Summer 2012

Case Study: The Optimization of Information Flow & Organization in Software Development

Farshad Madani
Portland State University

Bing Wang
Portland State University

Xiaowen Wang
Portland State University

Corey White
Portland State University

Liying Wang
Portland State University

ETM

ENGINEERING & TECHNOLOGY MANAGEMENT

Case Study: The Optimization of Information Flow & Organization in Software Development

Course Title:	R & D Management
Course Number:	EMG 536/636
Instructor:	Dr. Tugrul Daim
Term:	Summer
Year:	2012
Author(s):	Farshad Madani, Bing Wang, Xiaowen Wang, Corey White, Liying Wang

ETM OFFICE USE ONLY

Report No.:	
Type:	Student Project
Note:	

Catalogue

Abstract

Trial and error are inevitable in the process of software development; the redesign is wasteful when unfocused communication occurs within the development process. To speed up this development process, we optimized the information flow and redesigned the development organization into chunks. First, we investigated the information flow between tasks and coupled tasks into phases utilizing the Design Structure Matrix (DSM) so that the information exchange among chunks of tasks is as few as possible. In this way, the redundant iterations of tasks are eliminated. Meanwhile, we also grouped the engineers into groups according to the tasks, in this; we eliminated wasteful communications between the groups of engineers. We depict the social network defined by the information flow within the software development sector and compared the new arrangement to the old arrangement in a visible way. This new arrangement will facilitate the communications between engineers and speed up the process of software development.

Introduction

As we all know, software development fails and developers fail to meet their goals if they spend all their time fighting against an inappropriate organizational structure. Therefore, understanding the importance and advantages of the proper organizational structure is vital to the ultimate success of a software development organization. Many Chief Information Officers recognize that the organizational structure of their software development group has an impact on the success of their application development efforts. An organization is defined by much more than boxes containing job titles and names connected by lines representing a reporting structure. Each individual in an organization has the skills; these skills are usually formal or informal performance, future performance incentives, and rewards (compensation) measurements. How to develop and maintain high levels of motivation in project teams in the face of congestion and distraction, while maintaining focus was a project goal to be researched. [1] While many articles have addressed the importance of such issues, few have conducted systematic investigations about their operation in software development.

There are three reasons for a greater emphasis on the organizational and social context of software processes. First, the human element is a critical and dominant factor in the most tool-intensive parts of software development such as the system building process. There are the efficiency and appropriateness tools that are obviously important, for example, the crucial job of tracking down the sources of inconsistency and negotiating their resolution is performed by people. Second, many process studies have indicated a large amount of unexplained variance in performance suggesting that significant aspects of the process are independent of the technological context [2][3][4]. Finally, it is noted that a significant proportion of project effort is devoted to non-programming activities, with some estimates indicating as much as 50% of a work week typically absorbed by machine downtime, meetings, paperwork, and company business, sick and personal days [2][5].

The design and development of complex engineering products needs the hundreds of participants from different backgrounds, leading to the efforts and cooperation of the complex relationship between people and tasks. There are many traditional project management tools such as PERT, Gantt and CPM methods, which do not address problems stemming from this complexity. These tools allow the modeling of sequential and parallel processes, but they fail to address interdependency (feedback and iteration), which is common in complex product development (PD) projects. To address this issue, a matrix-based tool called the Design Structure Matrix (DSM) has evolved. This method differs from traditional project management tools because it focuses on representing information flows rather than work flows. The DSM method is an information exchange model that allows the representation of complex task (or team) relationships in order to determine a sensible sequence (or grouping) for the tasks (or teams) being modeled. This paper will cover how the basic method works and how you can use the DSM to improve the planning, execution, and management of complex PD projects using different algorithms (i.e., partitioning, tearing, banding, clustering, simulation, and eigenvalue analysis). [6]

Software development is predominantly a social activity. It is important to view software development groups, departments, and corporations as social bodies. We learned to use a new data collection method, combined with several techniques commonly used in social network analysis of software organizations. Our technology is different from the ordinary social anthropology, we help the organization to reflect on their internal structure, i.e. the technique is a “mirror” for the subject organization. We cataloged the social network diagrams by using a variety of visualization techniques. The development of an analytical model is to capture the nature of social networks, employing similar strategies to those who establish social networks in the working process of software development. Based on the architectural pattern of emerging design technologies, we provide a communication and organizational model. A set of patterns has been captured by the excellent organization of the production of software development pattern language approach. [7]

Methodology

The Design Structure Matrix (DSM) is a project management tool that aids in business analysis and in managing projects. It is a powerful tool for visualizing, analyzing, innovating, and improving systems, including product architectures, organizational structures, and process flows.

The DSM makes the processes of management easier to visualize, allowing for identifying and representing the elements in a project, keeping track of cyclic task dependencies as well as task flows, and aiding in analyzing how and where to make improvements in the dependencies between systems. The Design Structure Matrix management tool can generate a good flow of information between departments so that each department could know the progress of others and plan for it accordingly. The DSM has also been used to solve the problems of a software development and system architecture. The DSM has been utilized by many of the top blue-chip companies, as it was developed in the 1970s, and it has been shown to help solve problems and improve the organization [8]. It is a suitable process model and very pivotal for product developing. The DSM emphasizes the interdependencies of information, task sequences, and product design: to provide an effective modeling method. In this paper, the establishment of the principle was first introduced.   As an example, DSM customized corrected stent-based process models. Additional models are proposed such as task decomposition and the establishment and optimization of the DSM. The method of establishment and optimization of process model of product design based on DSM can shorten design cycle of product effectively, which will improve competitiveness of developers.

In recent years, analyzing social networks based on data clustering has become a popular topic as it has much significance in a variety of industries. It can be used to prevent terrorist attacks and detect the spread of diseases by detecting communities to give credit to its importance and significance. Moreover, it is easy to boost social development and social cooperation after understanding structures of social networks clearly because social networks are dynamic networks and data clustering can predict changes of social ties. [7] Effective knowledge among team members is a critical factor that can facilitate team knowledge innovation and increase organizational knowledge as a whole. With understanding of the aforementioned, one can conclude that effectively understanding and taking advantage of social networks analysis is beneficial to any organization. From the data mining aspect, social networks are incomplete, huge, complex and dynamic networks, and traditional data clustering methods do not work well in social areas due to these features. Conversely, spectral clustering, as one of most popular modern data clustering algorithms, offers a systematic, flexible and practical solution to problems about social networks. [9] This paper will attempt to depict the current theories and methodology of spectral clustering and its advantages to the information flow within the software development sector and compare the new arrangement to the old arrangement in a visible way.

Data Collection

We interviewed the software development department in a Chinese company named “Datashine” located in Shanghai, China. This software company is focused on the development of enterprise information management systems, consulting, and implementation services. Datashine is a leading enterprise-class information management system and service provider. We collected all the related development data including the relationships between each developer and each task. Firstly, we interviewed the manager of the software department of the Datashine company. They identified 17 tasks for the development of Customer Relationship Management (CRM) platform as Table 1. CRM platform development is a regular business of Datashine, who has engaged in it for about 5 years.

We then defined the 17 tasks as shown in Table 1.

Table 1 Project Schedule

Phase (results)	Task	Work Content	Plan	Person(s)
Preparatory phase (the project plan document)	Task 1	Development Preparation	(1) To determine the demand for programs and the development of the development process. (2) To prepare the database and test server address. (3) To determine the development environment. (4) To complete the project flowchart. (5) To determine the user rights Personnel to freeze the activities automatically obtain and page functions	Tom
Design phase (database and project design document)	Task 2	Software Framework Design	Build solutions to determine the application development framework	Joe
	Task 3	Database Design	(1)The design of the database table fields (2)Completion of the database documents	Robert
Development stage (program code, the solved problems, BUG report)	Task 4	Schedule	Module programming, test and BUG report	Jesse Jack
	Task 5	Attendance Management	Module programming, test and BUG report	Jesse Jack
	Task 6	Reimbursement of Expenses	Module programming, test and BUG report	Jesse Jack
	Task 7	Internal Messaging	Module programming, test and BUG report	John Ben
	Task 8	External mail	Module programming, test and BUG report	John Ben
	Task 9	SMS	Module programming, test and BUG report	George Apple
	Task 10	Reminding System	Module programming, test and BUG report	George Apple
	Task 11	Process Management	Module programming, test and BUG report	Martin Alan Dean

	Task 12	Attachment Upload	Module programming, test and BUG report	Jim
	Task 13	Online Contract Modification	Module programming, test and BUG report	Daniel Kevin
	Task 14	Order Management	Module programming, test and BUG report	Daniel Kevin
	Task 15	Receivables Management	Module programming, test and BUG report	Peter Vincent
	Task 16	Invoice Management	Module programming, test and BUG report	Peter Vincent
Test phase (test documentation)	Task 17	Program Test & Optimization test	(1)To complete the internal testing procedures in the test server. (2)To hand it over to the testing group. (3)To improve the database document. (4)To complete the development documentation. (5)To sum up the development experiences.	Simon Alice Rose
--	--	--	------	

We presented a DSM to them in the form of a questionnaire according to the work tasks. The questionnaire sample is shown in Appendix 1. There totally 20 persons in the CRM software department who undertake different tasks. Every one of them filled out the questionnaire according to their information flow facts in working for the allocated tasks. This questionnaire is suitable for DSM as well as SNA because it not only addresses the information flow among tasks but also depicts the information flow between persons. We gathered the data for the Analysis of DSM and SNA.

Data Analysis & Result

As discussed, we first depict the DSM matrix as shown in Table 2.

Table 2. The Information Flow among the 17 Tasks

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Task1	#																
Task2	D	#	D														
Task3	D	D	#														
Task4		D	W	#			M										M
Task5		D	W		#												M
Task6		D	W			#					M	M					M
Task7		D	D	M			#						M	M	M	M	M
Task8		D	W					#						M	M	M	M
Task9		D	W						#				M				M
Task10		D	W							#				M	M		M
Task11		W	D			M					#		M	M	M	M	M
Task12		W	D			M						#				M	M
Task13		W	D				M		M		M	M	#			W	M
Task14		W	D				M			M	M			#	W		M
Task15		W	D					M		M	M			W	#		M
Task16		W	D					M			M	M	W			#	M
Task17	M			W	W	W	W	W	W	W	W	W	W	W	W	W	#

D-Daily information flow from the task in the column to the task in the row

W-Weekly information flow from the task in the column to the task in the row

D-Monthly information flow from the task in the column to the task in the row

Blue characters are feedback, and red characters are feedforward.

When we received the initial DSM, we found that Task1, Task2 and Task3 formed the first chunk. This chunk prepares for the development and forms the development design including the framework and database design. This chunk periodically gives information to the programming tasks. Task17 forms a specific chunk, which is testing. This task has inflow and outflow to the programming tasks. The other tasks are programming tasks, which the “W” and “M” are distributed randomly across. So we tried to align all the “W” and “M” closer to the diagonal of the matrix. This helps us to identify the chunks in the programming tasks as shown in Table 3.

Table 3. The DSM with Chunks Optimized

	1	2	3	7	4	15	14	10	8	11	16	13	6	12	9	5	17
Task1	#																
Task2	D	#	D														
Task3	D	D	#														
Task7		D	D	#	M	M	M				M	M					M
Task4		D	W	M	#												M
Task15		W	D			#	W	M	M	M							M
Task14		W	D	M		W	#	M		M							M
Task10		D	W			M	M	#									M
Task8		D	W			M	M		#		M						M
Task11		W	D			M	M			#	M	M	M				M
Task16		W	D						M	M	#	W		M			M
Task13		W	D	M						M	W	#		M	M		M
Task6		D	W							M			#	M			M
Task12		W	D								M		M	#			M
Task9		D	W									M			#		M
Task5		D	W													#	M
Task17	M			W	W	W	W	W	W	W	W	W	W	W	W	W	#

*Yellow areas are the weekly communications that we try to eliminate by rearrangement of persons in order to speed up development. We rearrange the same persons for Task 14 and

Task 15, the same persons for Task 13 and Task 16. In this way the frequent weekly communications are all avoided.

In this way we divided the CRM software department into 5 chunks as shown in Table 4. We then looked at the persons responsible for the tasks and find out that many of the staff belonged to different chunks, which makes the communication more frequent than it should be. In order to eliminate the unnecessary communication between chunks, we rearrange the tasks of conducted by the staff, as shown in Table 5. The rearranged tasks are identified in Table 6.

Table 4. The Chunks of Tasks

Task1	Development Preparation	Chunk 1	Tom			
Task2	Software Framework Design		Joe			
Task3	Database Design		Robert			
Task7	Internal Messaging		Chunk2	John Ben		
Task4	Schedule			Jesse Jack		
Task15	Receivables Management			Chunk 3	Peter Vincent	
Task14	Order Management				Daniel Kevin	
Task10	Reminding System	George Apple				
Task8	External mail	John Ben			Chunk 4	
Task11	Process Management	Martin Alan Dean				
Task16	Invoice Management	Peter Vincent				
Task13	Online Contract Modification	Daniel Kevin				
Task6	Reimbursement of Expenses	Jesse Jack				
Task12	Attachment Upload	Jim				
Task9	SMS	George Apple				
Task5	Attendance Management	Jesse Jack				
Task17	Program Test & Optimization	Simon Alice Rose				Chunk5

Table 4. The Rearrangement of Persons to Tasks

Task1	Development Preparation	Chunk 1	Tom		
Task2	Software Framework Design		Joe		
Task3	Database Design		Robert		
Task7	Internal Messaging		Chunk2	Martin Alan Dean	
Task4	Schedule			Jesse Jack	
Task15	Receivables Management			Chunk 3	Peter Vincent
Task14	Order Management				Peter Vincent
Task10	Reminding System				Jim
Task8	External mail	John Ben			Chunk 4
Task11	Process Management	John Ben			
Task16	Invoice Management	Daniel Kevin			
Task13	Online Contract Modification	Daniel Kevin			
Task6	Reimbursement of Expenses	Jesse Jack			
Task12	Attachment Upload	George Apple			
Task9	SMS	George Apple			
Task5	Attendance Management	Jesse Jack			
Task17	Program Test & Optimization	Simon Alice Rose			Chunk5

Table 6. The Person Change to Tasks

Task	Work Content	Prior Rearrangement	Post Rearrangement
Task1	Development Preparation	Tom	Tom
Task2	Software Framework Design	Joe	Joe
Task3	Database Design	Robert	Robert
Task4	Schedule	Jesse Jack	Jesse Jack
Task5	Attendance Management	Jesse Jack	Jesse Jack
Task6	Reimbursement of Expenses	Jesse Jack	Jesse Jack
Task7	Internal Messaging	John Ben	Martin Alan Dean
Task8	External mail	John Ben	John Ben
Task9	SMS	George Apple	George Apple
Task10	Reminding System	George Apple	Jim
Task11	Process Management	Martin Alan Dean	John Ben
Task12	Attachment Upload	Jim	George Apple
Task13	Online Contract Modification	Daniel Kevin	Daniel Kevin
Task14	Order Management	Daniel Kevin	Peter Vincent
Task15	Receivables Management	Peter Vincent	Peter Vincent
Task16	Invoice Management	Peter Vincent	Daniel Kevin
Task17	Program Test & Optimization	Simon Alice Rose	Simon Alice Rose

In order to show the relationship of persons in the CRM software department, we depicted the social network chart derived from the questionnaire before and after the rearrangement of tasks, shown as Table 6, Table 7, Figure 1, and Figure 2.

Table 6. The Communication Matrix of Persons before Rearrangement

	Tom	Joe	Robert	Jesse	Jack	John	Ben	George	Apple	Martin	Alan	Dean	Jim	Daniel	Kevin	Peter	Vincent	Simon	Alice	Rose
Tom	#	1	1															1	1	1
Joe		#	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
Robert		1	#	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Jesse				#		1	1			1	1	1	1					1	1	1
Jack					#	1	1			1	1	1	1					1	1	1
John				1	1	#								1	1	1	1	1	1	1
Ben				1	1		#							1	1	1	1	1	1	1
George								#						1	1	1	1	1	1	1
Apple									#					1	1	1	1	1	1	1
Martin										#										
Alan				1	1						#			1	1	1	1	1	1	1
Dean				1	1							#		1	1	1	1	1	1	1
Jim				1	1								#	1	1	1	1	1	1	1
Daniel						1	1	1	1	1	1	1		#		1	1	1	1	1
Kevin						1	1	1	1	1	1	1			#	1	1	1	1	1
Peter						1	1	1	1	1	1	1	1	1	1	#		1	1	1
Vincent						1	1	1	1	1	1	1	1	1	1		#	1	1	1
Simon				1	1	1	1	1	1	1	1	1	1	1	1	1	1	#		
Alice	1			1	1	1	1	1	1	1	1	1	1	1	1	1	1		#	
Rose				1	1	1	1	1	1	1	1	1	1	1	1	1	1			#

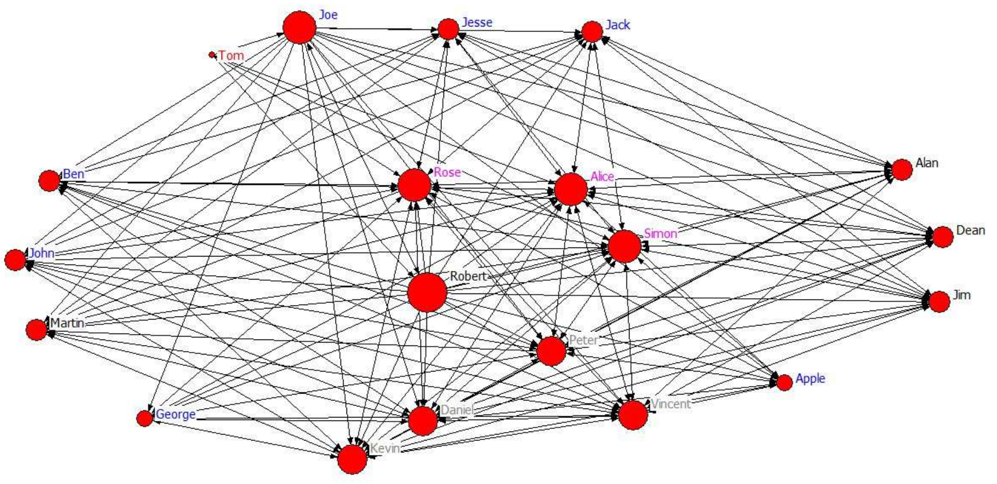


Figure 1. The Social Network Diagram before Rearrangement

Table 7. The Communication Matrix of Persons after Rearrangement

	Tom	Joe	Robert	Jesse	Jack	John	Ben	George	Apple	Martin	Alan	Dean	Jim	Daniel	Kevin	Peter	Vincent	Simon	Alice	Rose
Tom	#	1	1															1	1	1
Joe		#	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
Robert		1	#	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Jesse				#		1	1	1	1	1	1	1	■					1	1	1
Jack					#	1	1	1	1	■	■	■	1					1	1	1
John				1	1	#								1	1	1	1	1	1	1
Ben				1	1		#							1	1	1	1	1	1	1
George								#						1	1	1	1	1	1	1
Apple									#					1	1	1	1	1	1	1
Martin										#										
Alan				1	1						#			1	1	1	1	1	1	1
Dean				1	1							#		1	1	1	1	1	1	1
Jim				1	1								#	1	1	■	■	1	1	1
Daniel						1	1	1	1	1	1	1	1	#		1	1	1	1	1
Kevin						1	1	1	1	1	1	1	1		#	1	1	1	1	1
Peter						1	1	1	1	1	1	1	1	1	1	#		1	1	1
Vincent						1	1	1	1	1	1	1	1	1	1		#	1	1	1
Simon				1	1	1	1	1	1	1	1	1	1	1	1	1	1	#		
Alice	1			1	1	1	1	1	1	1	1	1	1	1	1	1	1		#	
Rose				1	1	1	1	1	1	1	1	1	1	1	1	1	1			#

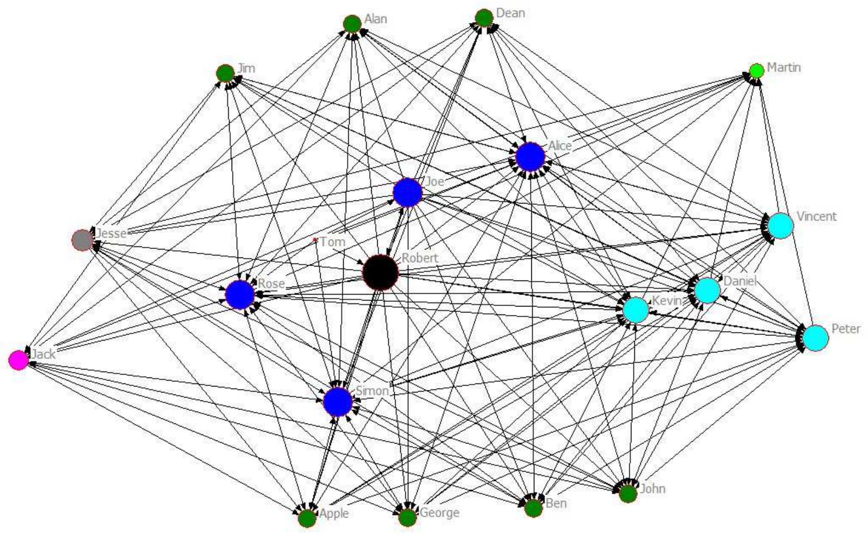


Figure 2. The Social Network Diagram after Rearrangement

We cannot see big changes to the social network diagram since the change is not very prominent. However, from the matrix, we can see that we added 4 connections which are red "1"s, the 6 grids with blue background are the eliminated connections. While the social network cannot embody the improvement totally since it doesn't show the frequencies of communications. It can, however, see that the Task14 and Task15 need communication to occur weekly. We rearranged these tasks to the same people, Peter and Vincent, which eliminated the weekly communication. In a similar fashion to the pervious reassignment, we rearranged Daniel and Kevin (Task13 and Task 16) in order to eliminate the weekly communication between these two tasks.

Discussion

As demonstrated in the table 1, in software development most of the information is communicated weekly and monthly. Around half of said communication is feedback, i.e. blue characters, since they are above of the diagonal. We learn from the position of the weekly and monthly communications that the associated people and tasks have not been well organized. This culminates to a software development project that is very vulnerable to wasting time and money due to rework and wasted communications. As stated by [10], gaining awareness of existing information exchange routes, information providers can act on information opportunities and make changes to information routes to improve the delivery of information services. Upon analysis, we discovered that they were opportunities to optimize the software development in the development state. Opportunities to optimize software development are mostly available in development stage including all programming tasks, Task4 through Task16 in this case, as other tasks in other stages comprise preparatory, design, and test and are inherently independent and cannot be assigned to other teams. Majority of the information fed forward to other tasks resides in the top of the DSM without any change (to include Task 17, test phase). Therefore, Tasks 1, Task2, and Task3 are considered a chunk.

In this case, the main reason that the programming tasks have the potential to get optimized is that Datashine is a small company; they have to assign more than one subsystem to each programmer. Since managers may have not assigned the subsystems well based on their technical and organizational aspects, DSM can help to improve this assignment. Optimized DSM showed that reassigning some subsystems can lead to better information flow and less feedback in the CRM Department.

Recommendation

Datashine Company as a software company must be cautious about its new product development projects to not being too long. Optimization information in the projects by rearrangement people is a vital solution but not enough. Datashine can make information flow more efficient in software development projects by:

- ✓ Preparing more efficient feedbacks
- ✓ classifying feedback to *ordinary*, *critical*
- ✓ using prior lesson to recognize critical feedbacks to focus on them
- ✓ accelerating critical feedbacks (monthly to weekly, or even weekly to daily)
- ✓ Preparing feedbacks among projects to transfer experiences and lessons to the others
- ✓ Capturing feedbacks and organizing them to use for future projects by applying

knowledge management techniques

Additionally, Datashine could utilize a skill-based allocation in conjunction with the DSM presented above. As described previously, human resource allocation or staffing, i.e., given a group of available developers and a set of project activities, which developers to activity allocation yields more value to the organization is a major problem in software development [11]. Reassignment of tasks based on the skill of the developer is a wise financial decision as it will lead to less project rework due to error and as a result save the company resource [11]. This skill-based reallocation in conjunction with the DSM suggested previously could be a viable solution to Datashine.

Conclusion

Software development is potentially vulnerable for wasting time and money due to reworks made in development stage. Most of feedbacks come from development stage and some from test stage. DSM matrix as a powerful tool can help to improve new product development process such as software development process. DSM focuses on information flow but relation among people is another critical aspect which can be improved by rearrangement of people to tasks. Social network analysis (SNA) can be applied to visualize the improvement made by DSM and task reassignment.

References

[1]: F.P. Brooks, Jr. The Mythical Man-Month. Addison-Wesley Publishing Company, Menlo Park, CA, 1975.

[2]: B.W. Boehm and P.N. Papaccio, "Understanding and Controlling Software Costs," IEEE Transactions on Software Engineering, Vol 14, No 10, October 1988, pp. 1462 - 1477.

[3]: R.E. Brooks. "Studying Programmer Behavior Experimentally: The Problems of Proper Methodology, "Communications of the ACM, Vol 23, No 4, April 1980, pp. 207 - 213.

[4]: J. Vosburgh, B. Curtis, R. Wolverton, B Albert, H. Malec, S. Hoben and Y. Liu,"Productivity Factors and Programming Environments," Proceedings of the 7th International Conference on Software Engineering , Washington D.C. IEEE Computer Society, 1984, pp.143 - 152.

[5]: F.P. Brooks, Jr. The Mythical Man-Month. Addison-Wesley Publishing Company, Menlo Park, CA, 1975.

[6]: Ali A. Yassine, "An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method", Product Development Research Laboratory, University of Illinois at Urbana-Champaign Urbana, IL. 61801

[7]: LIU Hongli, FU Yao, CHEN Zhigao, "Effects of Social Network on Knowledge Transfer within R&D Team", 2009 International Conference on Information Management, Innovation Management and Industrial Engineering, 978-0-7695-3876-1/09 © 2009 IEEE, DOI 10.1109/ICIII.2009.348

[8]: Tyson R. Browning, "Applying the Design Structure Matrix to System Decomposition and Integration Problems:A Review and New Directions", IEEE TRANSACTIONS ON ENGINEERING

MANAGEMENT, VOL. 48, NO. 3, AUGUST 2001

[9]: Brendan G. Cain, James O. Coplien and Neil B. Harrison, "Social patterns in productive software development organizations", Annals of Software Engineering Volume 2, Number 1 (1996), 259-286, DOI: 10.1007/BF02063813

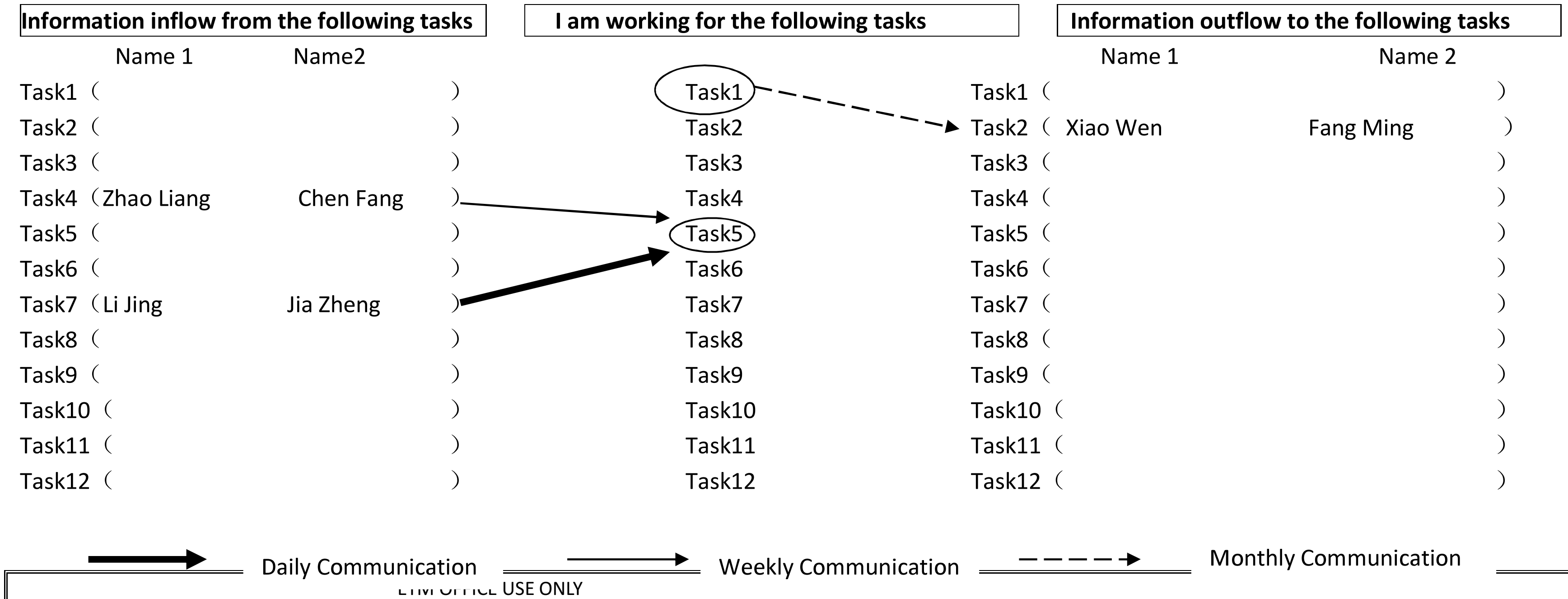
[10]: Caroline Haythornthwaite, “Social Network Analysis: An approach and Technique for the Study of Information Exchange,”Social Information Science Research, Volume 18, Issue 4, Autumn 1996, Pages 323-342, ISSN 0740-8188, 10.1016/S0740-8188(96)90003-1. (http://www.sciencedirect.com/science/article/pii/S0740818896900031)

[11]: Tad Gonsalves and Kiyoshi Itoh, “Multi-Objectives Optimization for Software Development Projects,” Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 Vol. 1, IMECS 2010, March 17-19, 2010

ETM
ENGINEERING & TECHNOLOGY MANAGEMENT

Appendix 1. Questionnaire Sample for DSM and SNA

Name: Ping Wang * Each person of the Software Department should fill out the form
Unit: Software Department



Information inflow from the following tasks	Name 1	Name2		**I am working for the following tasks**	**Information outflow to the following tasks**	Name 1	Name 2	
Task1 (			)	[✓ Task1]	Task1 (			)
Task2 (			)	Task2	Task2 (	Xiao Wen	Fang Ming	)
Task3 (			)	Task3	Task3 (			)
Task4 (	Zhao Liang	Chen Fang	)	Task4	Task4 (			)
Task5 (			)	[✓ Task5]	Task5 (			)
Task6 (			)	Task6	Task6 (			)
Task7 (	Li Jing	Jia Zheng	)	Task7	Task7 (			)
Task8 (			)	Task8	Task8 (			)
Task9 (			)	Task9	Task9 (			)
Task10 (			)	Task10	Task10 (			)
Task11 (			)	Task11	Task11 (			)
Task12 (			)	Task12	Task12 (			)

━━▶ Daily Communication ──▶ Weekly Communication - - - -▶ Monthly Communication

ETM OFFICE USE ONLY

Report No.:

Type: Student Project

Note:

Guidelines：

Please circle the tasks that you are working for in the middle column.

Label the tasks that are giving input to your tasks with incoming arrows, list out the person’s name you are contacting at the left column.

Label the tasks that your tasks are giving output to with outing arrows, list out the person’s name you are contacting at the right column.