.....

.....

.....

.....

.....

.....

D2 *Please describe the aspect(s) of the JBuilder integrated development environment used within ICT221/521 that are most in need of improvement:*

.....

.....

.....

.....

.....

.....

.....

.....

D3 *Please describe the least useful aspect(s) of the JBuilder integrated development environment used within ICT221/521:*

.....

.....

.....

.....

.....

.....

.....

.....

Please continue on the following page

E – PROGRAMMERS NOTEPAD UTILITY AND EASE OF USE

The following set of questions asks about the usefulness and usability of Programmers Notepad used within ICT221/ICT521. *Please indicate your agreement or disagreement with each statement by ticking the appropriate response on the 1-to-5 point scale:*		1= Strongly Disagree	2= Tend to Disagree	3= Neutral	4 = Tend to Agree	5 = Strongly Agree
E1	Learning how to use Programmers Notepad is straightforward	❑1	❑2	❑3	❑4	❑5
E2	Once you have learned how to use Programmers Notepad then producing Java software with it is straightforward	❑1	❑2	❑3	❑4	❑5
E3	Using Programmers Notepad is enjoyable	❑1	❑2	❑3	❑4	❑5
E4	Programmers Notepad gives me assistance in its use	❑1	❑2	❑3	❑4	❑5
E5	The amount of time and effort required to perform tasks using Programmers Notepad is low	❑1	❑2	❑3	❑4	❑5
E6	I feel I am in control of Programmers Notepad when I use it	❑1	❑2	❑3	❑4	❑5

Please continue on the following page

F – UNDERSTANDING / PRODUCTIVITY MEASURES

The following is a list of aspects of Programmers Notepad used within ICT221/521 with two different sets of responses required.

In the 1st set of responses – (a) Rate how each aspect of Programmers Notepad used in ICT221/521 improves or impairs your understanding of the course concepts: *(please indicate your response on the 1-to-5 point scale of improvement/impairment, or give a response of 0 if you have never used the feature)*

In the 2nd set of responses – (b) Rate how each aspect of Programmers Notepad used in ICT22/521 improves or impairs your productivity in producing software: *(please indicate your response on the 1-to-5 point scale of improvement/impairment, or give a response of 0 if you have never used the feature)*

		Improves/Impairs Understanding (a)						*Improves/Impairs Productivity (b)*					
		0 = Never used this feature	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*	*0 = Never used this feature*	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*
F1	Programmers Notepad's code indenting (when lines of code are selected, pressing tab moves them all one tab space to the right)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F2	Programmers Notepad's bookmarks (bookmarks can be defined on selected lines of program code; these can then be navigated to)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F3	Programmers Notepad's code folding (each block of code can be 'folded' into a single line and then later unfolded to its original state)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F4	Programmers Notepad's case conversion (allowing selected code to be made upper or lower case)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F5	Programmers Notepad's automatic matching of brackets and braces (when a regular or curly bracket is selected, the matching bracket is highlighted)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F6	Programmers Notepad's search and replace	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F7	Programmers Notepad's compiler integration (allowing the Java compiler to be accessed within Programmers Notepad)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F8	Programmers Notepad's automatic code colouring (this makes comments green, reserved words blue, etc)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F9	Programmers Notepad's display of line numbers within program code	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5

Please continue on the following page

G – PROGRAMMERS NOTEPAD COMMENTS

G1 *Please describe the best aspect(s) of Programmers Notepad used within ICT221/521:*

.....

.....

.....

.....

.....

G2 *Please describe the aspect(s) of Programmers Notepad used within ICT221/521 that are most in need of improvement:*

.....

.....

.....

.....

.....

G3 *Please describe the least useful aspect(s) of Programmers Notepad used within ICT221/521:*

.....

.....

.....

.....

.....

H– JBUILDER AND PROGRAMMERS NOTEPAD

H1 If you had free choice, which development environment would you prefer to have used to learn programming? *(Please tick one box only)*

JBuilder only ❑1
Programmers Notepad only ❑2
Both JBuilder and Programmers Notepad ❑3
Other (please specify below) ❑4
.....

H2 If you had free choice, which development environment would you prefer to use to do programming now? *(Please tick one box only)*

JBuilder only ❑1
Programmers Notepad only ❑2
Both JBuilder and Programmers Notepad ❑3
Other (please specify below) ❑4
.....

Thankyou for completing this survey