# An Exploratory Assessment of the Pedagogical Effectiveness of a Systems Development Environment

**Peter Meso**
**Jens Liegle**
Dept. of Computer Information Systems
J Mack Robinson College of Business,
Georgia State University
35 Broad Street, Atlanta, GA 30303
pmeso@cis.gsu.edu jliegle@cis.gsu.edu

**ABSTRACT**

We employ the theory of technology acceptance to assess the suitability and fit of a new systems development environment, the .NET suite of technologies, as a pedagogical tool for teaching a technical information system (IS) course. The performance of students who adopted this technology for the completion of a class project requiring them to design and build an object-oriented distributed-system is compared to that of students who opted for the more conventional technologies (J2EE). Results of this study indicate that the factors that led to the selection of .NET over the other technologies were consistent with the technology acceptance theory: Those project-teams that opted for .NET performed as well as the J2EE teams on the implementation/deployment part of the project, but reported significantly less technical difficulties than those who used the conventional technologies. This study suggests the effectiveness of the technology acceptance theory and similar IT innovation diffusion theories as approaches for assessing the pedagogical fit and suitability of specific IT for teaching specific IS courses.

**Keywords**: Technology Acceptance Model, TAM, J2EE, .NET, Software Design

## 1. INTRODUCTION

The information technology revolution, an ongoing evolution, has a direct bearing on the nature of instruction provided to business students. This is especially true for students of software engineering – be they in the business, information sciences, or computer science disciplines. As such, new computing technologies that promise to change the computing landscape significantly are a constant concern to software engineering educators. However, it is usually cumbersome and difficult to determine whether or not to adapt a new software engineering technology as a pedagogical tool and when to undertake such an adoption decision. For one, adopting the new technology requires significant investments not only in the new technology itself, but in the know-how and instructional material for the technology. It also calls for a re-tooling and restructuring of course content to fit the new technology. Further, care is needed to ensure that the pedagogy does not migrate in focus towards teaching the new technology, but is able to use the new technology to expound and convey the core principles of the subject area – say programming or systems design for example.

As is the case in most scientific courses, the teaching of software engineering courses requires the use of definite technologies either within a laboratory setting or individually outside of the classroom (Liegle and Madey, 2003). However, whereas many scientific courses require instructor's attention to technical support only during the class period, the nature of information technology (IT), especially in the present Internet era, demands that an Information Systems (IS) instructor attends to technical problems and systems administration issues on an on-going basis – even when not on duty. This tends to be the case even where separate personnel are available to perform technical-support duties, because students naturally communicate their technical problems directly to the instructor. Further, the perception that students have about the effectiveness of the instructor – and the true worth of the course – are directly affected by how effectively the course (pedagogical) technology functioned in terms of its ease of use and its usefulness in allowing the students to fulfill pertinent course requirements (Meso and Liegle, 2003). The effectiveness with which technology-related difficulties that the students encounter are resolved also impacts their perception about the course and its instructor. These perceptions have a direct bearing on the instructor's evaluation by students – evaluations that are heavily relied upon at most institutions to assess the teaching effectiveness of instructors (Liegle and Johnson, 2003). Therefore, identifying useful, effective and relevant IT pedagogical tools that are actually well received

by the students is of significant importance to the instructor. Such tools/technologies may leverage the performance and effectiveness of the instructor.

The complexity of the technology, or its cumbersomeness, may also mitigate the students' ability to grasp and understand the core body of knowledge being disseminated in the course. According to pedagogical research, effective teaching tools and technologies enhance the learning capability of students and make the mastery of difficult principles simpler (Liddle, Brown et al., 1995; Janicki and Liegle, 2001). Research in this area also points out that the teaching tools and technologies that prove to be effective in most cases are those that:

1. are easy to use and easy to learn,
2. map a clear and direct path from the problem to its correct solution, allows for hand-on-learning or learning-by-doing rather than passive learning such as demonstrations by an instructor, and
3. minimize the technological barriers between student and the core-knowledge or principles being disseminated to the student (Janicki and Liegle, 2001).

Therefore, our objective in this study is to assess the suitability and fit of a new systems development environment as a pedagogical tool for teaching a technical IS course. We do this by comparing the new technology of Visual Studio.NET to the more conventional technology of J2EE. The .NET suite of technologies is selected because it is a new technology recently launched to the public and, so far, it has experienced minimal use as a pedagogical tool at our institution. The conventional technology in this study is Java and the J2EE framework. J2EE has been in use as an enterprise-capable software engineering and information systems development environment for over a decade. It is well entrenched as the development platform of choice for web applications and multi-tier enterprise information systems (Sun Microsystems, 2002; Oestereich, 2002; Hall, 2003). For the past five years, J2EE has been the pedagogical tool of choice in the teaching of systems design at the site where this study was conducted.

The paper is structured as follows: The ensuing section provides a discussion of the use of technology acceptance theory in past studies on IT diffusion, and explains the variables that we adopted from this theory to assess the pedagogical fit of a new software development environment. Part 3 presents the research methodology and hypotheses. Part 4 reports the results of an experiment while part 5 provides inferences drawn from these results and discusses the contributions of the study. Part 6 presents a discussion of the results, while Part 7 draws conclusions and provides directions for future research.

## 2. INTRODUCTION OF THE RESEARCH MODEL

When instructors are faced with the decision of whether to introduce a particular new technology into the classroom, they can use several methods to evaluate the 'goodness' of a particular technology as a pedagogical tool. First is a feature comparison using weighted scores. A second alternative would be to poll actual users or a set of experts. These two methods have their limitations: In the end, in order to come up with an unbiased decision, one needs to be an expert in all the technologies, and this is usually not the case when new ones are introduced.

Another problem of the previous two methods is that they represent a "top-down" decision making approach, where student opinion is not taken into consideration. Conventionally, students will be attracted to a technology that is easy to use and directly relevant to the course requirement tasks that they must complete, or to one that they perceive as bearing these traits. Where students have a choice between two or more technologies, a comparative assessment of their reactions to each becomes possible. This provides a means for determining the technology that best fits the course based on the feedback received from the students. Therefore, a third approach is to assess students' acceptance of a technology. This is especially relevant, since student acceptance will ultimately influence their evaluation of the course and the instructor.

The argument can be made that actual usage of a technology is a strong indicator of acceptance of a technology (Davis, 1989). Therefore, actual usage of a technology can be a determinant of its appropriateness for the course. However, within the context of using a new systems development environment as a pedagogical technology, there are a number of problems related to measuring its perceived or actual usage. Traditional measures such as frequency of use, number of features used, or amount of time spent using a technology as employed in previous technology acceptance studies (Szajna, 1996; Gefen and Straub, 1997) are inadequate for a number of reasons. For example, someone might have to use the systems development environment for a long time due to lack of knowledge on how to use it, or, on the other hand, because they actually implement a large number of features in their project. Further, the number of times an application is opened, and the number of features used within the system may not be indicative of qualitative usage.

Since qualitative usage measures can not be employed objectively to evaluate students' acceptance of a new technology, alternative measures have to be used. Fortunately, the theory of technology acceptance presents a causal model called TAM (Technology acceptance model, see figure x) that identifies two predictor variables for actual usage of a technology (Davis, 1989). These two variables are ease of use and usefulness respectively.

The Technology Acceptance Model (TAM) is the leading theory used to explain adoption of information technology by individuals in business and industry (Gallivan, 2001; Chircu et a., 2000; Straub et al., 1997). Research abounds on the use of TAM in explaining individual adoption and acceptance of IT and the antecedent and consequent factors that propel such diffusion within groups and organizations (Davis, 1989, Davis et al. 1989; Szanja, 1996; Agrawal and Prasad, 1997; Thompson, Higgins and Howell, 1991; Moore and Benbassat; 1991; Karahana, Straub and Chervany, 1999;

Gallivan, 2001). While TAM has been widely used in IS diffusion research, it has rarely been used to explain the selection of an IT tool or technology to support the teaching of a technical IT course. There has been little research focusing on why certain IT tools and technologies are more favored than others in the teaching of certain IT skills and domain knowledge. Instead, existing studies have tended to compare particular technologies or skill sets with respect to predefined outcomes such as subject's productivity, subject's cognitive performance and some output artifacts (e.g. higher quality analysis diagrams, higher quality program code, etc) (Basili et al., 1999; Burton-Jones and Meso, 2002; Havelka, 2003; Howard et al., 1999; Wang, 2003; Morris et al., 1999; Vessey and Conger, 1994).

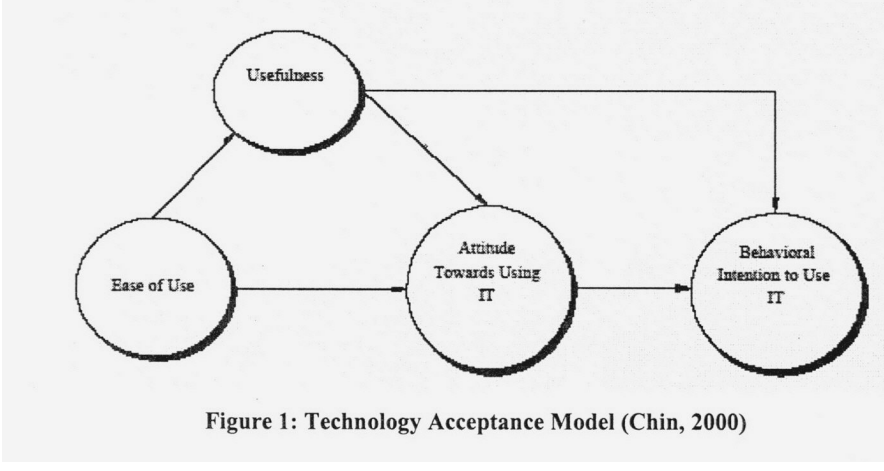



**Figure 1: Technology Acceptance Model (Chin, 2000)**

The Technology Acceptance Model states that the factors that propel the diffusion of an Information technology are its ease-of-use and its usefulness (Davis, 1989; Chin, 2000; Gallivan, 2001). TAM has been used to explain diffusion using user-perception measures as well as actual usage measures. Within the context of an IT course, we expect that students will be attracted to that technology that is easy to use and directly relevant to the course requirement tasks that they must complete, or to that technology that they perceive as bearing these traits. Therefore, assessing the reactions of students toward a particular technology can determine the effectiveness of that technology as a pedagogical tool for the course in question.

In past TAM studies, the ease of use and usefulness variables have been operationalized as either perceived or actual measures (Davis, 1989; Deane, Podd, and Henderson, 1998; Henderson and Divett, 2003; Szajna, 1996;). Perceived measures have been more frequently employed than actual measures due to the fact that actual measures are difficult to obtain (Deane, Podd, and Henderson, 1998). In fact, there is a paucity of research in TAM using actual behavioral measures instead of perceived measures (Henderson and Divett, 2003). However, there have been problems reported with the use of perceived ease of use instead of actual ease of use (Henderson and Divett, 2003). One of the rare studies using actual measures showed that they are highly correlated with perceived measures (Deane, Podd, and Henderson, 1998). The use of surrogate measures in terms of perceived usefulness or ease of use is actually based on an assertion by Davis (1993) that perceived measures are an accurate though not direct measure of technology acceptance. In summary, perceived measures are said to be appropriate in situations where users have yet to use the technology, in other words, pre-implementation (Szajna, 1996; Deane, Podd, and Henderson, 1998). In post-implementation scenarios, actual measures have been found to be superior to perceived measures (Henderson and Divett, 2003; Szajna, 1996).

In our case, students had to take two consecutive programming courses of a given track before taking the systems development course. The currently offered tracks include Java, C++, and Visual Basic. For Java, the J2EE environment was taught, while Visual Studio was used for both C++ and Visual Basic. For the systems design course, however, the new Visual Studio.NET environment had just become available for students to use, and was new to all students. Due to the steep learning curve for both J2EE and Visual Studio, we did not expect students who had taken the Java Track using J2EE to develop their system using Visual Studio, or any student with Visual Studio training to suddenly use J2EE.

Thus, the use of perception measures in evaluating the ease of use and usefulness were not applicable to our study, since students had already selected their programming track and completed two courses in their respective track prior taking the systems design course, and as such already selected one of the two system development environments.

Given the fact that we could not capture actual usage through any meaningful measures, we employed the predictor variables ease of use and usefulness, and since we evaluated the technology in a post-implementation setting, we opted to operationalize these variables through actual measures.

## 3. THEORY AND HYPOTHESES

According to TAM theory, two factors propel the diffusion of an information technology – its ease of use and its usefulness.

### 3.1 Ease of Use

With respect to technologies for systems design and construction, the technology is considered easy to use if it allows the designer to effortlessly (Davis, 1989; Davis, 1993), or in simple straightforward steps,

- translate design models (blue prints) into program code,
- ensure that the specifications outlined in the design models (blue prints), remain are mapped into the functional program code without any loss in structure, logic, or elements stipulated in the design models

- integrate the multiple program-code modules or components into one seamless and unitary system, and
- deploy the system into production.

Said otherwise, a technology is easy to use if it provides intuitive-like features that include cues, graphical user interfaces, wizards, menus, and similar artifacts to guide the user through well established automated procedures of converting specifications into functional program modules or components, to integrate these components into a functional unitary system, and to deploy the system into production. Such a system simplifies the systems building function by allowing the user to focus on the systems solution, rather than on how to operate the technology. Therefore, ease-of-use relates to the navigability, user-interfaces, automated capabilities, and the functionality of the technology – the ability to quickly and effortlessly discover how the technology works and how to use it.

The .NET framework comes with a new integrated development suite, Visual Studio.NET (VS.NET). VS.NET is a significant step forward from prior releases, in that it not only provides the programmer with an extensive online help, a fully integrated editor with intellitype – auto completion of method arguments –, excessive wizard support and graphical drag/drop construction of user interfaces such as menus, windows, buttons etc., but it also does this in a totally integrated manner. The underlying byte code allows developers to write part of the code in one language, and use it from another language, i.e. define classes in C# and use them in a Visual Basic.NET program. The integrated debugger will switch from one language to another in real-time (Cooper 2002).

The consequences for a development team are that each participant can develop in the programming language they are most familiar with, and the different parts can be easily integrated and debugged – without the overhead of COM (Component Object Model). This was especially helpful for the students of this study, since they came from one of three programming tracks (C++, Java, and Visual Basic) and often did not know the other languages at all. And since students from either track were familiar with the basic Visual Studio environment and Windows technology in general, the initial learning curve for this tool was relatively low.

### 3.2 Usefulness

Usefulness relates to the ability of the system to do what it was designed to do. A tool or technology is useful if the user is able to achieve pre-defined end-goals by following the pre-defined directions of how to use the tool or technology. Vessey and Conger (1994), in proposing the theory of cognitive fit, state that a technology is most useful when it is matched correctly with the task to be completed using that technology. Where the fit between the technology and the task is poor, the technology's usefulness deteriorates. Therefore, in the case of the use of a software development environment (SDE), if a user is not able to successfully implement a system's design solution when using a specific SDE, then that tool is less useful. In other words, the usefulness of a tool relates to the capabilities of the technology to deliver the anticipated outputs.

In the context of systems development, these outputs are a functioning production system that meets established quality standards. Therefore an IS technology for building IS solutions is useful if it enables the designer to create information-system solutions that are: efficient (solutions with minimal lines of code and high through-put), robust, scalable, reliable, capable of rigorous data quality and integrity (data-validation, data verification, run-time error detection and correction, etc) and secure.

For Visual Studio.NET, scalability is superb, since it is designed as an enterprise-level programming environment. Efficiency is greatly enhanced through the use of multiple wizards that generate most of the code for the programmer, i.e. user interfaces and database connectivity. Error detection etc. is supported by the most powerful debugger that exists to date, which allows the programmer to start debugging a client side application and seaming less switch to the code of the server side program – in the same session (Cooper, 2002).

J2EE technologies, on the other hand, are traditionally known to be difficult to learn and use. This has largely been to the fact that most use text-based interfaces, and the various tools required to complete a project are usually not integrated in one comprehensive suite of programs. In recent years, an increasing number of Java integrated development environments (IDE) vendors have built graphic user-interfaces into their IDEs and some have also implemented wizards to simplify the systems construction, testing and deployment processes. Examples include IBM's Websphere Studio (IBM, 2004), and Sun Microsystems's NetBeans (Sun Microsystems, 2002; Sun Microsystems, 2004). Most still require that a separate deployment tool be initiated when a developer transitions from code-generation and testing to system deployment. This notwithstanding, J2EE remains a powerful and robust development framework for enterprise information systems.

### 3.3 Hypothesis

Given the newness of .NET and our inexperience with its use as a pedagogical tool, we selected to use a neutral hypothesis when assessing the efficacy of .Net as a tool for teaching object oriented systems design. Therefore we hypothesized that .NET will be as easy to use as J2EE technologies. That is, users will not require more technical information and know-how about how the .NET technology works in order to successfully use .NET to develop an IS system solution than they would require were they to use J2EE technologies:

> *H1: There will be no significant difference in the ease-of use measures between student groups who use .NET and those who use J2EE technologies.*

Given that both technologies have been developed to address object-oriented enterprise systems development, and both have capabilities for developing web-based systems, we hypothesized that there would be no significant difference in

the usefulness of both technologies for developing systems solutions. In other words, the capabilities demonstrated by .NET in facilitating the construction of a distributed web-based object-oriented information system would also be evident in J2EE technologies.

*H2: There will be no significant difference in usefulness measures between student groups who use .NET and those who use J2EE technologies.*

## 4. RESEARCH METHODOLOGY

In order to test the hypothesis, we chose to use a quantitative field study approach. This enabled us to operationalize the research constructs within normal teaching practice. The following subsections describe the details of our methodology.

### 4.1 Subjects

This study was conducted at a leading south-eastern university. It was designed as a semester long systems design project, which was repeated over three semesters. Participants in the study (study's subjects) were required to design and build an object-oriented web-based distributed enterprise information system. Replication was chosen as a strategy for validating the findings of the study. Therefore, the same study was replicated at the same university, using subjects enrolled in the same course over a period of three consecutive semesters.

In all the studies, the subjects were senior-year undergraduate students that were enrolled for the systems design course. All were computer information systems majors. They were drawn from sections of the same course taught by the same instructor.

### 4.2 Experiment

The instructor provided the narrative for the project, describing similar business cases. The project had four parts to it: a conceptual design part (part 1), a technical design part (part 2) and a systems-building part (part 3), followed by a technical presentation (part 4). All teams used the rational-rose software as the standard CASE tool in the completion of the conceptual design part and the technical design part of the project.

Subjects were organized into project teams for purposes of completing the design project. The project teams were formed by students self-selecting themselves into groups of three to five members. In the original study, a total 13 teams were formed, followed by 6 Teams in the 2nd semester, and 7 teams in the 3rd semester for a total of 26 teams.

The nature of the study, being to use TAM theory to assess the suitability and fit of a new technology for the pedagogical support of a technical IS course, mitigated the need for allocating technologies to the project teams. Therefore each project team was allowed to select any technology or set of technologies of their choice for the systems-building (implementation) part of the project. The technologies they selected thus determined what treatment group they belonged to. In total four different technologies were selected. These were .NET (10 teams), J2EE technology (11 teams), ASP 6.0 (3 teams) and HTML/CGI (2 team). Students also made a presentation of their project at the end of the semester. In the analysis of the results, project teams that did not use .NET or J2EE were dropped from the analysis.

All teams used the rational-rose CASE tool to develop the conceptual and technical design solutions, and were at liberty to select technologies of their choice for the systems-building (implementation) part of the project. The selected technology thus determined their treatment group.

### 4.3 Measurements for Ease of Use and Usability

The quantitative field study approach that we employed provided for natural data collection in the form of actual student performance scores instead of only perceived measures. The use of actual performance measures such as post-test quiz scores or time as a surrogate for measuring the ease of use and/or usability of an object oriented technology is common practice (Lui and Gradon, 2002; Agarwal et al., 1996; Venkatesh and Davis, 1996; Vessey and Conger, 1994; Sharp and Griffyth, 1999).

For our study, we measured usefulness based on the performance of the student project-teams on the third part (part 3) of a four-part semester long teaching project on object oriented systems design. Here (part 3), students constructed the system solution according to the conceptual design specifications developed in part 1 and the technical design specifications developed in part 2 of the project respectively. Scoring was done by the instructor using a scale of 0 to 100.

The ease-of use of the technology was measured based on the student-project-team's performance on their presentation assignment (part 4). In this assignment, each team demonstrated the technology they used to build the system, outlining how that technology supported the generation of program code, the debugging and testing of the same, the integration of the various programs into one system and the deployment of the final solution as a production system. Teams that demonstrated a superior understanding of the development tool by being able to demonstrate its various features and how they used these features in developing their solution received a high score. Those that had difficulties demonstrating the use of the tool, and/or reported having had problems using the tool to develop their system received a low score. Scoring was done by the instructor using a scale of 0 to 100.

As control variables, students of both the original study and the replication completed three examinations in the course of the semester. We used the results from the average total exam performance of each group, and their performance on the conceptual design part of the project to test for the differences among the treatment groups in terms of their knowledge of the subject matter.

## 5. RESULTS

Descriptive statistics were evaluated for the student-group performances on each part of the project and on the presentation assignment. First we ran tests to confirm if the data collected from the study was normally distributed. This was done because the treatment groups in this study had not been deliberately designed. Rather, in keeping with the principle of diffusion, the subjects had been allowed to select their own development technologies – and these technologies established the respective treatment groups for the study. Skewness and kurtosis tests confirmed that the data used as measures for each of the hypotheses above, namely Performance on Part 3 of the project, and Performance on the Presentation, was normally distributed (Table 1 and 2). Therefore, we were able to use it in the ANOVA (Analysis of Variance)(Hair et al, 1998, pg. 332) analysis to statistically analyze the differences between the treatment groups, with respect to the hypotheses.

An analysis of their performance for project and presention over the three semesters for the .NET (See Table 3) and the J2EE teams (See Table 4) revealed that there was no significant difference (p-values of .054 and .214 for .NET, and .142 .125 for J2EE respectively) with the treatments over the three semesters.

Since there was no significant differences within treatments with respect to their presentation and project scores allows for the aggregation of the data when evaluating the hypotheses (see Tables 3 and 4), the pooled data over the three semesters was used for the analysis of the hypothesis. Therefore, we reanalyzed descriptive statistics for the subjects at team-level. Table 5 presents these statistics for the two treatment groups. Consequently, the following analysis uses team-level data. Across the three semesters, a total of ten (10) teams used .NET, and eleven (11) teams used J2EE (See Table 5). Teams that used other technologies such as ASP6.0 or CGI/Perl were excluded from the analysis.

Analysis of variance (ANOVA) tests for the pooled data confirmed support for hypotheses H1 and H2. Table 6 presents these results. Although the .NET group performed worse on the project than the J2EE (83.8 vs. 86.0), this difference was not significant (p=0.686). This result indicates that the two technologies ranked equally regarding the ease-of-use construct.

Surprisingly, the results of the Presentation score were just reversed, with .NET teams performing better than the J2EE teams (86.4 vs. 82.), but yet again the difference was not statistically significant (p=0.354). Again, this result indicates that the two technologies ranked equally regarding the usefulness construct.

**Table 1: Test for Normality of Data in the Original Study's Research Variables**

| | N | Minimum | Maximum | Mean | Std. Dev | Kurtosis | |
|---|---|---|---|---|---|---|---|
| | | | | | | Statistic | Std. Err |
| Exam Total | 62 | 16.40 | 29.00 | 25.0258 | 2.3343 | 2.062 | .599 |
| Conceptual Design Score | 62 | 90.00 | 100.00 | 97.0000 | 3.1100 | -.547 | .599 |
| Implementation Score | 62 | 64.00 | 99.00 | 77.5000 | 11.6348 | -.758 | .599 |
| Presentation Score | 62 | 80.00 | 98.00 | 88.1129 | 5.7747 | -1.148 | .599 |

**Table 2: Test for Normality of Data in Replication Study's Research Variables**

| | N | Minimum | Maximum | Mean | Std. Deviation | Kurtosis | |
|---|---|---|---|---|---|---|---|
| | | | | | | Statistic | Std. Error |
| Exam Total | 26 | 22.20 | 28.50 | 25.3385 | 1.5297 | -.101 | .887 |
| Conceptual Design Score | 26 | 71 | 85 | 80.65 | 4.23 | 1.103 | .887 |
| Implementation Score | 26 | 87 | 98 | 92.73 | 4.07 | -1.396 | .887 |
| Presentation Score | 26 | 90 | 100 | 95.08 | 3.72 | -1.280 | .887 |

**Table 3: ANOVA Comparison of Project and Presentation Scores across Three Semesters for the .NET Group**

| | | Sum of Squares | df | Mean Square | F | Sig. |
|---|---|---|---|---|---|---|
| Project | Between Groups | 1025.517 | 2 | 512.758 | 4.543 | .054 |
| | Within Groups | 790.083 | 7 | 112.869 | | |
| | Total | 1815.600 | 9 | | | |
| Presentation | Between Groups | 184.067 | 2 | 92.033 | 1.939 | .214 |
| | Within Groups | 332.333 | 7 | 47.476 | | |
| | Total | 516.400 | 9 | | | |

**Table 4: ANOVA Comparison of Project And Presentation Scores Across Three Semesters for the J2EE Groups**

| | | Sum of Squares | df | Mean Square | F | Sig. |
|---|---|---|---|---|---|---|
| Project | Between Groups | 352.533 | 2 | 176.267 | 2.512 | .142 |
| | Within Groups | 561.467 | 8 | 70.183 | | |
| | Total | 914.000 | 10 | | | |
| Presentation | Between Groups | 676.930 | 2 | 338.465 | 2.582 | .125 |
| | Within Groups | 1310.762 | 10 | 131.076 | | |
| | Total | 1987.692 | 12 | | | |

**Table 5: Descriptive Statistics of pooled Treatment Groups**

| | 1=Net 2=J2EE | N | Mean | Std. Deviation | Std. Error Mean |
|---|---|---|---|---|---|
| Project | 1 | 10 | 83.80 | 14.20 | 4.49 |
| | 2 | 11 | 86.00 | 9.56 | 2.88 |
| Presentation | 1 | 10 | 86.40 | 7.57 | 2.40 |
| | 2 | 11 | 82.00 | 13.08 | 3.95 |

**Table 6: ANOVA of Pooled Treatment Groups**

| | t | df | Sig. (2-tailed) | Mean Difference | Std. Error Difference | 95% Confidence Interval of the Difference | |
|---|---|---|---|---|---|---|---|
| | | | | | | Lower | Upper |
| Project | -.412 | 15.564 | .686 | -2.20 | 5.34 | -13.54 | 9.14 |
| Presentation | .953 | 16.275 | .354 | 4.40 | 4.62 | -5.37 | 14.17 |

These findings show that that there was no significant difference between the .NET and the J2EE suite of technologies in both ease-of-use and usefulness.

## 6. DISCUSSION

TAM allowed us to empirically and objectively compare the effectiveness of a new technology. The findings, that the subjects who used the new technology in this study performed on par with the subjects who used the existing technology provides grounded evidence that the new technology is as effective a pedagogical tool as the existing technology. It allowed the students who used it to achieve the stipulated course objective of building a quality solution to a systems design problem and perform as well at the task than those who did not. However, this conclusion is limited to the specific context in which the emergent technology was compared to existing technologies and cannot be generalized to all IS courses, or even to systems design courses that take on a different focus (e.g. process oriented as opposed to object-oriented design).

As stated earlier, these findings simply confirm the suitability of the new and emergent technology as a pedagogical tool for a well-defined technical IS course. They do not provide support for the notion that the emergent technology is a superior technology to the existing technologies.

These findings further indicate that using TAM may facilitate the identification and selection of suitable IT tools for teaching technical IS courses. Selection of such technologies, for which .NET is an example, provides gains to instructors of technical information systems courses by mitigating the course-administration burden that such courses place on the instructor.

This paper would be incomplete without some documented observations of the instructor made during the lifetime of the course. In retrospect, these observations may explain or point to the reasons why the student teams that adopted .NET may have performed as well as their J2EE colleagues.

First, the student-groups that selected to use .NET were able to install and configure their selected technologies without the instructor's assistance. On the other hand, all but one of the student groups that used J2EE technologies experienced installation and set-up problems significant enough to have them consult with the instructor. This was likely due to the complex nature of J2EE-based technologies, most requiring add-ons and components from different vendors in order to function. The most common problems that students had regarding technology installation and configuration had to do

with establishing database connectivity and getting the server-side run-time environment to function correctly.

Most of the course preparation and administration time in technical IS courses is spent on issues relating to installing and configuring the IS technologies to be used by the class, and administering/maintaining these through the lifetime of the course. Thus the first lesson learned from this study is that a technology that allows the students to perform these tasks simply and effectively, without excessive dependence on the instructor, greatly reduces the technological burden of delivering the course. Both the instructor and the students benefit in this set-up. Students get to learn the nuances of a technology that otherwise only the instructor knows, and gain confidence in technology management. The instructor, on the other hand, is able to allocate more course time to addressing subject knowledge issues rather than administrative issues. As such, providing students with more in-depth knowledge of the subject matter becomes not only possible, but feasible, even within the time constrains of a single semester.

Second, groups that selected .NET required little or no assistance in solving problems regarding how to use the IS tool, and how to generate, compile and run software code using the tool. Their consultations largely involved translation of UML symbols, diagrams and logic into appropriate software-code and persistent date repositories. Student groups that selected to use J2EE had significantly more difficulty in learning how to use the integrated development environment they had selected without assistance from the instructor. This tended to inhibit their focus on, and performance in, completing the actual IS development project. Said otherwise, because groups that opted for .NET and simple text editors spent less time in learning tool specific knowledge, they were able to allocate more time to learning domain specific knowledge (systems design knowledge needed to complete the IS project). On the other hand, those groups that opted for J2EE-based tools spent so much time learning tool-specific knowledge that they had little time left to concentrate on domain specific knowledge.

Therefore, the second key lesson learned from this study is that selecting an easy-to-use, easy to learn IS tool as per the parameters of TAM allows for more, if not all, class time to be used in teaching the core knowledge and principles pertinent to the technical IS course being taught rather than in teaching how to use a specific IS tool that supports, facilitates, or implements the learned core-knowledge. The result is a purer pedagogical coverage of the core-knowledge in the teaching of technical information systems courses.

Still, more studies need to be conducted before our findings can be generalized. In addition, one should keep in mind that .NET is a fairly new technology and that students who opted for using .NET for their project are therefore early adopters – a group that often consist of students that are very comfortable with new technology and usually the better IS students overall.

## 7. CONCLUSION

We employed the Technology Acceptance Model (TAM) of information technology (IT) innovation diffusion to assess the suitability and fit of a new IT, the .NET suite of technologies, as a pedagogical tool for teaching a technical information systems (IS) course. Results of this study indicate that the factors that led to the selection of .NET over the other technologies were consistent with the TAM theory. Those project-teams that opted for .NET performed equally well on the implementation/deployment part of the project, and reported significantly less technical difficulties than those who used the conventional technologies. They also performed as well as the other teams on the presentation of the project. The results supported the use of TAM for selecting a new technology such as .NET based on the usefulness and the ease-of-use constructs. The results suggest that .NET is an appropriate technology for this particular course. However, before the results can be generalized, more research must be done.

Our study was constrained by a number of limitations: First, we did not consider or measure different learning styles of the students. Learning styles can affect the usage of a system; however, little research has been done to show how learning styles impact ease of use. Second, our study did not focus on assessing reflective learning outcomes. Mumford (1985) argues that individuals can record and review incidents and events and - through analysis and critical reflection - can develop different approaches to guide future action. Due to environmental constraints, our study was limited to a one time assessment of the subjects, which constrained the observation of any long term actions by the subjects. Third, we did not employ an instrument to capture perceived measures. Using such an instrument could have validated the claim by Davis (1989) that perceived measures are a viable substitute for actual measures. Fourth, the sample size in the study was relatively small, in particular the number of groups. We tried to compensate for this limitation by replicating the study. Similar results in both the original and the two replications provide some validation for the study. These limitations provide possible and viable extensions of this study. We propose future studies investigate the long term usage of new technology and its effects on an individuals' learning.

This paper demonstrates that TAM can be used as a basis of identifying the suitability and fit of a particular technology for teaching a select body of IS knowledge. Using the TAM framework provided us with an established framework with well established decision criteria for evaluating technology before adopting it for use in the classroom. This is especially relevant when the technology is new and its benefits vs. the more established technologies are largely unknown and untested.

## 8. REFERENCES

Agarwal, R. and J. A. Prasad. "Conceptual and Operational Definition of Personal Innovativeness in the Domain of

Information Technology," Information Systems Research, Vol. 9, No. 2, (1998), pp. 204-215.

Basili, V., Shull, F., and Lanubile, F., (1999). "Building Knowledge Through Families Of Experiments," IEEE Transactions on Software Engineering, Vol. 25 No.4, pp. 456-473

Burton-Jones, Andrew and Peter Meso, "How Good are These Uml Diagrams? An Empirical Test of the Wand and Weber Good Decomposition Model," 23rd Annual International Conference on Information Systems, 15 – 18 December 2002, Barcelona, Spain

Chin W. "Partial Least Squares for Researchers: An Overview and Presentation of Recent Advances Using the PLS Approach," 2000, http://disc-nt.cba.uh.edu/chin/indx.html

Chircu, Alina M; Kauffman, Robert J, "Limits to value in electronic commerce-related IT investments," Journal of Management Information Systems, Vol. 17 No 2, 2000, 59-80

Cooper, J. "Visual Basic Design Patterns. VB 6.0 and VB.NET," Addison Wesley, 2002

Cooper, Randolph B.; Bhattacherjee, Anol, "Preliminary Evidence for the Effect of Automatic Responses to Authority on Information Technology Diffusion," Database for Advances in Information Systems, Vol. 32 No. 3, 2001, pp. 36-50

Davis, F.D., "Perceived Usefulness, Perceived Ease-of-Use and User Acceptance of Information Technology," MIS Quarterly, Vol. 13 No. 3, 1989, pp. 319-339.

Davis, F.D., Bagozzi, R.P, and Warshaw, RR., "User Acceptance of Computer Technology: Comparison of Two Theoretical Models," Management Science, Vol. 35 No. 8, 1989, pp. 982-1003.

Davis, F.D. "User acceptance of Information Technology: System characteristics, user perceptions, and behavioral impacts," International Journal of Man-Machine Studies, Vol. 38, 1993, pp. 475-487.

Deane, F., Podd, J., Henderson, R. "Relationship between self-report and log data estimates of information system usage," Computers in Human Behavior, Vol 14, 1998, pp. 621-636.

Gallivan, Michael, "Organization Adoption and Assimilation of Complex Technological Innovations: Development and Application of a New Framework," Database for Advances in Information Systems, Vol. 32 No. 3, 2001, pp. 51-85

Hair, J., Anderson, R., Tatham, R., Black, W. "Multivariate Data Analysis," 5th Edition, 1998, Prentice Hall, Upper Saddle River, NJ.

Hall, M and Brown, L., "Core Web Programming," second edition, Prentice Hall (Sun Microsystems Book Series), 2003

Havelka D., "Predicting software self efficacy among business students: A preliminary assessment," Journal of Information Systems Education, Vol. 14 No. 2, 2003, pp. 145-152

Henderson, R., and Divett, M. "Perceived usefulness, ease of use and electronic supermarket use," International Journal of Human-Computer Studies, Vol 59 No. 3, September 2003, pp. 383-395

Howard, G., Bodnovich, T., Janicki, T., Liegle, J., Klein, S., Albert, P., and Cannon, D. "The Efficacy of Matching Information Systems Development Methodologies with Application Characteristics - An Empirical Study." Journal of Systems and Software, Vol. 45 No. 3, 1999, pp. 177-195

IBM, WebSphere Studio Application Developer, IBM Corporation, White Plains, New York , 2004, http://www-306.ibm.com/software/awdtools/studioappdev/

Janicki, Thomas N. and Liegle, Jens O. "Development and Evaluation of a Framework for Creating Web-based Learning Modules." Journal of Asynchronous Learning Networks, Vol. 5 No. 1, June 2001, pp. 58-84

Karahanna, E., Straub, D.W., and Chervany, N.L., "Information Technology Adoption Across Time: A Cross-Sectional Comparison of Pre-Adoption and Post-Adoption Beliefs," MIS Quarterly, Vol. 23 No. 2, 1999, pp. 183-213.

Lederer, A.; Maupin, D.; Sena, M.; and Zhuang, Y. "TAM and the World Wide Web," Proceedings of the Association for Information Systems (AIS) Americas Conference, Indianapolis, IN, 1997. http://hsb.baylor.edu/ramsower/ais.ac.97/papers/lederer2.htm

Liddle, J., Brown, K., Slater, A., MacDonnchadha, S. : "Utilizing Multiple Training Strategies within Intelligent Industrial Training Systems," AI-ED 95 workshop on Authoring shells for intelligent Tutoring Systems, Washington D.C August 1995, 1-7

Liegle, J. and Johnson R. "A Review of Premier Information Systems Journals for Pedagogical Orientation," Information Systems Education Journal, Vol. 1 No. 8, 2003, http://isedj.org/1/8/. ISSN: 1545-679X.. (Also appears in The Proceedings of ISECON 2003: §2511. ISSN: 1542-7382.)

Liegle, J., Madey, G.: "A Classification of Programming Knowledge and applicable Teaching Strategies," IEEE Transaction on Education, pp. 1-27, resubmitted Sept. 2003 (under review)

Meso, P and J. O. Liegle. "An Exploratory Comparative Assessment of .NET as a Pedagogical Tool for Teaching Object-Oriented Systems Design," In D Colton, M J Payne, N Bhatnagar, and C R Woratschek (Eds.), The Proceedings of ISECON 2002, v 19 (San Antonio): 244a. AITP Foundation for Information Technology Education. ISSN: 1542-7382, 2002.

Morris, M. G., Speier, C., and Hoffer, J., , "An Examination of Procedural And Object -Oriented Systems Analysis Methods: Does Prior Experience Help Or Hinder Performance?" Decision Sciences, Vol. 30 No. 1, 1999, pp. 107-136

Mumford, E. "Researching people problems: Some advice to a student," in Research Methods in Information Systems. E. Mumford. North-Holland, Elsevier Science Publishers B.V., 1985, pp. 315-320.

Oestereich, B., "Developing Software with UML: Object-Oriented Analysis and Design in Practice," Addison Wesley, 2002

Straub, Detmar W., Mark Keil and Walter Brenner, "Testing the Technology Acceptance Model across Cultures: A

Three Country Study, Information & Management," Vol. 31 No. 1, 1997, pp. 1-11

Szajna, B., "Empirical Evaluation of the Revised Technology Acceptance Model," Management Science, Vol. 42 No. 1, 1996, pp.85-92.

Sun Microsystems, "Enterprise JavaBeansTM Components and CORBA Clients: A Developer Guide," Sun Microsystems, Inc., Palo Alto, CA, 2002, http://Java.sun.com/j2se/1.4.2/docs/guide/rmi-iiop/interop.html

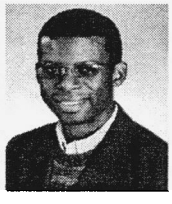
Sun Microsystems, "The NetBeans Platform," Sun Microsystems, Inc., Palo Alto, CA, 2004, http://www.netbeans.org/products/platform/index.html

Thompson, R.L., Higgins, C.A., and Howell, J.M., "Personal Computing: Toward a Conceptual Model of Utilization," MIS Quarterly, Vol. 15 No. 1, 1991, pp. 124-143.

Vessey, I., and Conger, S., "Requirements Specification: Learning Object, Process, and Data Methodologies." Communications of The ACM, Vol. 37 No. 5, 1994, pp.102-113

Wang, M., "E-business application development with Java technology and Oracle: The Fortune Invest Inc. case," Journal of Information Systems Education, Vol. 14 No 3, pp. 293-300.

## AUTHOR BIOGRAPHIES



**Peter Meso** is an Assistant Professor of Information Systems at Georgia State University. His current research deals with the contributions of software engineering in knowledge management, and consequences of information systems in underdeveloped nations. He earned his PhD degree in Information Systems from Kent State University, and holds a Bachelor of Science (Information systems) and an MBA degree from the United States International University - Africa. His published works have appeared in the Communications of the ACM, Journal of Global Information Management, Information Systems Journal, and the Journal of Knowledge Management, among others.

**Jens Liegle** is an Assistant Professor of Information Systems at Georgia State University. His research interests include emergent technologies, user modeling, adaptive hypermedia, and IS education. He received his PhD in Management Information Systems from Kent State University and his MBA from the University of Akron. He has a Diplom Betriebswirt (FH) degree from the Fachhochschule für Wirtschaft, Pforzheim, Germany. His published works have appeared in Journal of Systems and Software, Computers in Human Behavior, Information Systems Education Journal, and the Journal of Asynchronous Learning Networks, among others.


Information Systems & Computing
Academic Professionals


## STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.



ISSN 1055-3096