Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2004 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2004

A Longitudinal Study of Information Systems Developers' Understanding of Software Development Concepts During a Transition from Structured to Object-Oriented Development

Teresa Shaft
University of Oklahoma

Leslie Albert
University of Oklahoma

Jon Jasperson
University of Oklahoma

A Longitudinal Study of Information Systems Developers' Understanding of Software Development Concepts During a Transition from Structured to Object-Oriented Development

Teresa M. Shaft
Michael F. Price College of Business
University of Oklahoma
tshaft@ou.edu

Leslie Jordan Albert
Michael F. Price College of Business
University of Oklahoma
leslliealbert@ou.edu

Jon (Sean) Jasperson
Michael F. Price College of Business
University of Oklahoma
jjasperson@ou.edu

ABSTRACT

Object-oriented development (OOD) has attracted great interest in MIS due to the belief that OOD makes it easier to develop and maintain software and achieve the benefits of software reuse. However**,** the transition from a structured development (SD) to an OOD methodology may be especially challenging and some argue that SD knowledge may interfere with the ability to learn OOD.

We assessed a team of developers' understanding of SD and OOD concepts via a survey administered at the beginning of a development project and one year later. Developers increased their understanding of some OOD concepts, but these were associated with increases in understanding of SD concepts. This challenges the idea that developers must "unlearn" SD concepts to adopt OOD. We also found that developer's mental models were highly consistent with regard to SD and most OOD concepts at the beginning and end of the project.

Keywords

Object-oriented development, structured development, shared mental models, longitudinal case study

INTRODUCTION

For over twenty-five years, structured development (SD) has been the dominant methodology for building information systems (IS). More recently, object-oriented development (OOD) has attracted increasing interest among IS academics and practitioners. This interest arises due to the belief that adopting an OOD approach makes it easier to develop and maintain software and achieves the benefits of software reuse (Alvarez, Diaz, Llopis, Pimentel and Troya, 2003). In addition, OOD is considered to be more consistent with software developers' cognitive processes (Rosson and Alpert, 1990). More recently, it has been argued that OOD may not be particularly consistent with human cognitive processes (Hatton, 1998). Regardless, despite the touted benefits of OOD, it has not been widely adopted (Sumit, Nerur and Malhapatra, 2001).

Regardless of whether or not OOD is more consistent with human cognitive processes, the majority of professional software developers have experience with SD. Because these developers typically possess extensive business knowledge that firms wish to retain (Shaft and Vessey, 1995), if a firm wishes to adopt OOD, they frequently will transition experienced developers from SD to OOD rather than hire new OOD developers. As SD and OOD are different methodologies, transitioning from SD to OOD requires systems developers to learn new techniques to model system requirements, new languages to write computer programs, and new software tools. While the change to any new development paradigm can be difficult (Fayad, Tsai and Fulghum, 1996), the differences in the cognitive processes required by the two methodologies may make the transition to an OOD environment especially challenging (Briand, Arisholm, Counsell, Houdek and Thevenod-Fosse, 1999).

While it seems logical to assume that any prior programming experience would ease the transition to OOD, some prior research suggests that SD knowledge may interfere with the ability to learn OOD (Nelson, Armstrong and Ghods, 2002). Therefore, experienced developers moving to OOD may be hindered by their SD knowledge. We focus on the cognitive aspects of the transition to OOD by studying the changes in a team of developers' knowledge of concepts considered fundamental to each methodology. The following research questions guided our research: (1) What effect, if any, does prior knowledge of SD concepts have upon the learning of OOD concepts? (2) Is there a difference in the rate OOD concepts are acquired? (3) Do team members' mental models (MM) become more or less consistent as team members learn OOD concepts?

THEORETICAL DEVELOPMENT

The transition to OOD requires a systems developer to learn a new software development methdology. One way an individual learns is to match novel concepts to concepts or schema he/she already understands. When the learner makes a good match between the new and existing concepts, prior knowledge aids in the process of learning the new concepts. When a poor match is made, prior knowledge makes learning new concepts more difficult than if no prior knowledge existed. This leads to a cognitive block referred to as proactive interference. Prior researchers argue that proactive interference is why developers must "unlearn" SD concepts in order to "learn" OOD concepts and why the transition is so difficult (Nelson, et al., 2002).

Armstrong (2001) identified five OOD concepts: creating objects, object characteristics, interaction, nesting, and system (ranging from easiest to most difficult to learn), and two SD concepts, function and flow. Creating objects refers to those concepts to which individuals are first introduced. Object characteristic refers to the classification of objects and their function. Interaction deals with the relationships between objects and sets of objects. Nesting refers to the complexity of interaction relationships including issues of inheritance. Lastly, the system construct is concerned with the way objects function within the whole of the program. The SD concepts (function and flow) represent the physical structure of sub-programs and the ways in which they related to create a whole.

When organizations initiate transition projects, managers expect that the software developers acquire knowledge of OOD. However, we questioned whether developers' existing knowledge of SD concepts remained stable, declined, or increased. If OOD and SD are distinct, understanding of SD and OOD are unrelated, then SD knowledge would remain stable as developers gain OOD knowledge (H1). However, if SD knowledge causes proactive interference with the acquisition of OOD knowledge (Nelson, Armstrong and Ghods 2002), SD knowledge may decline. If, however, SD knowledge transfers to OOD, then it is possible that developers may increase their knowledge of SD even in the context of learning OOD. Existing research has emphasized issues of negative transfer and proactive interference while ignoring the potential for positive transfer between SD and OOD. Based on the above arguments, we state the following null hypotheses:

> H1: No association between changes in a developer's knowledge of OOD concepts and changes in knowledge of SD concepts.

In addition, which OOD concepts changed over the course of the project? The five OOD concepts are considered to have differing levels of difficulty (Armstrong, 2001). However, longitudinal studies of the development of OOD expertise have not been conducted, so it is an empirical question if developers' acquire knowledge of various OOD concepts at different rates.

> H2: At the end of the project, more difficult to understand OOD concepts will be less developed compared to easier to understand OOD concepts.

H2 concerns the changes that individual developers experience. However, it is also worth investigating if some developers gain knowledge of OOD concepts more easily than others. If we find that certain developers gained knowledge of OOD concepts that other developers found difficult, we can try to determine what allowed them to develop their knowledge.

> H3: At the end of the project, some developers will gain a better understanding of more difficult to understand concepts than other developers.

In addition to assessing changes in developers' understanding of OOD concepts, we investigated the consistency of their mental models (MMs) (Espinosa, 2001). One would expect more similarity among team members' MMs at the end of the project than at the beginning. Further, as noted above, it is an empirical question if SD knowledge declines, remains stable, or

increases in conjunction with OOD knowledge. Therefore, we wish to examine if the team members' MMs are more or less consistent with regard to the OOD and SD concepts. Finally, one would expect that team members' MMs of difficult-to-understand concepts to be less consistent than for easier-to-understand concepts. Generally, consistency among team members' MMs leads to higher levels of performance (Mathieu et al., 2000). Therefore, it is we wish to identify individuals whose understanding was less consistent with other team members and attempt to identify factors that may have lead to the inconsistency. If certain factors inhibited an individual in creating a MM consistent with that of other team members, we may be able to identify ways that organizations and teams can help future teams.

H4a: The consistency between the development team's mental models with regard to SD concepts should be unchanged at the end of the project compared to the beginning.
H4b: The consistency between the development team's mental models with regard to the easier to understand OOD concepts should be higher at the end of the project compared to the beginning.
H4c: The consistency between the development team's mental models with regard to difficult to understand OOD concepts should be less consistent than easier to understand OOD concepts.

DATA COLLECTION

We used The Software Development and Maintenance Approach Instrument (Armstrong, 2001). This 90-question survey has a 5-point Likert-type scale with the following responses: strongly disagree, disagree, neither agree or disagree, agree and strongly agree, plus an "I don't know" option. Armstrong (2001) developed this instrument through a process of interviewing experts, coding the interviews and causal mapping. The instrument was pilot tested then used to assess the differences in the knowledge of Structured with OO developers.

We studied a team of developers employed by an international petroleum company who were engaged in the firm's pilot OOD project. The system they developed was a real-time reporting system, appropriate for OOD (Sumit et al., 2001). A modified RUP methodology was employed. Modifications to RUP were based on the advice of a OOD consultancy firm that provided on-site training. To understand the transition, we administered the survey to assess developers' knowledge at the beginning (shortly after the development team completed five weeks of OOD training courses) and end (one year later) of the project. By the time of the second survey, the new system had been in use for approximately three months and the development team was involved in its on-going maintenance and revision.

In addition to the survey instrument, we also asked the developers to complete a systems analysis task to assess reliance on SD v. OOD techniques. The task presented a short description of a business problem and a proposal to develop an information system to address the business situation. We asked developers to spend no more than 30 minutes describing their approach to the analysis and design of the proposed system. Developers completed this task during both waves of data collection.

We compare developers' responses to the two administrations of the survey and task to assess the extent to which they had shifted from a SD to an OOD approach during the course of the development project.

ANALYSIS & RESULTS

Participants included three women and five men with an average of 8.3 years of professional development experience (including one OOD expert and the project manager); average experience of 7.6 years when the OOD expert and project manager are excluded. Seven of the respondents had Bachelor's degrees, one also had a master's degree and one participant completed her MBA during the course of the project. Participants were assigned exclusively in the OOD project during the study.

To analyze the survey data, for each software developer, scores were developed for the five OOD and two SD concepts identified via the factor analysis presented in Armstrong (2001). We rely on her factors because our sample size precludes us from confirming the factor structure. Due to the small sample size, α is set to .1 for all analyses.

Hypothesis 1: Relationship between changes in understanding of OOD and SD concepts

The first hypothesis is concerned with the relationship between individual developer's knowledge of OOD and SD concepts. Correlation analyses between the changes scores for each concept were conducted to assess the hypotheses. We created change scores by subtracting a developer's score for each concept on the first administration of the instrument from the score on the second administration of the instrument.

The correlations and p-values are presented in Table 1. We can see that changes in understanding of functionality (an SD concept) are associated significantly with changes in four of the five OOD concepts. However, the flow concept (the final SD concept) was not significantly associated with changes in any OOD concept. The hypothesis has only marginal support: the findings regarding the flow concept support the hypothesis while those regarding the functionality concept would lead us to reject the hypothesis of no association between SD and OOD concepts. One conclusion is that not all OOD concepts are distinct from all SD concepts. Instead, some concepts may be essential to software development regardless of paradigm. Additionally, correlations between all OOD and SD concepts were positive (although not necessarily significant) therefore, it does not appear that developers must "unlearn" SD concepts to "learn" OOD concepts.

	1	2	3	4	5	6	7
Object-Oriented Concepts							
1. Creating Objects	1.00	0.96[a]	0.91[b]	0.74[c]	0.69[b]	0.75[c]	0.45
2. Object Characteristics		1.00	0.77[c]	0.66[d]	0.55	0.75[c]	0.36
3. Interaction Between Objects			1.00	0.61	0.78[c]	0.74[c]	0.6
4. Nesting Of Objects				1.00	0.76[c]	0.51	0.23
5. System Level Object Issues					1.00	0.81[c]	0.57
Structured Development Concepts							
6. Functionality						1.00	0.67[d]
7. Flow							1.00

[a]p-value < 0.001; [b]p-value < 0.01; [c]p-value < 0.05; [d]p-value < 0.10

Table 1. Correlations Between Change Scores

Finally, because the correlations between the concepts were typically high, one might question the discriminant validity of the instrument. Therefore, we ran the correlations on the initial and final administrations of the survey separately (see Tables 2 and 3). The correlations between concepts on the initial administration of the survey are all significant. However, the analysis of the final administration of the instrument reveals very few significant correlations, which provides some evidence of discriminant validity. It appears that at the outset of the software development project, the concepts (OOD or SD) were less distinct than at the end. At the time of the first administration of the survey, the software developers had completed coursework in OOD, but had not applied their knowledge to a project. The separate correlation analyses of the two administrations of the survey data indicate that at the end of the project developers had more distinct knowledge of the concepts.

	1	2	3	4	5	6	7
Object-Oriented Concepts							
1. Creating Objects	1.00	0.95[d]	0.96[d]	0.80[b]	0.73[b]	0.80[b]	0.95[d]
2. Object Characteristics		1.00	0.88[b]	0.81[c]	0.75[c]	0.79[c]	0.85[b]
3. Interaction Between Objects			1.00	0.81[c]	0.78[c]	0.84[b]	0.96[a]
4. Nesting Of Objects				1.00	0.90[b]	0.76[c]	0.72[c]
5. System Level Object Issues					1.00	0.90[b]	0.72[c]
Structured Development Concepts							
6. Functionality						1.00	0.87[b]
7. Flow							1.00

[a]p-value < 0.001; [b]p-value < 0.01; [c]p-value < 0.05; [d]p-value < 0.10

Table 2. Correlations Between Scores on Initial Administration of Survey

	1	2	3	4	5	6	7
Object-Oriented Concepts							
1. Creating Objects	1.00	0.88[b]	0.34	0.40	0.66[d]	0.32	-0.22
2. Object Characteristics		1.00	0.13	0.59	0.59	.022	-0.10
3. Interaction Between Objects			1.00	0.17	0.74[c]	0.57	0.36
4. Nesting Of Objects				1.00	0.68[d]	-0.23	0.20
5. System Level Object Issues					1.00	0.17	0.33
Structured Development Concepts							
6. Functionality						1.00	0.00
7. Flow							1.00

[a]p-value < 0.001; [b]p-value < 0.01; [c]p-value < 0.05; [d]p-value < 0.10

Table 3. Correlations on Scores from Final Administration of Survey

Hypothesis 2: Changes in Understanding of OOD and SD concepts for Individual Developers

Hypothesis 2 is concerned with changes in individual developer's knowledge of OOD and SD concepts. We anticipated that developers would experience less change in understanding of the more difficult to understand concepts (nesting and system level object issues). We conducted paired t-tests using developers' responses to the initial and final administrations of the survey. A mean (or t-value) of zero indicates no change between the administrations, a value less than zero indicates that the developer increased their understanding of the concept (see Table 4).

	Mean (pre – post)	**t-value**
Object-Oriented Concepts		
Creating Objects	-.79	- 1.40
Object Characteristics	-.42	- 0.84
Interaction Between Objects	-.57	- 1.04
Nesting	-1.10	- 2.86[c]
System Level Object Issues	-.79	- 2.27[d]
Structured Development Concepts		
Functionality	-.42	- 0.85
Flow	-.14	- .073

df=6

[a]p-value < 0.001; [b]p-value < 0.01; [c]p-value < 0.05; [d]p-value < 0.10

Table 4. Paired t-test of Developers' Responses to the First and Second Administrations of the Survey

The OOD concepts are ordered from easy to more difficult to understand (Armstrong, 2001). The results are inconsistent with the hypothesis. Only the most difficult to understand concepts reflect significant changes in understanding. A possible explanation for the lack of change in the more easily understood concepts is that the first administration of the survey occurred at the outset of the systems development process, but after the developers completed their OOD classroom training. It is possible that the formal training provided developers with a sufficient understanding of the three more easily understood

concepts prior to development. However, if this is the case, this calls into question the belief that OOD is an extremely difficult transition for experienced developers (Briand et al., 1999).

Hypothesis 3: Changes in Understanding of OOD and SD concepts Between Developers

Hypothesis 3 focuses on changes in understanding of the concepts between developers. Two categorizations were created based on an analysis of developers' solution approach to the OOD tasks collected at the same time as the survey data. The first classification created two groups by identifying those making the greatest and least change from SD to OOD during the course of the study (the developer who was an OOD expert was excluded as he would not be expected to change his approach). The second classification was based on greater and lesser use of OOD at the end of the study (the OOD expert is included in this analysis).

Independent (between groups) t-tests were conducted with the change in understanding for each construct serving as the dependent variables (the change scores used to analyze H1). We present the results comparing the group that demonstrated the most change (four developers) to those that demonstrated the least change in solution approach (three developers) in Table 5 (a pooled variance is used when the test for equality of variances is significant for a concept):

	Greater Change to OO	Least Change to OO	t-value
Object-Oriented Concepts			
Creating Objects	-.33	-1.40	0.79
Object Characteristics	.04	-1.03	0.91
Interaction Between Objects	-.22	-1.04	0.70
Nesting	-.61	-1.76	1.68
System Level Object Issues	-.29	-1.47	2.08[d]
Structured Development Constructs			
Functionality	.33	-1.41	2.31[d]
Flow	0	-.32	0.82

df = 6
[a]p-value < 0.001; [b]p-value < 0.01; [c]p-value < 0.05; [d]p-value < 0.10

Table 5. Differences in Changes of Understanding Between Groups greater and lesser change to an OOD

In general, there is little difference in the changes in understanding of the concepts between the two groups. The group that demonstrated a greater shift to OOD only differs from the other group with regard to the system level object issues and functionality concepts. Note that functionality is a SD concept.

We conducted a parallel analysis based on the other grouping (most v. least OOD approach in the final solution), presented in Table 6.

Again, understanding of most of the concepts does not vary between groups. The group that demonstrated a more OOD solution approach at the end of the study (including the OO expert) appears to also have experienced greater changes in their understanding of the nesting and system level object issues concepts, the two most difficult to understand OOD concepts. It appears that those who use a more OO approach in their final solution experienced more development in their understanding of the two difficult to understand OOD concepts. Therefore, we find some support for H3 from this analysis; the use of an OOD solution may require a good understanding of the difficult to understand concepts.

	Most OOD final solution	Least OOD final solution	t-value
Object-Oriented Concepts			
Creating Objects	-.06	-1.76	1.44
Object Characteristics	.05	-1.03	0.91
Interaction Between Objects	.27	-1.74	2.12
Nesting	.76	.64	1.98[d]
System Level Object Issues	-.26	-1.47	2.43[c]
Structured Development Constructs			
Functionality	1.13	2.78	1.08
Flow	.51	1.05	0.56

df = 6
[a]p-value < 0.001; [b]p-value < 0.01; [c]p-value < 0.05; [d]p-value < 0.10

Table 6. Comparison of Groups with most and least OOD approach in Final Solution

Hypothesis 4: Consistency of Developer's Mental Models

Hypotheses 4a-4c concern the consistency between developers' MMs at the beginning and end of the project. To investigate these hypotheses, scores were created to assess the consistency of developers' MMs based on the r_{wg} score developed by James, Demaree and Wolf (1993), and the procedure recommend by Espinosa (2001). The results are presented in Table 7. The r_{wg} score measures within-group interrater agreement. To the best of our knowledge there are no test statistics to determine significant differences between these scores. Despite this limitation, it has become widely used to assess consistency of teams' MM (cf. Espinosa, 2001). Higher scores (closer to 1) indicate greater agreement (or consistency) in responses to the survey.

	r_{wg} - beginning of project	r_{wg} - end of project
Object-Oriented Concepts		
Creating Objects	0.79	0.70
Object Characteristics	0.99	0.98
Interaction Between Objects	0.88	0.98
Nesting	0.95	0.96
System Level Object Issues	0.96	0.98
Structured Development Constructs		
Functionality	0.98	0.98
Flow	0.98	0.99

Table 7. Consistency of All Developers' Mental Models.

We anticipated that developers' MMs concerning the SD concepts would be relatively consistent at both the beginning and end of the project (H4a). This hypothesis is retained as there are only minor differences in the consistency with regard the functionality and flow concepts (Table 7).

We also expected more MM consistency among easier-to-understand OOD concepts (creating objects, object characteristics, and interaction between objects) at the end of the project (H4b). The interaction concept demonstrates greater consistency at the end of the project. Surprisingly, the creating objects concept (considered the easiest to understand concept) is less consistent than at the beginning. H4b is not supported.

Finally, we expected that the difficult-to-understand OOD concepts would be less consistent than the easier-to-understand OOD concepts (H4c). However, it is not supported as there seems to be as much consistency between the developers MM regarding the difficult-to-understand as the easy-to-understand OOD concepts at both the beginning and end of the project (Table 7). Note that this analysis includes one developer who was hired as an OOD expert. It seemed worthwhile to consider how consistent or inconsistent the OOD expert's MM was with the rest of the team compared to the consistency between the other team members.

To investigate if team members are more consistent with one another or with the OOD expert, r_{wg} statistics were computed for each pair of developers. Then, each pair that included the OOD expert was averaged, then each pair excluding the OOD expert averaged. These results, based on the initial administration of the survey, appear in Table 8.

	Average r_{wg} – pairs including OOD Expert	**Average r_{wg} – pairs excluding OOD Expert**
Object-Oriented Concepts		
Creating Objects	0.97	0.89
Object Characteristics	0.95	0.81
Interaction Between Objects	0.97	0.83
Nesting	0.98	0.91
System Level Object Issues	0.96	0.96
Structured Development Constructs		
Functionality	0.97	0.97
Flow	0.97	0.97

Table 8 – Consistency of Mental Models – Pairwise Comparisons (Initial Administration of Survey)

The OOD non-experts were less consistent between themselves than they were with the OOD expert. The creating objects, object characteristics, and interaction OOD concepts are less consistent between the pairs excluding the OOD expert than the pairs include the OOD expert. This is surprising as one would expect the pairs excluding the OOD expert, who attended the same training and were long time employees of the company, to have more consistent MMs among themselves.

Inspection of the data revealed that one subject's understanding of many concepts was inconsistent with other team members. This subject is the project manager who did not attend all the training courses. Scores were re-computed excluding this subject from the analysis and the level of consistency among the pairs excluding the OOD expert on the team, as well as the consistency between the pairs including the OOD expert are consistently quite high. These results are summarized in Table 9.

	Average r_{wg} – pairs including OOD Expert	**Average r_{wg} – pairs excluding OOD Expert**
Object-Oriented Concepts		
Creating Objects	.97	.96
Object Characteristics	.96	.94
Interaction Between Objects	.97	.94
Nesting	.99	.98
System Level Object Issues	.97	.96
Structured Development Constructs		
Functionality	.97	.97
Flow	.97	.97

Table 9. Consistency of Mental Models – Pairwise Comparisons Excluding the Project Manager (Initial Administration of Survey)

However, that team members' MMs were highly consistent with the OOD expert at the outset of the project again questions the difficulty that expert SD developers face in acquiring OOD knowledge. A similar analysis was conducted of the consistency of team members' MM at the end of the software development project and presented below in Table 10. The MMs are highly consistent.

	Average r_{wg} – pairs including OOD Expert	**Average r_{wg} – pairs excluding OOD Expert**
Object-Oriented Concepts		
Creating Objects	.97	.97
Object Characteristics	.98	.97
Interaction Between Objects	.98	.98
Nesting	.97	.97
System Level Object Issues	.97	.97
Structured Development Constructs		
Functionality	.97	.97
Flow	.97	.97

Table 10. Consistency of Mental Models – Pairwise Comparisons (Final Administration of Survey)

The analysis was repeated again excluding the project manager subject. By the end of the project, this subject's MM was much more consistent with the other team members as evidenced by lower differences between her scores and those of other team members. Further, the measures of the consistency of team members' MM are highly consistent regardless of whether or not the project manager's responses are included in the analysis (compare Tables 10 & 11).

	Average r_{wg} – pairs including OOD Expert	Average r_{wg} – pairs excluding OOD Expert
Object-Oriented Concepts		
Creating Objects	.99	.97
Object Characteristics	.98	.97
Interaction Between Objects	.97	.98
Nesting	.97	.98
System Level Object Issues	.97	.97
Structured Development Constructs		
Functionality	.97	.97
Flow	.97	.97

Table 11. Consistency of Mental Models – Pairwise Comparisons Excluding the Project Manager (Final Administration of Survey)

DISCUSSION

To better understand the challenges faced by software developers transitioning from SD to OOD, this study investigated changes in developers' knowledge of OOD and SD concepts over a one-year development project. Our results suggest that understanding functionality (an SD concept) is positively and significantly associated with four of five OOD concepts. These results support the idea that developing knowledge of OOD is not incompatible with existing SD knowledge. We also investigated whether individual developers understanding of OOD concepts changed. However, only the two most difficult-to-understand concepts reflect significant changes in understanding. It appears that developers understanding of the three easiest OOD concepts was acquired via classroom instruction and changed little through a one-year project. This finding calls into question the belief that OOD is an extremely difficult transition for experienced developers. However, developers that used a more OOD approach at the end also had greater development of the most difficult to understand OOD concepts; hence understanding of these concepts may be essential to applying OOD.

We also assessed the consistency between developers' MMs. Only the OOD concept of interaction appears to have greater consistency between developers by the end of the project. Overall, developers' MMs were quite consistent at the beginning and end of the project. Surprisingly, the easiest OOD concept (creating objects) demonstrated the least consistency among developers' MMs.

LIMITATIONS

There are two primary limitations to this study: the small sample size and the lack of a pre-test. However, the longitudinal design allowed us to assess the understanding of OOD and SD concepts before and after a one-year software project. Despite the small sample (n=8), differences were detected implying adequate power however, additional differences might be detected with a larger sample.

CONCLUSION

Firms desiring to reap the benefits of OOD often train experienced SD developers in OOD to retain their knowledge of business practices. However, some suggest that prior SD knowledge may interfere with the ability of those developers to transition to an OOD methodology; that developers must "unlearn" SD concepts to "learn" OOD concepts (Nelson, et al., 2002). Our longitudinal study did not support the presence of proactive interference between SD and OOD. The implication for managers is that firms may benefit from the switch to OOD by training their current developers. For academics, the results of this study suggest that prior research may have over-emphasized issues of negative transfer and proactive interference while ignoring the potential for positive transfer between SD and OOD.

REFERENCES

1. Alvarez, J. M., Diaz, M., Llopis, L., Pimentel, E. and Troya, J. M. (2003) An Object-oriented Methodology for Embedded Real-time Systems, *The Computer Journal,* 46, 2, 123-147.
2. Armstrong, D. (2001) Charting the Rocky Shoals of an Object-Oriented Mindshift, Unpublished Doctoral Dissertation. The University of Kansas.
3. Briand, L., Arisholm, E., Counsell, S., Houdek, F. and Thevenod-Fosse, P. (1999) Empirical studies of object-oriented artifacts, methods, and processes: state of the art and future directions, *Empirical Software Engineering: An International Journal,* 4, 4, 387-404.
4. Espinosa, J. A. (2001) Shared Mental Models: Accuracy and Visual Representation, *Proceedings of the Americas Conference on Information Systems* (AMCIS), Boston, MA.
5. Fayad, M. E., Tsai, W. T. and Fulghum, M. L. (1996) Transition to Object-Oriented Software Development, *Communications of the ACM* 39, 2, 108-121.
6. Hatton, L (1998) Does OOD Sync With How We Think? *IEEE Software*. May/June, 46-54.
7. James, L. R., Demaree, R. G. and Wolf, G. (1993) r_{wg}*:* An assessment of within-group interrater agreement, *Journal of Applied Psychology*, 78,2, 306-309.
8. Mathieu, J., Goodwin, G. F., Heffner, T. S., Salas, E. and Cannon-Bowers, J. A. (2000) The influence of shared mental models on team process and performance, *Journal of Applied Psychology*, 85,2, 273-283.
9. Nelson, H. J., Armstrong, D. J. and Ghods, M. (2002) Old Dogs and New Tricks, *Communications of the ACM*, 45,10, 132-137.
10. Sumit, S., Nerur, P. and Mahapatra, R. (2001) Revolution or Evolution? A Comparison of Object-Oriented and Structured Systems Development Methods, *MIS Quarterly*, 25, 4, 457-471.
11. Rosson, M. B. & Alpert, S. R. (1990) The cognitive consequences of object-oriented design, *Human-Computer Interaction*, 5, 345-379.
12. Shaft, T. M. and Jasperson, J. (2001) Knowledge Transfer of Modeling Practices to Object-Oriented Development, *Presentation, Center for MIS Studies Fall Retreat*.
13. Shaft, T. M. and Vessey, I. (1995) The Relevance of Application Domain Knowledge: The Case of Computer Program Comprehension, *Information Systems Research,* 6, 3, 286-299.