A comprehensive model for IS development: information processing perspective

Shih-Wei Chou[1)], Chales Chen, Pi-Yi Chen, Mong-Young He

[1)] National Kaohsiung First University of Science of Technology, Department of Information Management (swchou@ccms.nkfust.edu.tw)

Abstract

In order to develop IS (information systems) effectively, this study proposed a comprehensive model that delineates the relationships among contingency configurations of design variables, risk management, and software development success. Specifically, this paper examined the fit between coordination strategy and task interdependence, and their impact on risk resolution (such as reorienting initiative and modifying strategy) and IS success (such as productivity and process satisfaction). Drawing on information processing theory, we developed a research model. Experimental design was conducted to test the model. Projects with low task interdependence exhibited greater risk resolution achievement, productivity, and process satisfaction than projects with high task interdependence. Also, organic coordination strategy demonstrated better risk resolution achievement, productivity, and process satisfaction than those with mechanic coordination strategy. Finally, surprisingly, our findings regarding the interaction effect between task interdependence and coordination strategy only partially supported our hypotheses.

1. Introduction

How to develop a software project successfully remains a theoretical as well as a managerial challenge. Using inappropriate software development strategy not only entails high cost and risk, but usually results in low quality software project. To address this, two different streams of approach were proposed. The first one emphasized technical innovations such as development of new and enhanced methods and tools, from which a software team may enhance the software development performance [5] [8]. The second one argued that the success of a software design is contingent on the fit between project management practices and software development process. This perspective focuses on a variety of context variables such as task-context variable (task difficulty, task interdependence, task nonroutineness, task variety, work group coordination) [1] [10], and psychosocial-context variable (goal conflict, cognitive fit) [1] [18] [21]. This study uses the second approach as an analysis lens. More specifically, we examine the fit between two variables—task interdependence and coordination strategy, which in turn influence the success of software development.

In addition to the above context variables, achieving risk management may influence the success of software implementation [9]. The concepts of risk management have been applied successfully to software development over the last decades. The purpose of software risk management is to identify the possible barriers that may lead to system failure. Lyytinen et al. [15] confirmed this by defining software risk as a particular aspect of a development task, process, environment, which if ignored, will increase the likelihood of project failure. In other words, risk management helps software practitioners focus on many aspects of a problematic situation. For example, it underlines potential causes of failure, and it helps link potential threats to possible actions and risk resolution approaches.

Software development is a highly social task that usually entails interactive process and a variety of information-intensive activities such as identifying and analyzing user requirements. A variety of factors have been identified that may exert an influence on the success of IS (information systems) development and implementation, for example management's control over scare resources [19], user participation and involvement [2], resource interdependence [14], environmental uncertainty and task uncertainty [1] [10], and so on. However, there is relatively few study addressing IS development success from both the fit for effectively processing information exchange and risk management approaches. To fill this gap, this study has two research objectives. The first one is to develop a contingent model that deals with the match or fit between a project's task interdependence and the coordination strategy [1] [10].

The second objective is to investigate the role of risk management in mediating the relationships between the above contingent model and IS success.

2. Theoretical background and Hypotheses Development

2.1 Risk management of IS development

To facilitate the creation of high quality software (high quality and IS development process satisfaction), previous studies [1] [4] [14] [20] have identified two critical issues—fostering information exchange and eliminating IS development risk. To address the former, this paper proposes a contingent model that focuses on the information exchange and interactions among stakeholders in a most effective way. Success of a IS development relies on the "fit" between contextual variables such as task interdependence and coordination strategy [1] [4] [20].

Although the critical role of risk management in affecting IS development success has been recognized by previous research, what is the context that fosters the implementing of risk resolution remains to be specified. To deal with this, this study proposes a model by specifying two contextual variables in IS design—task interdependence and coordination strategy, and identifies their influence on the risk management within software development. As noted by prior studies [1] [10], a project's task characteristics (task interdependence) might dictate the application of the most appropriate coordination strategy. This also indicates that the match or fit between these two variables may provide the extent of required message exchange and information integration. According to information processing theory, in a IS development environment, the uncertainty is associated with an absence of information. In addition, the uncertainty may result in inappropriate risk resolution strategy [9]. Thus, given that effective information processing, such as the fit between task interdependence and coordination strategy, IS development team may reduce the uncertainty to the maximum extent, which implies that the team may conduct a more feasible and effective risk resolution strategy. On the other hand, since using an appropriate risk management strategy may help IS development team examine software requirements and goal in a more comprehensive way, better task performance and team member satisfaction may be expected. Based on the above arguments, we developed a research framework as shown in Figure 1.

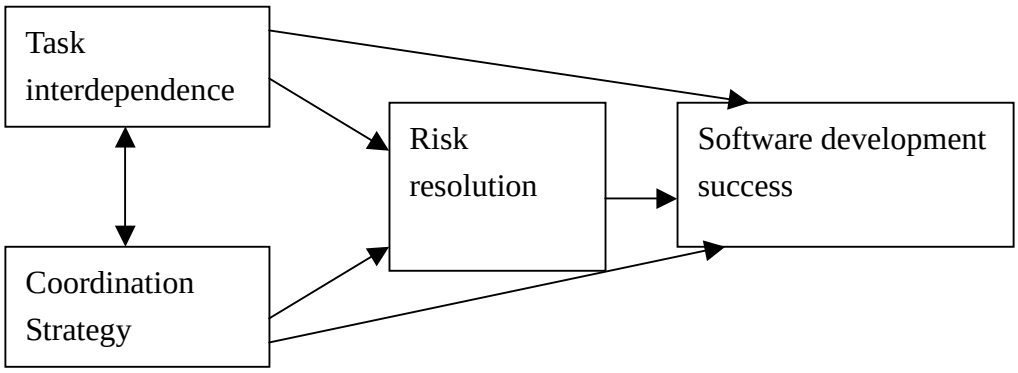

Fig. 1 Research framework

H 1: The implementation of risk resolution strategy will have a positive effect on software development success.
H 2: Risk management strategy will mediate the effects of goal conflict on software success, and coordination strategy on software development success.

2.2 Task interdependence

Task interdependence refers to the extent to which a task requires organizational units to engage in workflow exchanges of product, information, skills, or resources. Task interdependence also implies that actions taken in one unit influence the processes and products of other units [1] [12] [19]. Under conditions of low task interdependence, the contributions of individuals and work units are additive. High task interdependence implies that when a work unit or team member needs to integrate his/her effort with others and the output of another is needed as the input to do his/her tasks.

In an IS development context, low task interdependence stands for minimal real-time data exchange or system state

dependence (such as running, shutdown, or database updated) generated among modules. The design and coding of OLTP systems are typical examples of low task interdependence. In contrast, for high interdependence tasks such as OLAP systems, the design and coding of software requires collaboration across multiple cross-functional teams or among a diversity of IS staffs or stakeholders, such as executive users, programming language specialists, database administrators, network administrators, system and security engineers, third party vendors, and subcontrators [1] [13].

According to information processing theory [10] [16], information processing must meet the dual needs of reducing both ambiguity and uncertainty. Ambiguity is due to lack of understanding, and the existence of multiple and conflicting interpretations about an organizational situation. Uncertainty is associated with an absence of information. In terms of risk resolution strategy, in order to analyze risks and propose appropriate risk resolution strategy, software development team requires a wide scope of information processing and sharing to reduce the uncertainty and ambiguity. This is because risk resolution usually needs information that is timely, has broad scope, has various forms of aggregation, and is integrated [9] [10]. Regarding the high interdependence, it entails higher uncertainty and more information exchange than those of low interdependence task, this implies that the risk management is more difficult to achieve successfully for high interdependence task. In addition, the performance of IS development for high interdependence task is diminished owing to more integration and exchange of information. The above analysis leads to the following hypotheses:

H3: The effect of task interdependence will have a positive impact on risk management
H4: The effect of task interdependence will have a positive impact on software success
H5: Risk resolution strategies characterized by low task interdependence conditions will be more successful than those characterized by high task interdependence conditions.
H6: Software design and coding activities characterized by low task interdependence conditions will be more successful than those characterized by high task interdependence conditions.

2.3 Coordination strategy

Coordination refers to the mode of linking together different parts of an organization to accomplish a set of collective tasks [22]. According to coordination theory, in order to collaborate and make decision effectively, it is important that organizational members with allocated tasks use the communication and control mechanisms that facilitate information exchanges and decisional autonomy.

In an IS development context, as IS developers are usually confronted with a diversity of task requirements, which may be unexpected, constantly changing, difficult to analyze, and interdependent, they need an effective coordination strategy to reduce the uncertainty and ambiguity. One of such strategy is using informal horizontal communications channels that enable the timely sharing of problem-solving expertise and clarification of one's task outputs, which must ultimately be integrated with the work of others.

To reduce the uncertainty and ambiguity due to the execution of both software development and risk resolution strategy, it is essential to provide a coordination strategy that exchanges information for the timely sequencing, scheduling, and synchronization of activities between team members. Consequently, IS developers may adjust risk resolution mission and strategy easily owing to the available information that is timely, has broad scope, has various forms of aggregation, and is integrated [10] [13] [17]. Thus, we have hypothesis 7 to 10.

H7: The effect of coordination strategy will have a positive effect on risk management.
H8: The effect of coordination strategy will have a positive impact on software success.
H9: Risk resolution strategies managed via an organic coordination strategy will be more successful than those managed via a mechanistic coordination strategy.
H10: Software design and coding activities managed via an organic coordination strategy will be more successful than those managed via a mechanistic coordination strategy.

2.4 Fit between task interdependence and coordination strategy

Increasing task interdependence usually implies the requirement for higher frequency and volume of data exchange as well as decision-making between work units [1] [10]. If a mechanism coordination mode were used with high task interdependency, information-processing capacity would be insufficient. On the other hand, an organic coordination

strategy provides a greater capacity to clarify ambiguity, define problems, and reach agreement due to exchange and process information in a timely manner, as well as employ rich media—e.g. fact-to-face and direct contact. However, organic coordination under conditions of low task interdependence can overload the decision-making structure because unnecessary interactions and information exchange may disturb ongoing, already effective tasks, consuming both time and effort and frustrating workers. In contrast, well-established rules, procedures, and formal work plans regarding tasks that minimize the unnecessary information exchange and peer involvement (i.e. mechanistic coordination strategy) demonstrate to be more appropriate for low task interdependence. The above arguments suggest the following hypotheses:

H11: As software design and coding tasks become more task interdependence, the greater the effect of an organic coordination strategy on the performance of risk management.
H12: As software design and coding tasks become move task interdependence, the greater the effect of an organic coordination strategy on IS success.

3. The study's research design

Laboratory experiments were conducted that use a 2*2 factorial design. The contingency factors employed in this study are task interdependence and coordination strategy. While task-process satisfaction and team productivity were adopted to measure the success of software development (i.e. dependent variables), reorganize task and modify strategy were used to measure risk resolution strategy. The model is shown in Figure 1.

3.1 Subjects and sub-team composition

Ninety-six subjects were derived from a population of graduate students enrolled in evening sections of a software engineering course that was offered by MIS (management of information systems) department at a public university. The students who participate in this experiment have full-time job in the daytime, and have experience of achieving software project or implementing IS as system analysts or application developers in a variety of industries such as construction, food, retail, computer, wholesale, financial, and so on. The students were randomly assigned to the software development groups. Subjects were asked to complete a short software project, which required them to use software development skills such as systems analysis and design, risk management, and implementation (using visual basic (VB) programming language). It was felt (and validated in a pretest) that the academic and practical experience of these part-time students was capable of accomplishing the assigned, reasonably complex IS project, including software analysis, design, risk analysis and resolution, and coding tasks, within a reasonable amount of time. In addition, a comparison (t-test) of the mean grade point average of the student subjects indicated no significant differences in grade point average.

3.2 Experimental tasks

The experimental task assigned to each project team contains two software subsystems. The duration of the task for each team did not exceed a total of twelve hours (two three-hour sessions on two consecutive days). Since the task assigned to the project team contains only the fundamentals of software development, a pilot study showed that the task can be accomplished within twelve-hour time period for a three-person team. Task complexity was controlled across team members by ensuring equality in assigned function points for the modules to be completed by team members. Function points were evaluated by assigning weights to and summing the count of outputs from the modules, inputs/outputs of user queries, inputs to the modules, file data items, and data items sent to, shared with, or received from other modules [6]. The total function point count was 58 across all groups.

4. Experimental manipulations

4.1 Experimental procedures

The experimental groups executed the entire set of experimental tasks in a lab with personal computers that provide the Visual Basic programming language. The experimental tasks achieved by each IS development team include system and requirements analysis, design, coding, testing, risk management, and integration. Subjects were instructed to limit their interactions concerning the experiments. The task executed in the experiment contains three major steps: preexperimental activities, task execution, and a ten-minute debriefing.

4.2 Data analysis and results

In order to validate our research framework, this study employed PLS (partial least square) and ANCOVA. PLS allows latent constructs to be modeled, and it makes minimal demands in terms of sample size to validate a model compared to alternative structural equation modeling techniques [3].

To validate our measurement model, three types of validity were examined: content validity, convergent validity, and discriminant validity. The purpose of content validity is to ensure the consistency between the measurement variables and the extant literature. This was done by interviewing experienced practitioners and pilot-testing the instrument. To assess convergent validity, we examined composite reliability and average variance extracted (AVE) from the measures [7]. Table 1 illustrates the results of confirmatory factor analysis. Table 2 shows that values of AVE and composite reliabiliability by our measures are above the acceptable value. In addition, Figure 2 lists the detailed information concerning the measures of the paths in our research model in Figure 1, all measures are significant on their path loadings at the level of 0.05. Finally, as shown in Table 2, we verified the discriminant validity of our instrument by looking at the square root of the average, which is acceptable.

Table 1. Comfirmatory Factor analysis, Constructs, and Item wording

	1	**2**	**3**	**4**	**t-value**
1. task interdependence	**1.00**	0.00	0.31	0.43	
2. goal conflict	0.00	**1.00**	0.54	0.55	
3. Risk management					
3.1 formulate measure goals	0.13	0.23	**0.57**	0.22	7.92***
3.2 organize the improvement initiative as a project	0.21	0.51	**0.83**	0.62	10.42***
3.3 coordinate other improvement initiatives	0.25	0.60	**0.91**	0.61	15.48***
3.4 identify the relationships among plans, problems, progress, and results	0.36	-0.02	**0.57**	0.31	8.91***
3.5 conduct reviews at regular intervals	0.32	0.60	**0.83**	0.59	10.67***
3.6 make the results visible	0.05	0.14	**0.54**	0.32	8.74***
3.7 emphasize collaboration	0.30	0.46	**0.85**	0.62	11.54***
3.10 use an incremental improvement strategy	0.23	0.44	**0.89**	0.65	12.53***
3.11 consider alternative improvement ideas	0.38	0.29	**0.66**	0.48	9.53***
3.14 identify and solve specific problems	-0.13	0.53	**0.55**	0.38	8.95***
3.15 adapt the strategy to the task	0.26	0.22	**0.65**	0.31	9.84***
3.16 design effect measures	0.19	0.19	**0.56**	0.18	9.02***
4. Software development success: Concerning the software development process…					
4.1 I am satisfied with the time of conducting the IS project	0.33	0.43	0.55	**0.88**	12.01***
4.2 the interaction between members is efficient	0.39	0.28	0.43	**0.84**	10.52***
4.3 the assigned task is fair	0.25	0.68	0.67	**0.87**	11.45***
4.4 the available resource is fair	0.42	0.38	0.44	**0.82**	10.41***
4.5 I am satisfied with the results of the IS	0.37	0.41	0.55	**0.80**	10.23***
4.6 Team productivity (function point count)	0.40	0.47	0.58	**0.74**	9.41***

*p<0.1, **p<0.05, ***p<0.01

5

Table 2. Correlation of constructs, square root of AVE value, composite reliability, Cronbach's Alpha, and AVE

		1	2	3	4	CR	Cronbach's Alpha	AVE
1	Coordination strategy	**1**				1	1	1
2	Goal conflict	0	**1**			1	1	1
3	Risk management	0.31	0.54	**0.71**		0.92	0.89	0.51
4	Software success	0.43	0.55	0.66	**0.82**	0.93	0.91	0.68

1. The bolded diagonal values are square roots of the AVE (average variance extracted).
2.CR is composite reliability

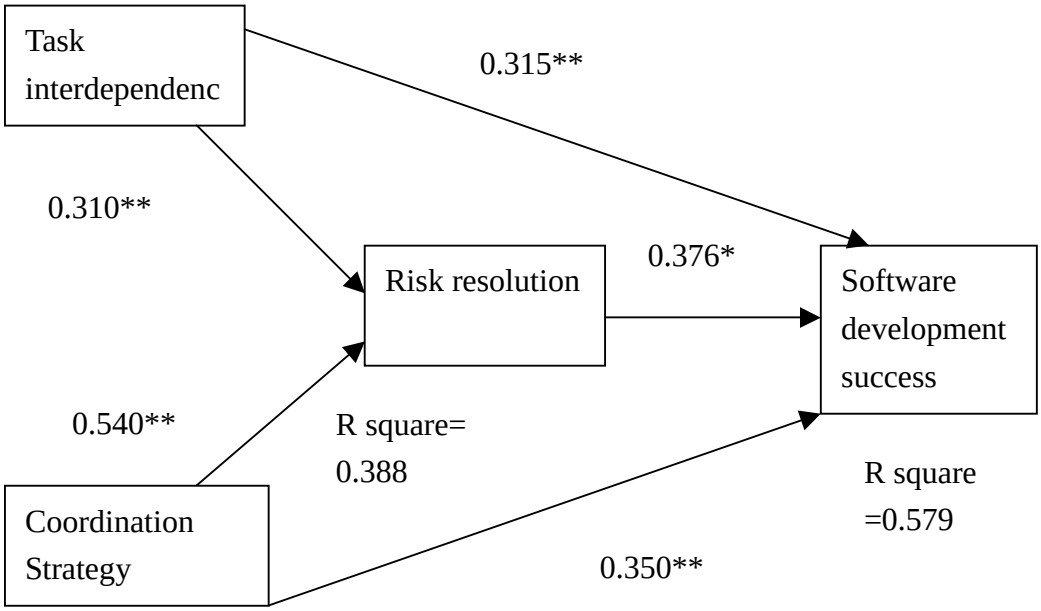

Fig. 2 Results of PLS analysis (Note: *p<0.05, **p<0.01)

With an adequate measurement model and an acceptable level of multicollinearity, hypothesis 1, 3, 4, 7, and 8 were tested with PLS. The results of the analysis are shown in Figure 2. These hypotheses are supported as expected. In addition, the results also show that risk resolution actions have a mediating effect between "software design and coding activities" and two independent variables (task interdependence and coordination strategy). Thus, the mediating effect of risk management is supported as hypothesized (H2).

To test the interaction effects, we used ANCOVA. Table 3 illustrates the biserial and Pearson correlations among the variables. The correlations between productivity and process satisfaction were not significant, and the covariate (programming ability) was correlated with productivity. According to Table 4-6, the main effects of coordination strategy and task interdependence on three different variables are significant, risk management (p values for both coordination strategy and task interdependence are 0.003 in Table 4), process satisfaction (p values for coordination strategy and task interdependence are 0.001 and 0.004 respectively in Table 5), and productivity (p values for coordination strategy and task interdependence are 0.004 and 0.015 respectively in Table 6). These findings are consistent with the results of PLS as indicated in Figure 2.

Table 3. Pearson correlation matrix of variables

Variables	1	2	3	4	5	6
1. Coordination strategy	1					
2. Task interdependence	0.	1				
3. Risk management	0.49**	0.32*	1			
4. Software development process satisfaction	0.48**	0.41**	0.58**	1		
5. Productivity	0.52**	0.49**	0.37*	0.24	1	
6. Programming ability	0.215	0.268	0.107	0.19	0.32*	1

Notes: **p<0.01; *p<0.05 (two-tailed tests of significance)

Table 4. ANCOVA results for risk management

Source of variation	Degree of Freedom	Mean square	F	Significance of F	Power
Coordination strategy	1	172.80	17.79	0.003**	0.94
Task interdependence	1	242.49	24.98	0.003**	0.96
Coordination strategy * Task interdependence	1	0.18	0.19	0.89	0.06
Programming ability	1	0.70	0.07	0.79	0.05
**p<0.01; *p<0.05, Total degree-of-Freedom= 32, R*R= 0.64					

Table 5. ANCOVA results for software development process satisfaction

Source of variation	Degree of Freedom	Mean square	F	Significance of F	Power
Coordination strategy	1	54.19	15.76	0.001**	0.93
Task interdependence	1	33.42	9.72	0.004**	0.84
Coordination strategy * Task interdependence	1	1.27	0.37	0.549	0.13
Programming ability	1	4.37	1.27	0.269	0.23
**p<0.01; *p<0.05, Total degree-of-Freedom= 32, R*R= 0.40					

Table 6. ANCOVA results for productivity

Source of variation	Degree of Freedom	Mean square	F	Significance of F	Power
Coordination strategy	1	188.04	9.93	0.004**	0.87
Task interdependence	1	108.87	6.73	0.015**	0.74
Coordination strategy * Task interdependence	1	82.85	4.82	0.037**	0.54
Programming ability	1	9.55	0.59	0.449	0.17
**p<0.01; *p<0.05, Total degree-of-Freedom= 32, R*R= 0.41					

Considering the influence of different task interdependence conditions, as expected, the values of cell means and adjusted marginal means (AMM) show that the low task interdependence (AMM=3.78) produced more successful risk management than did the high task interdependence situation (AMM=3.54). Regarding software success, low task interdependence (AMM=3.76) situations resulted in more process satisfaction than did the high task interdependence (AMM=3.38) case. Finally, the groups under low task interdependence conditions (AMM=19.98) had greater productivity than those receiving high task interdependence treatment (AMM=15.31). In sum, these findings support Hypothesis 5 and 6. Concerning coordination strategy, from the AMMs, hypotheses 9 and 10 are also supported.

As shown in Table 4-6, the results of the interaction effect (between task interdependence and coordination strategy) on three different variables (i.e. risk management, process satisfaction, and productivity). These results show that among these three variables, only the interaction effect on productivity is significant—in Table 6, p= 0.037< 0.05. Further, we

used post hoc pairwise comparisons to evaluate the relative difference between organic versus mechanistic strategy at each level of task interdependence (high and low). Surprisingly, in terms of productivity, the difference between organic coordination strategy and mechanistic coordination strategy is not significant ($F= 0.419$, $p=0.529>0.05$) under high task interdependence situation. On the other hand, the above difference under low task interdependence condition is significant—the teams using organic coordination strategy has better productivity than did the teams using mechanistic coordination strategy (mean difference = 7.85, $p=0.002<0.05$). Thus, neither H11 nor H12 is supported.

5. Discussions, implications, and limitations

The interpretations of our findings are three-fold. First, as expected, our data indicates that low task interdependence creates greater IS success (productivity and satisfaction) than that of high task interdependence. In addition, low task interdependence also exerts higher influence on risk management strategy than that of high task interdependence. These findings are consistent with prior work [1] [18]. Under low task interdependence situation, owing to the potential for greater task focus, it is less likely that members of the IS development team achieve the tasks that do not directly relate to IS implementation. For example, IS team does not have to perform role clarification, negotiation, and task-integration among team members [14]. In the light of IS project management, the above findings suggest that modularized software design should be reinforced. Since it allows project team members to be assigned largely independent design tasks, this may lead to more successful IS implementation and risk resolution.

Second, as the theory predicted, IS teams using an organic coordination strategy were observed to achieve both IS implementation and risk management more successfully than those using mechanic coordination strategy. Organic coordination strategy fosters the open and spontaneous communication and information exchange among team members. Thus, in the light of information processing theory, organic coordination strategy facilitates knowledge acquisition, sharing, and integration, from which IS teams may complete their assigned software design and coding more efficiently and effectively—i.e. greater productivity [1] [10]. For IS teams, lack of immediate access to the information relating to task or environment (mechanistic coordination) not only retards the process of overall code integration, but hinders the capability of analyzing risks and modifying strategy [9]. This further confirms organic coordination is a better choice than mechanistic coordination in terms of risk management.

Finally, surprisingly, the hypothesized task interdependence and coordination strategy interactions were not supported. As shown in Table 6-8, while the above interaction effects on either risk resolution or process satisfaction was not significant, such effects exerted significant influence on productivity. However, the above contingent effect is inconsistent with our hypothesis. Prior work [1] suggested that when IS team was confronted with the task of high interdependence, employing organic coordination is better than mechanistic coordination in terms of IS implementation success. This is because high task interdependence entails more intensive information flow among members than does low task interdependence, but using mechanistic coordination rather than organic coordination impedes the communication and message exchange to a certain extent.

Three implications may be drawn. First, the overhead associated with high task interdependence was alleviated by organic coordination, since knowledge integration and exchange was fostered by a powerful communication mechanism—organic coordination strategy [1] [9]. Second, organic coordination under conditions of low task interdependence did not necessarily overload the decision-making structure with redundant information. This indicates that coordination strategy plays a more important role than task interdependence in affecting productivity. Finally, the interaction effect on either process satisfaction or risk management was not significant. The result indicates that the advantage of using organic coordination may not compensate for the complaints and displeasure caused by high task interdependence, such as waiting for team members to complete the required subtask or revising completed code to achieve code integration. To what extent is the above argument valid, and what are the possible factors that may affect the interaction effects? Clearly, we need further research to address these issues.

6. Limitations

Although laboratory experimentation enables one to conduct empirical research by isolating treatment influence and controlling of the confounding variables, it is threatened with external validity. Using students to achieve the assigned IS projected may restrict the generalizability of results to an organization. This study avoided the above problem to a certain extent by using the student subjects who had working experience.

7. Conclusions

The contribution of this study is to specify the influence of design (coordination strategy) and task (interdependence) factors on three critical issues of IS development--risk management, productivity, and process satisfaction. Consisting with the information processing theory, our findings imply that IS teams that employed organic coordination achieved IS development more successful than those used mechanistic coordination. On the other hand, the teams that were confronted with the task of low interdependence have better IS development performance than did those with high task interdependence. Surprisingly, our results only partially supported the contingent influence of task interdependence and coordination strategy on the results of project management. Classical software project management has often relied on the use of predefined methodologies, scheduled periodic design meetings, and formal documentation of system design and change activities. In contrast, our study indicates that, as the functional complexity of information systems increases such as globally interconnected online analytical processing systems, it is important for project manager to adapt traditional software development approach. Under such situation, IS project manager should not only deal with the fit between theoretically prescribed designs (such as organic versus mechanistic coordination) and their task context (such as task interdependence, task variety, task uncertainty) [1] [9], but emphasize the risk resolution strategy such as reorganizing initiative and modifying strategy. Future research may explore other types of contingent relationship and its influence on IS success, for example task analyzable and cognitive ability.

References

[1] Andres, H. P., and Zmud, R., "A Contingency Approach to Software Project Coordination," JMIS, (18:3), 2002,pp. 41-70.

[2] Barki, H., Rivard, S., and Talbot, J., "Toward an Assessment of Software Development Risk," Journal of MIS, (10:2), 1993, pp. 203-225.

[3] Chin, W. W., Marcolin, B. L., and Newsted, P. R., "A Partial Lease Squares Latent Variable Modeling Approach for Measuring Interaction Effects: Results from a Monte Carlo Simulation Study and an Electronic-mail Emotion/Adoption Study," Information Systems Research, (14:2), 2003, pp. 189-217.

[4] Crowston, K., "A Coordination Theory Approach to Organizational Process," Organization Science, (8:2), 1997, pp. 157-175.

[5] Fichman, R. G., and Kemerer, C. F., "Adoption of Software Engineering Software Innovations: The Case of Object Orientation," Sloan Management Review, (34:2), 1993, pp. 7-22.

[6] Freger, J. B., Function Point Analysis. Englewood Cliffs, NJ: Prentice Hall, 1989.

[7] Hair, J. F., Anderson, R. E., and Tatham, R. L., and Black, W. C., Multivariate Data Analysis (5th ed.), Prentice Hall, Englewood Cliffs, NJ., 1998.

[8] Henderson, J. C., and Cooprider, J. G., "Dimensions of IS Planning and Design Aids: A Functional Model of CASE Technology," Information Systems Research, (1:3), 1990, pp. 227-254.

[9] Iversen, J. H., Mathiassen, L., and Nielsen, P. A., "Managing Risk in Software Process Improvement: An Action Research Approach," (28:3), 2004, pp. 395-433.

[10] Karimi, J., Somers, T. M., and Gupta, Y. P., "Impact of Environmental Uncertainty and Task Characteristics on User Satisfaction with Data," Information Systems Research, (15:2), 2004, pp. 175-193.

[11] Keil, M., "Pulling the Plug: Software Project Management and The Problem of Escalation," MIS Quarterly, (19:4), 1995, pp. 421-447.

[12] Kim, K. K. and Umanath, N. S., "Structure and Perceived Effectiveness of Software Development Subunits: A Task Contingency Analysis," Journal of MIS, (9:3), 1994, pp. 157-181.

[13] Kraut, R. E., and Streeter, L., "Coordination in Software Development," Communications of the ACM, (38:3), 1995, pp. 69-81.

[14] Larsen, K. T., "A Taxonomy of Antecedents of Information Systems Success: Variable Analysis Studies," Journal of MIS, (20:2), 2003, pp. 169-246.

[15] Lyytinen, K., Mathiassen, L., and Ropponen, J., "Attention Shaping and Software Risk—A Categorical Analysis of Four Classical Risk Management Approaches," Information System Research, (9:3), 1998, pp. 233-255.

[16] Malhotra, A., Gosain, S., and EI Sawy, O. A., "Absorptive Capacity Configurations in Supply Chains: Gearing for Partner-enabled Market Knowledge Creation," MIS Quarterly, (29:1), 2005, pp. 145-187.

[17] Rasch, R. and Tosi, H., "Factors Affecting Software Developers' Performance: An Integrated Approach," MIS Quarterly, (16:3), 1992, pp. 395-413.

[18] Shaft, T. M., and Vessey, I., "The Role of Cognitive Fit in the Relationship Between Software Comprehension and

Modification," MIS Quarterly, (30:1), 2006, pp. 29-55.
[19] Sharma, R., and Yetton, P., "The Contingent Effects of Management Support and Task Interdependence on Successful Information Systems Implementation," MIS Quarterly, (27:4), 2003,pp. 5333-555.
[20] Shenhar, A. J., "From Theory to Practice: Toward a Typology of Project Management Styles," IEEE Transactions on Engineering Management, (45:1), 1998, pp. 33-48.
[21] Sherif, K., Zmud, R. W., and Browne, G. J., "Managing Peer-to-peer Conflicts in Disruptive Information Technology Innovations: The Case of Software Reuse," MIS Quarterly, (30:2), 2006, pp. 339-356.
[22] Van de Ven, A. H., Delbecq, A. L., and Koenig, R., "Determinants of Coordination Modes Within Organizations," American Sociological Review, (41:2), 1976, pp. 322-338.