

AN EMPIRICAL INVESTIGATION FOR DIFFERENT SEQUENCES OF PEDAGOGICAL FRAMEWORK DOCUMENTATION IN RAD

Ho Sin Ban *, Ian Chai **, Tan Chuie Hong *

* Faculty of Information Technology, ** Faculty of Engineering

Multimedia University, 63100 Cyberjaya, Malaysia.

Phone: +603 8312 5408, +603 8312 5379; Fax: +603 8312 5264; email: {sbho, ianchai, chtan}@mmu.edu.my

Abstract

Frameworks are increasingly employed as a useful way to enable object-oriented reuse. However, their size and complexity make understanding how to use them difficult. Previous work concentrated on different ways to document frameworks, but it was unclear which ones actually were better. This paper presents a novel way of investigating the sequences of three philosophies for new framework documentation. Step-by-step documentation is the one traditionally used for software documentation. This philosophy gives clear and sequential steps so that people can follow them and learn how to do the task. Minimalist documentation is random-access so that the reader can proceed in a self-directed manner, and promotes using small cards or pages that can be read in the order the reader desires. Patterns centres upon the idea of giving the solution to a problem in its context. It also has the random-access idea of minimalist documentation, but disagrees with it in that contextual information is not minimal - it usually contains information other than what needs to be done. This investigation discovered some guidelines for effective framework documentation, particularly in a Rapid Application Development (RAD) environment. The results suggest that different documentation sequences are better for different goals.

Key Words

Empirical Software Engineering, Software Maintenance, Documentation, Framework Techniques

1. Motivation

Since the introduction of object oriented frameworks, the classes that frameworks provide have continuously become more sophisticated. From presenting a static frame for texts, applications have evolved to make use of concrete characterizations in reusable parts, such as menus and buttons for the user interface (UI) objects. This particularly contributes to the importance of rapid applications development using frameworks like MacApp, Microsoft Visual Basic (VB), MFC, ACE and implementations of OMG's CORBA [8]. Frameworks

have a number of reuse problems, e.g. those identified by Kirk et al [11]. Frameworks have large amounts of behavioural and structural information. Both designing and learning to use a framework is difficult because much of this information is unfamiliar to developers new to a framework. The real benefits to a framework only come after one knows how to use it.

The best way to learn a framework is to sit with an expert. However, handholding by an expert will not be practical for most framework users. This is a reason why we need good framework documentation to make mass learning possible. To achieve good documentation, we need to find well-established rules for documenting frameworks. The general problem of how to document frameworks is large. Thus, we have decided to tackle new user documentation or tutorials. This is a very important part because once past the new user stage, one often has the familiarity to figure the details out.

Doc. Philosophies	Steps Order	Background Information
Traditional Step by Step	Chronological	Optional
Minimalist Documentation	Random access	Let user figure out
Patterns Documentation	Random access	Give problem context

Table 1. The overview of three documentation (doc.) philosophies

Table 1 summarizes some key points of the three philosophies being evaluated: *traditional step-by-step instructions*, *minimalist documentation*, and *patterns-style documentation*. These are directly competing philosophies in that some of their guidelines contradict each other. See [7], [9] and [10] for more details on these philosophies. Each of these philosophies sounds reasonable from their description. Each is compatible with the idea of testing one's documentation on end-users and making changes based on their reactions. Each is open to the idea of mixing text, examples, and diagrams.

In previous experiments on documentation [7] [9], the subjects were exposed to only one documentation style, and were not exposed to the other existing documentation styles. If the subjects have the opportunity to try other styles, they may rate their documentation preferences differently. Thus, we intend to discover which sequence arrangement of documentation philosophy would work better than the others for documenting frameworks for new users. If we know the right way to structure and

present to users what type of the documentation they should use first, we could have more effective documentation. In this paper, we use the VB framework as the basis of our study on the impact of different documentation sequences in a Rapid Application Development (RAD) environment. Many people recommended implementing a prototype in a RAD tool such as VB before converting the final product into a true object-oriented programming language such as Java. The RAD environment in this experiment makes use of the evolutionary approach. Through the four work tasks in the documentation, the experiment subjects fix mistakes and expand a prototype into the final implementation solution, which is checked via a survey at the end.

We chose Visual Basic (VB) due to its relative availability and convenience. This investigation was carried out during tutorials of software engineering course in February 2004, and VB was the RAD tool taught in this course. Furthermore, Raadt [17] showed in his census that VB is the most commonly used language to introduce the RAD world, rather than other popular RAD tools such as Delphi and JBase. After giving an overview of the related work in section 2, section 3 presents a brief overview of the empirical investigation. Section 4 describes the results of data and analysis of the investigation. Section 5 discusses threats to validity. Finally, section 6 concludes the paper and raises questions for future research.

2. Related Work

A number of empirical experiments for documentation have been proposed over time. However, they mostly focus on different ways to document frameworks, but not the impact in exposing different sequences of documentation [1] [13]. Large RAD software systems are almost always expressed in free text documentation. The informal way of expressing user manuals created different ways to document frameworks in the RAD environment.

The assessment on Pattern Comment Lines (PCL) presented by Prechelt et al [14] suggests that design patterns always be documented explicitly in program source code. PCL in a program may considerably reduce the time required for a program change or improve the quality of the change. Another controlled experiment compares design using patterns versus simpler solutions, which are alternative straight-forward designs in the context of program maintenance [15]. Prechelt et al. conclude that in most of the maintenance tasks, design patterns help to reduce maintenance time. It is wise to choose the flexibility provided by the design pattern as unexpected new requirements often appear.

Douglas Kirk et al in their 2002 report [11] identify the major problems of reuse and the impact of current documentation techniques on these problems. They presented a number of suggestions for improving developer support during reuse. Table 2 presents a summary of the identified problem categories and the

documentation techniques that seem to best address them. Techniques that support more than one category, e.g. design patterns, often provide differing levels of support to each category. Pattern languages stand out as the only technique that is at least partially effective across all problem categories. From a pattern language, developers gain knowledge or the range of modification options available to them.

Problem Category	Documentation Techniques
Understanding Functionality	Design patterns (DP.), Examples, JavaDoc, Pattern language, Source code
Understanding Interactions	DP., Pattern language, Source code
Mapping Modifications with Expectations	Examples, JavaDoc, Pattern language
Understanding Architecture	DP., Examples, Pattern language

Table 2. Framework Problems vs. Documentation Techniques

The two most related experiments so far appears to be [7] and [9], which empirically investigated three documentation philosophies for new users in Habanero and VB framework respectively. These reports presented experimental results on the respective framework, to determine which documentation styles are better for various goals. However, we are still missing an investigation on the impact of exposing different sequences of documentation to the new users. The experimental design and results presented in this paper are carried out to address this need. This paper complements to the earlier findings in Habanero framework [7] and VB framework [9]. The sample size of data in this investigation involves more than 80 subjects.

3. An overview of our experiment

The study was conducted to investigate two separate effects of documentation sequences and their interaction. Does a particular documentation sequence increase the effectiveness in learning a new framework? We now give a short description of our experiment design and conduct. Further details, including the original experiment documents, such as the programs, questionnaires, work tasks, and documentation materials used in this experiment are available from <http://pesona.mmu.edu.my/~sbho/Writing/>.

3.1 Hypotheses

Standard significance testing was used to clearly specify the effects of the two documentation sequences in a Rapid Application Development (RAD) environment. For the sake of brevity, we have supplied only two null hypotheses, which are stated as follows.

E1H₀ - There is no difference between documents sequence, in terms of completion time, ease of composition, accuracy, workings and numbers of difficulties faced regardless to any sequences of the documentation techniques applied.

E2H₀ - Documents sequence does not affect the subjects' preference in documentation type.

The interpretations of this experiment are derived from the rejection or non-rejection of these hypotheses for each expectation.

3.2 Subjects and Groups

We divided the subjects into two groups according to the subjects' tutorial sections. Each stage takes approximately one hour of the tutorial session. The subjects are all undergraduates taking a software engineering course at Multimedia University, Cyberjaya, Malaysia. The participants of our study were mixture students of two faculties, i.e. the Faculty of Information Technology (FIT) and the Faculty of Management (FOM). On average, they had been studying at Multimedia University for 2.8 years.

During the lectures, the students were taught basic software engineering principles as well as being introduced to the rapid application development technique. The lectures were supplemented by practical tutorial sessions where the students had the opportunity to make use of what they had learned through completion of various software development exercises using the two different sequences of on-line documentation. Our data were collected during the first three tutorials in introducing the students to Visual Basic framework. The 86 students knew that analysis would be performed on this data, but were unaware of the experimental hypotheses that were being tested.

3.3 Setup of our Experiment

To be able to test the hypotheses of our experiment, two different groups of the RAD system documentation were required. These system documents were designed according to the sequence presented in Table 3. The two sets of on-line programming documents were developed according to the different documentation philosophies.

{Stage i} Documentation Sequence	Group A	Group B
{Stage 1} Simple Payroll Program	Patterns	Step By Step
{Stage 2} Enhanced Payroll Program	Minimalist	Patterns
{Stage 3} Menu & Text Program	Step By Step	Minimalist
Total subjects undergone this sequence	56	30

Table 3. Sequence of documentation treated to experiment subjects

The technical setup for this experiment includes developing the documentation on a workstation using an html editor such as Dreamweaver. These documents were subsequently uploaded to a web server so that the users can access the on-line documentation. Before our experiment began, a digital clock was displayed on the projector for the subjects' common reference.

Our method of observation consisted of a survey with two sections. The first section requested the subjects to record their completion time after each task, while the

second section include 8 questions of various types including multiple-choice, ratings and free-form questions. Questions raised by more than two subjects were recorded as qualitative findings.

3.4 Experimental materials

Our experimental materials used two sets of documentation. The full documentation consists of the *Quick Start Example (Simple Payroll Program)*, *Basic Topics (Enhanced Payroll Program)* and *Intermediate Topics (Menu and Text Program)*. Figure 1 shows part of an example in the *Intermediate Topics*, which was originally the traditional step by step instruction. This base document was further cut into smaller pieces to build the minimalist documentation. The background information section is added to the top of each piece in order to form the patterns documentation.

```

Writing code for Pop-up Menu
3. For this form, type in the following code. You may
copy and paste only the bold faced code to the
respective object and procedure.

Object: '(General)' - Procedure '(Declaration)'
' Demonstrating pop up menus
Option Explicit ' General Declaration

Object: 'Form' - Procedure 'MouseUp'
Private Sub Form_MouseUp(Button As Integer,
Shift As Integer, X As Single, Y As Single)
' When right button is clicked display Pop Up menu
If Button = vbRightButton Then
Call PopupMenu(mnuPopUp)
End If
End Sub

```

Figure 1. Example of the documentation fragment

This paper focuses on the *Intermediate Topics* where the subjects were in the third stage after they had gone through the different sequences of documentation. To give a picture of the documentation relative total length, the documentation size is measured in kilobytes, as proposed by Beizer [2]. In this manner, we can quantitatively characterize the documents. Table 4 gives quantitative information about the character of the documents used in the *Intermediate Topics*.

Quantitative Characterization	Group A	Group B
1. Relative total length (in kilobytes)	101 KB	150 KB
2. Information that is relatively missing	Overview list of work tasks	Background information
3. Number of document files	3 files	7 files
4. Total sections in the documentation	6 sections	8 sections
5. Total paragraphs in the documentation	15 paragraphs	19 paragraphs

Table 4. Characterize documentation quantitatively for this experiment

Since the documentation contains multiple files and figures, we adjusted each document sets in such that some

subjects may be starved of information, such as the list of work tasks in minimalist documentation. Meanwhile, some subjects may find that they were flooded with irrelevant information, such as background information in the traditional step by step documentation.

3.5 Experimental Design

Our experimental design uses seven dependent variables and one independent variable (factor). The independent variable is the documentation group; while the dependent variables are the completion time, number of difficulties faced and semi completion time, documentation preference, accuracy, workings and comprehension (understanding of the exercise).

- **“Documentation Sequence”**: We use two different documentation sequences, each with a similar purpose: to complete a ‘Menu & Text’ program using a Rapid Application Development environment.
- **Dependent variable “completion time”**: The time taken to finish the entire exercise.
- **Dependent variable “semi completion time”**: Time taken for the subjects to do their first compilation.
- **Dependent variable “number of difficulties faced”**: Instead of giving all the detailed steps, some parts of the documentation let the learner interact with the system. The subjects were to record and accumulate numbers of problems they encountered.
- **Dependent variable “comprehension”**: The subjects had to identify the method, procedure, line of the code and constants that perform the given task. There were four questions to test their understanding of the code.
- **Dependent variable “accuracy”**: This indicates whether the participant’s solutions fulfilled the requirements of the task or not.
- **Dependent variable “workings”**: This is to test how well the subjects are able to follow the instruction in assigning default settings to the menu items.
- **Dependent variable “documentation preference”**: This denotes the preference of the subjects towards a particular documentation style after going through the three different documentation styles.

4. Data and Analysis of the Results

We analyzed the data to see if one of the documentation sets let the subjects compile (**SEMICOMPLETION**) and finish the fastest (**COMPLETION**) with the number of difficulties recorded by the subject at these intervals (**NUMBER OF DIFFICULTIES**), as well as understand the most (**COMPREHENSION**). We also checked for the correlation to the self-tested scores of how well the documentation taught them to rapidly build an application (**ACCURACY**) and test scores on how well their knowledge in the inner workings of Visual Basic

(**WORKINGS**). Since we did not want to rely on the assumption of normal distribution, we tested for the normality of the dependent variables. From the normality test, we discovered that all dependent variables except **NUMBER OF DIFFICULTIES** are normally distributed for each subject groups. As such, for this dependent variable, medians will be used as the expected values, rather than the means, as shown in Table 5.

Category (Dependent variable)	Mean		Std. dev.	
	Group A	Group B	Group A	Group B
1. Semicompletion (hh:mm)	0.08	0.15	0.02	0.09
2. Completion (hh:mm)	0.51	0.57	0.10	0.14
3. Comprehension (scale: 0-4)	3.09	3.00	0.88	0.87
4. Accuracy (scale: 0, 1, 2)	1.73	1.37	0.59	0.77
5. Workings (scale: 0, 1, 2)	1.82	0.97	0.47	0.93

Category (Dependent variable)	Median		Std. dev.	
	Group A	Group B	Group A	Group B
6. Number of difficulties	1.00	2.00	4.04	2.95

Table 5. The means, median & standard deviations of all categories

Table 6 presents a summary of results of the statistical tests for the first five dependent variables. These values were obtained via tests of between-subjects effects using Univariate Analysis of Variance (ANOVA) [12]. Post hoc tests are not performed for documentation type because the treatments in this experiment are fewer than three groups. From these results, we observed that four out of five independent variables are significant. The symbol ‘*’ indicates there is evidence of differences with p-value < 0.05 significance level.

Category:	Semi.	Compl.	Compr.	Accur.	Work.
Group A vs. Group B	* 0.045	* 0.000	0.654	* 0.020	* 0.000

Table 6. Univariate ANOVA results for Group A versus Group B

In terms of **SEMICOMPLETION** and **COMPLETION**, when looking for the standard significance level of 0.05 (i.e. 95% probability) in Table 6, we see that the treatments make a significant difference. The subjects in group A completed their first compilation and complete the experiment faster than the ones using the other documentation sequence. Therefore, we conclude that there is significant difference between the two documentation sequences as to how long it takes for the subjects to complete the experiment.

As for **COMPREHENSION**, there was no significant difference between how well the subjects understand the materials. This might be because the students still able to understand the code in the end, irrespective to the document sequences. Their learning might reach a maturation effect [5] after going through the three stages of documentation. The subjects learned enough from the prior stages to bias their performance in the third stage of the experimental run.

Regarding **ACCURACY** and **WORKINGS**, the subjects in group A exhibited significantly higher accuracy scores than group B documentation sequence. In

this particular sample, the subjects in group A also on average rated workings higher than those who used group B sequence. Interestingly, there are significant differences in these two factors, which mean that the $E1H_0$ hypothesis in section 3.1 is rejected. These rejections show that the two documentation sequences are not the same in teaching the subjects about the inner workings of Visual Basic, as well as completing the work tasks correctly.

Group	Sample	Sum of	Removal of
	size, n	Ranks	invalid cases
Group A (Pat-Min-SBS.)	55	2225.50	1 not valid case
Group B (SBS-Pat-Min.)	27	1177.50	3 not valid cases
Test Statistics	Mann-Whitney U	Wilcoxon W	Asymptotic Sig. (2-tailed)
	685.500	2225.500	0.563

Table 7. Mann-Whitney test results on the Number of Difficulties

Since **NUMBER OF DIFFICULTIES** was not normally distributed with comparison of two groups, we use the Mann-Whitney test [3]. With the two-sided asymptotic significant value in Table 7 more than 0.05, the number of difficulties faced by the subjects has no significant difference between the two groups. With the minimalist documentation in the final stage of Group B, users need not read background information that may not be relevant in reducing their number of difficulties. The traditional step by step documentation in the final stage of Group A provides this background information.

Documentation (doc.) type	Mean		Std. dev.	
	Group A	Group B	Group A	Group B
1. Patterns Documentation	2.21	1.87	0.77	0.95
2. Minimalist Documentation	2.21	1.65	0.70	1.08
3. Traditional Step by Step	1.58	1.68	0.81	1.02

Table 8. Preference of documentation type, according to group

In this study, the **DOCUMENTATION PREFERENCE** of the subjects is generally categorized into scale '1' for most preferred style, '2' for average and '3' for least preferred style. From Table 8, it is obvious that the subjects prefer the most recent documentation presented to them. This violates the dependency requirement since both the groups are influenced by the documentation type presented to them in the third stage. Hence we performed independent sample test on each of the documentation type to seek for further evidence.

Documentation Type:	t value	df	p-value (2 tailed)
1. Patterns documentation	1.788	84	0.077
2. Minimalist doc.	-0.477	49.074	0.635
3. Step by Step Doc.	2.594	42.479	* 0.013

Table 9. Independent sample test results on documentation preference

Table 9 reports results for two-tailed paired t-test on preference among the three documentation types. The statistical significance indicates a relationship exists, irrespective to the means in Table 8 [12]. At the standard

significance level of 0.05, there is a statistical difference between the two groups in the rating of traditional step by step documentation. As such, this poses the rejection of $E1H_0$ hypothesis in section 3.1 in particular for step by step documentation. Subjects in Group B are especially of interest in disliking step by step documentation, since they were using minimalist documentation at the final stage of the sequence. They may prefer to the availability of navigation link to the particular work task. In this situation, the approach of breaking the documentation into pieces attracts subjects in Group B.

Another possible reason is looking at the order of step by step documentation in the both group sequences. From the Table 3 in section 3.3, step by step documentation was placed at the first stage in the sequence of Group A, while it appeared as the third stage in Group B. This could contribute to such significant difference in the preference for step by step documentation.

5. Threats to Validity

This section discusses the various threats to validity of our investigation. It is important to point out that weaknesses imposed by these threats of one empirical study can be addressed by the strengths of another.

5.1 Construct validity

Construct validity is the degree to which the independent and dependent variables accurately measure the concepts they purport to measure. To converge towards credible answers, a series of experiments and studies with different designs are usually necessary [4]. One of the possible threats has been identified is **COMPREHENSION** (understandability), which is a difficult concept to measure. In a single controlled experiment, it is unlikely that different dimensions of a concept can be captured. The researchers must focus on what can be realistically achieved. Furthermore, since it is more reliable to use several different measures to capture a concept, we defined four measures for the concept of comprehension. However, comparing the significance level of comprehension with the other four dependent variables, comprehension shows there was no significant difference between the groups. Thus, we cannot be sure whether the 4 measures used to capture comprehension is adequate, so we can only state that we are unsure about the seriousness of this threat.

5.2 Internal validity

Internal validity is the degree to which conclusions can be drawn about the casual effect of independent variables on the dependent variable. The documentation preference effect may result from the most recent material employed. The threat to this study was that possible differences between the two documentation sequences. A serious attempt was made to control for such a threat by ensuring

that the same subjects to go through all the three documentation styles and the tasks in each stage required similar amounts of time to complete.

5.3 External validity

External validity is the degree to which the results of the research can be generalised to the population under study and other research settings. We have identified two possible external threats. Firstly, the subjects who participated in this experiment may be not the full representative of software learners. Due to the constraint in tutorial sections arrangement, all the subjects in Group A were FIT undergraduates, while the majority of Group B subjects were from FOM. FOM students may be weaker in their programming skills.

Another threat was the sequence settings. Three different documentation styles actually imply six different sequences in such that each style only appears once. Rather than conducting all the six sequences, we investigated two of them to analytically show that different sequences affect learning effectiveness. Laboratory settings such as the one we used allow an investigation of RAD tool at a lower cost than the full scale studies. An examination of our two sequences for constructing framework documentation reveals a good chance of discovering important and interesting findings.

6. Conclusions and Future Research

We have presented an evolutionary approach for investigating framework documentation in Rapid Application Development. The results not only point out that it is useful to show background information in the earliest stage, but also indicate the difference in organizing documentation styles. Just as good software requires testing, we found that good documentation requires testing. A good piece of documentation goes through much iteration of testing and revision.

These experiments answered some questions, but also raised many more. If we increase the resolution of the scale of COMPREHENSION, say, from four to nine, will our observation still hold? Perhaps, with many more questions in the exercise, there will be a more accurate reckoning of how much knowledge the users gain from the different documentation sequences. Another area that future experiments can explore is which kind of diagrams, e.g. UML [16] and Use Case Maps [6], work better in helping application programmers understand the framework in order to use it? Finally, we are striving to gain more empirical experience from other frameworks.

References

- [1] G. Antoniol, G. Canfora, G. Casazza, Recovering Traceability Links between Code and Documentation, *IEEE Trans. on Software Eng.*, 28(10), 2002, 970–983.
- [2] B. Beizer, Software is Different, *Proc. of 14th Int'l Internet and Software Quality Week Conf.*, San Francisco, CA: Software Research Inc. June 2001. [Available online: www.soft.com/News/QTN-Online]
- [3] A. G. Bluman, *Elementary Statistics: A Step by Step Approach, 5th Edition* (New York, NY: McGraw-Hill International Edition, 2004, 640–645).
- [4] L. C. Briand, Software Documentation: How Much is Enough? *IEEE Proc. 7th European Conf. on Software Maintenance and Reengineering (CSMR'03)*, Benevento, Italy, 26 - 28 March 2001, 13–15.
- [5] L. C. Briand, C. Bunse, J. W. Daly, An Experimental Evaluation of Quality Guidelines on the Maintainability of Object-Oriented Design Documents, *ACM Proc. 7th Workshop on Empirical Studies of programmers*, Alexandria, Virginia, USA, 1997, 1–19.
- [6] R. J. A. Buhr, Use Case Maps as Architectural Entities for Complex Systems, *IEEE Trans. on Software Eng.*, 24(12), Dec. 1998, 1131–1155.
- [7] I. Chai, *Pedagogical Framework Documentation: How to document Object-Oriented Frameworks: An Empirical Study*, (Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2000). [Available online: www.cs.uiuc.edu/Dienst/UI/2.0/Describe/ncstrl.uiuc_cs/UIUCDCS-R-99-2077]
- [8] M. Fayad, R. Johnson, & D. C. Schmidt, editors. *Domain-Specific Application Frameworks: Problems and Perspectives* (NY: John Wiley & Sons 1999).
- [9] S. B. Ho, I. Chai, & C. H. Tan, Modeling and Simulations of Pedagogical Framework Documentation: An Empirical Study for Prototyping in a Rapid Application Development Environment, *AIUB Int'l Journal of Business and Economics*, 2(1), August 2004.
- [10] R. Johnson, Documenting Frameworks Using Patterns, *Proc. ACM Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'92)*, ACM Press, 1992, 63–76.
- [11] D. Kirk, M. Roper, & M. Wood, Defining the Problems of Framework Reuse, *Proc. 26th IEEE Int'l Computer Software and Applications Conf. (COMPSAC'02)*, Oxford, England, U.K., 26 – 29th Aug. 2002, 623–626.
- [12] J. Neter, M. H. Kutner, C. J. Nachtsheim, W. Wasserman, *Applied Linear Statistical Models*, 4th Edition, Boston, Mass.: McGraw Hill, 1996, 776–780.
- [13] M. D. Penta, S. Gradara, G. Antoniol, Traceability Recovery in RAD Software Systems, *IEEE Proc. 10th Int'l Workshop on Program Comprehension, June 2002*, 207–216.
- [14] L. Prechelt, B. Unger, M. Philippsen, & W. F. Tichy, Two Controlled Experiments Assessing the Usefulness of Design Pattern Information During Program Maintenance, *IEEE Trans. on Software Eng.*, 28(6), June 2002, 595–606.
- [15] L. Prechelt, B. Unger, W. F. Tichy, P. Brössler, & L. G. Votta, A Controlled Experiment in Maintenance Comparing Design Patterns to Simpler Solutions, *IEEE Trans. on Software Eng.*, 27(12), 2001, 1134–1144.
- [16] M. Priestley, *Practical Object-Oriented Design with UML*, 2nd Edition (New York, NY: McGrawHill, 2004).
- [17] M. Raadt, R. Watson, & M. Toleman, Introductory Programming: What's Happening Today and Will There Be Any Students to Teach Tomorrow, *Proc. Int'l Sixth Conf. on Australian Computing Edu.*, Dunedin, NZ, Jan. 2004, 277–282.