ICED13
INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED13
19-22 AUGUST 2013, SUNGKYUNKWAN UNIVERSITY, SEOUL, KOREA

UNDERSTANDING TASK STRUCTURE IN DSM: MINING DEPENDENCY USING PROCESS EVENT LOGS

Lijun LAN, Ying LIU, Han Tong LOH
National University of Singapore, Singapore

ABSTRACT
Dependency Structure Matrix (DSM) has been widely used to analyze and present the inter-structure of design projects which are often characterized by many interrelated tasks. One essential role of DSM is to reveal the dependencies amongst different tasks. To succeed, it heavily relies on how well the initial dependencies are identified. Conventionally, it is accomplished by engineers through interview, survey and discussion that are obviously constrained by resource available in performing such activities. Hence, its liability heavily relies on the understanding of engineers. Moreover, for a complex design project where a higher number of tasks are involved, to gain a comprehensive understanding of their intricate relationships is also a non-trivial task. In order to tackle this, in this paper, we propose a flexible approach to mine dependency from process event logs which dynamically record the detailed information of task execution in a real context. Using process event logs, a number of dependencies can be steadily discovered and derived by focusing on different subsets of data for specific purposes. In the end, a case study is used to illustrate the proposed approach.

Keywords: design process, design informatics, decision making, information management

Contact:
Dr. Ying Liu
National University of Singapore
Mechanical Engineering
Singapore
117576
Singapore
mpeliuy@nus.edu.sg

1 INTRODUCTION

The methodology that is used to represent and analyze the dependency within an individual system is widely known as the Design Structure Matrix (DSM) or Dependency Structure Matrix (DSM). It has been well developed to help engineers to understand inter-structure of design projects. Until now, it has been applied in a variety of areas such as engineering change management, projects scheduling and complexity management, and so on. Along the way, it has been proved to be a very helpful tool to help engineers to understand inter-structure of design projects.

In DSM approaches, the dependencies amongst items (e.g., task, performer, component and so on) are presented in several matrixes that constitute the basis for subsequent analysis for structure understanding. Therefore, one crucial issue of DSM approaches is how well the dependencies can be identified. If the initial dependencies aren't correctly identified, the later analysis on the project's structure will be restricted. Conventionally, the identification of initial dependencies is carried out through interviews, surveys or discussion based on one group or several groups of engineers related to the given project. In this context, the reliability of the initial dependencies heavily relies on how well the relationships amongst items can be understood by engineers. However, in real complex processes, the multitudes of items (e.g., development-related tasks and activities, components in the product architecture, and performers involved in the process) and the intricate dependencies amongst them make it hard for engineers to have a complete, detailed view of the whole system. For example, a chief engineer in a project may have a broad understanding of the whole project but may know little about details of each part (Clarkson et al., 2004). In this case, dependencies are identified at a top level, which ignores the detailed information transacted at the low level of the project. In contrast, some engineers may understand their own work in detail but know little about their colleagues' works which have no direct interaction with their own works (Clarkson et al., 2004). Under this circumstance, engineers with different experience and knowledge understand the same project differently. Furthermore, the dependencies constructed by conventional ways mainly focus on particular problems and is not general for all purposes. For instance, a new series of interviews or surveys may be carried out when a new problem emerges, which is time-consuming and trivial.

Besides of interviews and surveys, the event log provides another way to discover the dependency amongst items within a specific process. During the past years, an increasingly number of operational processes are aided by the IT systems to promote the productivity. In this context, the event log dynamically and detailedly records the transaction events of the task executions in real processes. Furthermore, all kinds of information related to task executions are well recorded in logs as well, such as resource (e.g., person, device) executing the tasks, the timestamp when the tasks are executed, or information handled by the tasks (e.g., the size of the order, the cost of material). Such information can be utilized to discover the dependency amongst tasks at the bottom level. Furthermore, it is more objective and reliable, since the event logs are the actual records of a large quantity of task executions. However, this kind of data has not been fully utilized to discover the inter-structure, since most of the researches based on event logs mainly focus on the workflow perspective.

The research in this paper aims to present the dependency amongst tasks with a more flexible and objective way by mining initial dependencies from process event logs. By this way, the dependency between tasks can be constructed from the bottom up, combining the dynamical information in a long period. Different kinds of dependencies (i.e. the dependency in terms of workflow perspective, resource perspective and information perspective) can be extracted through exploring on different subsets of data in logs. Based on these initial dependencies, further computation and analysis can be carried out to support persons to understand the inter-structure of the entire process even they don't have the overview knowledge about the real process.

The rest of this paper is organized as follows. Section 2 reviews existing studies on DSM, and event mining. Section 3 illustrates the proposed approach for mining dependency from process event logs. Section 4 elaborates our approach based on a real case of a customer review process for Quality Function Deployment. Conclusions and future work are given in Section 5.

2 RELATED WORK

Dependency representation models play as useful tools when using past knowledge to support the design or redesign of new products and processes, because of its ability to visualize the inter-structure of the project. In general, the dependency representation models can be classified as matrix-based or

graph-based models. The DSM approach aims to represent the linkages between items, using matrix-based models. In contrast, the event mining approach aims to visualize the whole system through graph models with main focus on workflow perspective. In this section, we will generally introduce the related works of DSM approaches and event mining approaches.

2.1 Dependency Structure Matrix (DSM)

Research on DSM has come a long way. The first published formulation of DSM (Steward, 1981) is presented to model and analyze dependencies of one single type of item within one single domain. Since its first glance, a whole scientific community has developed to leverage its advantages. Along the way, DSM has been extended to Domain Mapping Matrix (DMM) (Danilovic and Sandkull, 2005) and Multiple Domain Mapping (Kortler et al., 2011, Lindemann et al., 2009) one after another. The goal of these extending formulations is to enable matrix-based models to include dependencies not only just within one domain at a time but also to allow for the relationships between two domains or even multiple domains.

Besides, the application domain of DSM also has expanded, such as Engineering Change Management (Clarkson et al., 2004, Keller et al., 2006, Koh et al., 2012, Tang, 2008), Project Scheduling (Chun-Hsien et al., 2003, Yanjun et al., 2011, Lin et al., 2012, Shi and Blomquist, 2012), Complexity Management (Li, 2011, Li and Mirhosseini, 2012, Xu et al., 2006) and so on.

- Engineering Change Management – it aims to control the change propagation from one part to other parts (Jarratt et al., 2005). The likelihood and impact of changes are computed to predict the change propagation (Clarkson et al., 2004).
- Project Scheduling – it provides a way to plan the execution sequence of many interrelated activities so as to minimize development duration of product. For example, as an extension of traditional DSM-based scheduling, Shi (2012) proposed an approach to utilize fuzzy set theory to address the problem that caused by the overlap of activities.
- Complexity Management – it supports to manage the complexity arising from the numerous elements and their multitudes of relationships within projects. For this application, partition is a main approach. For example, the large scale DSM-based design problem is decomposed into several subsystems to abate the complexity (Yanjun et al., 2011).

As mentioned in Section 1, the conventional ways to identify initial dependency mainly includes interviews, surveys and discussions. For example, in Clarkon's study based on a Westland Helicopters of rotorcraft design (Clarkson et al., 2004), 17 senior engineers and 5 chief engineers are interviewed for change propagation information, followed by 7 senior engineers conferring together to identify the final dependencies between items. For such a complex design project where various items are involved, to gain the dependency amongst items is a non-trivial task. Furthermore, it was also reported that the understandings of engineers differ with their own knowledge and experience.

2.2 Event Mining

As the operational processes are increasing aided by IT systems, event mining has emerged as a new direction to discover process structure from event logs in term of workflow perspective. The dependencies amongst tasks are ultimately visualized in the formalism of graph models (e.g., Petri net, event graph) (Rozinat et al., 2009, Zhang et al., 2010), based on which simulation works are carried out to support decision making. Based on the Petri net, various algorithms (van der Aalst et al., 2004, van der Aalst et al., 2003, Rozinat et al., 2009, Wen et al., 2009) have been reported. On the other hand, the application of event graph generates another way (Zhang et al., 2010, Ying et al., 2012) to discover and present process structure. In this case, an event graph model is obtained and within this type of graph model, the vertexes correspond to the well-defined tasks, the variables affiliated to each vertex indicate the information handled by the corresponding task, and the edges present the transfer conditions between tasks. The both models mainly visualize the inter-structure of the processes from workflow perspective.

Through literature review, we note that although many existing studies have made significant advancements in dependency representation for process understanding, significant limitations still exist. On the one hand, in most of existing works on DSM, the initial dependencies amongst items are generally constructed from a top-level view through interviews or surveys, which is time-consuming and trivial. Moreover, the liability of dependency heavily depends on the understanding of engineers. On the other hand, it lacks a full use of information recorded in historical data, since most of existing

approaches of process mining mainly focus on workflow aspect. In order to address the two problems, we propose to automatically mine dependency from the process event logs, which is flexible to help engineers to understand task structure from different aspects for different purposes. Furthermore, the derived dependency is objective and convincing as the interference of human factor has been reduced.

3 A DEPENDENCY MINING FRAMEWORK BASED ON EVENT LOG

In order to understand task structure, the focus of our research is to mine dependency from event logs automatically. The proposed framework is shown in Figure 1. Any real process aided by IT systems will provide task execution information in some forms (e.g., event log, email) and to some extent. An event log is a collection transaction events of the task executions, including information about resources, timestamp, the data dealt with. In a given event log, a case is an instance of process, which consists of a set of sequentially recorded events. It is assumed that each event refers to an activity that is a well-defined task in the process and is related to a particular case (Maruster et al., 2002). In our study, we aim to mining dependency of tasks from different aspects (i.e., workflow aspect, resource aspect and data aspect) by exploring on different subsets of data recorded in event logs. These dependencies are presented in initial dependency matrixes. Based on them, the DSM-based algorithms are applied to compute the derived dependency matrixes for different purposes. Finally, the analytical results based on derived dependency matrix are used for process improvement.

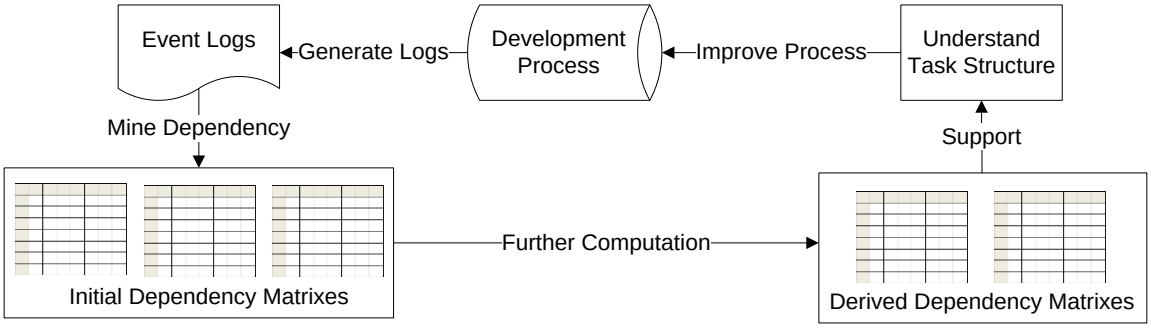

Figure 1. Framework for dependency mining based on event logs

3.1 Mine Dependency from Event Logs

The aim of this step is to extract initial information from event logs directly. In this step, four initial dependency matrixes are automatically constructed by focusing on different subsets of data recorded in log files. They are respectively named as distance dependency matrix, data dependency matrix, text similarity matrix, and performer dependency matrix, reflecting the interconnectivities amongst tasks from workflow aspect, resource aspect and data aspect. Subsequently, these basic matrixes can be used for further computation so as to present the task structure for different problems.

3.1.1 Construct Distance Dependency Matrix

A process consists of a sequence of distinct workflow tasks. Let T be a set of tasks $T = \{t_1, t_2, \ldots, t_N\}$. N denotes the number of tasks. In this context, we define distance relation.

Definition Distance Relation $t_i \rightarrow_k t_j$: t_j is directly or indirectly recorded after t_i in event logs with intervals of k tasks.

In order to evaluate on what degree a task is carried out after another, the distance dependency matrix (noted as DIS) is constructed following Equation 1.

$$DIS_{i,j} = W_D * \left(\frac{V(t_i \rightarrow_0 t_j)}{2V(t_i)} + \frac{V(t_i \rightarrow_0 t_j)}{2V(t_j)}\right) + W_I * \left(\frac{V(t_i \rightarrow_1 t_j)}{2V(t_i)} + \frac{V(t_i \rightarrow_1 t_j)}{2V(t_j)}\right)$$

$$\text{where, } 0 \leq W_D, W_I \leq 1 \text{ and } W_D + W_I = 1 \quad (1)$$

Here, $V(t_i \rightarrow_k t_j)$ is the counter of $t_i \rightarrow_k t_j$ recorded in logs and $V(t_k)$ is the frequency of t_k. The distance dependency between two tasks is calculated from two parts: direct contribution and indirect contribution. Meanwhile, W_D and W_I are the weights of the two parts. The sum of W_D and W_I equals 1, so that $DIS_{i,j}$ can fall in the range of 0 to 1.

In event logs, events are sequentially organized according to the orders of tasks in real process. Therefore, tasks recorded closely may have high dependency in real executions, which is represented by the direct contribution of equation 1. However, event logs may present wrong message when two or even more tasks are executed concurrently following the same task. For example, when task B and C are both carried out after task A concurrently, the event sequence corresponding to the three tasks may be recorded as A→B→C or A→C→B randomly. The order of B and C depends on which is executed firstly. In order to take this situation into account, the distance dependency is computed by combining both direct dependency and indirect dependency with weights W_D and W_I. In a simple project with little or no concurrency, W_D could be assigned a bigger value than W_I. Therefore, DIS could be constructed with more focus on direct contribution. On the contrary, in a complex project, concurrency should be carefully considered. In this case, the value of W_I should be increased to take the indirect following relationships into account.

3.1.2 Construct Data Dependency Matrix

In real process, tasks with strong interconnectivity are assigned to deal with a large amount of common information. In term of event log, it can be interpreted as two events share many of their data. The strength of data sharing can be computed through dividing the number of common data by the total number of the data belonging to the two events, as shown in Equation 2.

$$DATA_{i,j} = D_{t_i} \cap D_{t_j} / 2 * N_{t_i} + D_{t_i} \cap D_{t_j} / 2 * N_{t_j} \quad (2)$$

In Equation 2, D_{t_i} and D_{t_j} are the sets of data recorded accompanying tasks t_i and t_j in event logs. The express $D_{t_i} \cap D_{t_j}$ indicates the number of common data dealt with by the two tasks. N_{t_i} and N_{t_j} denote the data numbers of tasks t_i and t_j correspondingly. Therefore, the data dependency matrix (noted as $DATA$) is constructed by calculating the data dependency of each two tasks. The more data are shared between two tasks, the larger the data dependency strength is.

3.1.3 Construct Text Similarity Matrix

Generally, events are recorded in event logs with their job features included so as to organize events well. As a result, closely related tasks are likely to be described using similar names. For example, two tasks assigned to extract product feature for two kinds of products: *ProductA* and *ProductB* may be respectively named as *ExtractFeaturesForProductA* and *ExtractFeaturesForProductB*. In this context, the text similarity is also considered as an important factor to explore the relationship between tasks. Simply, text similarity can be computed by measuring how similarly two events are descripted in event logs.

3.1.4 Construct Performer Dependency Matrix

The performer dependency is identified in term of resource perspective. In real processes, one staff may operate different tasks. It is commonly considered that two tasks executed by the same person have high interconnection, because personal knowledge is specific and limited. Furthermore, when events are recorded in event logs, performers involved in highly related tasks are generally described using similar names with their job features included. For example, *AA Extractor* and *BB Extractor* may be used to express the performers of two similar or interconnected tasks. Therefore, performer dependency matrix can be constructed based on the similarity of corresponding persons' description in the logs.

3.2 Compute Derived Dependency Matrixes for Task Structure Understanding

In real applications, engineers generally focus on some specific problems which require them to understand the task structure from different aspects. For example, the execution sequence of some tasks may be rescheduled in order to reduce the time consumption of the entire process. In this case, engineers more concern about the workflow between two tasks. Taking collaboration improvement as another example, the transacted data are more interesting for engineers. In our study, three derived dependency matrixes representing task structure from different aspects are proposed as follows.

3.2.1 Compute Causal Dependency Matrix

Causal dependency, here, means that a given task must be executed after another task. In real applications, tasks with strong causal dependency are executed closely, which is reflected in the event

logs by events recorded one after another. This is particularly the truth when the entire process is relatively simple and the tasks aren't done concurrently. However, the observed event sequence differs from the actual causal dependency relationship among events when concurrence exists. In this context, the data dependency should be taken into account to discover the real workflow, because a subsequent task may need the outputs of the previous task as its inputs in reality. Hence, the causal dependency exists between tasks t_i and $t_j$with high distance dependency $DIS_{i,j}$ or high data dependency $DATA_{i,j}$, as shown in Equation 3. Additionally, we consider distance dependency plays a more important role than data dependency in determining the causal dependency. As a result, W_{data} which is less than 1 is introduced to reduce the influence of $DATA$.

$$CAU_{i,j} = \max\left(DIS_{i,j}, W_{data} * DATA_{i,j}\right), \quad 0 \le W_{data} \le 1 \tag{3}$$

3.2.2 Compute Communication Matrix

The dependencies presented in the communication matrix are measured in term of the information perspective to discover how tasks collaborate with each other. The events corresponding to highly collaborative tasks share a large quantity of data attributes and are likely to be expressed in similar ways in logs. Therefore, the communication matrix is constructed as the weighted summation of three initial matrixes, i.e., data dependency matrix $DATA$, text similarity matrix $TEXT$ and performer dependency matrix PER, as shown in Equation 4.

$$\begin{aligned} &COM = W_D * DATA + W_T * TEXT + W_P * PER, \\ &where,\ 0 \le W_D, W_T, W_P \le 1\ and\ W_D + W_T + W_P = 1 \end{aligned} \tag{4}$$

Based on dependency matrixes, partition and cluster are two prevailing ways to classify items into different groups with high interconnectivity within a group and low connectivity between groups. By this way, it is easy to observe the inter structure of tasks through visualizing which tasks interrelate to each other more closely than others. In our approach, a tree-based partition algorithm (Romesburg, 1990) is applied to group tasks based on the linear combination of three initial matrixes: data dependency matrix, text similarity matrix and performer dependency matrix. Generally, tasks in a larger group are more sensitive and influencing and should be paid more attention when managers are trying to understanding the task structure or making decision to improve the process.

3.2.3 Compute Significance Matrix

The influence of a given task to the entire process is evaluated from two perspectives: workflow perspective and data perspective, based on the causal matrix and communication matrix. On the one hand, a task plays a more important role in identifying the entire process structure when a higher number of successive tasks depend on it or itself depends on many other tasks simultaneously. On the other hand, based on the communication matrix, a task has huge impact on the performance of the entire process when it interconnects with many other tasks simultaneously. Under the both situations, a change to one task will influence the performances of many other tasks. As a result, the both kinds of tasks play a significant role in identifying the process structure.

In our study, for a given task t_i, the significance strength from workflow aspect is evaluated through computing the sum of corresponding row and column in the causal dependency matrix (noted as CAU), as shown in Equation 4. Similarly, based on the communication matrix (noted as COM), the significance strength from data aspect is calculated using Equation 5.

$$L_{I,I} = \left(\sum_{j=1}^{j=N} CAUu_{I,j} + \sum_{j=1}^{j=N} CAU_{j,I}\right) / \sum_{i=1}^{i=N} \sum_{j=1}^{j=N} CAU_{i,j} \tag{5}$$

$$W_{I,I} = \left(\sum_{j=1}^{j=N} COM_{I,j} + \sum_{j=1}^{j=N} COM_{j,I}\right) / \sum_{i=1}^{i=N} \sum_{j=1}^{j=N} COM_{i,j} \tag{6}$$

In Equation 4 and Equation 5, *N* is the total number of tasks involved in a process.

Ultimately, the two kinds of significances are visualized through the rectangle along the diagonal of the significance matrix. Hence, the comprehensive significance of each task can be obviously judged via comparing the area of each rectangle. At the same time, the length of every rectangle indicates the significance of corresponding task in term of workflow perspective and the width of per rectangle denotes the significance in term of data perspective. Besides, the collaboration relationships amongst tasks are also presented by the groups highlighted in significance matrix. In general, the group

involving more tasks and with more big rectangles should be focused on to a large extent before new measures are carried out, since these parts may more intricate than other parts.

4 CASE STUDY

4.1 Experiment Setup

This is a real case of a customer review process, where we extract features from product comments and collect them for Quality Function Deployment (QFD) (Yun and Zhongchun, 2009), as shown in Figure 2. When a comment is received, a person with the role of Distributor checks the product type and delivers it to corresponding extractors. There are three types of product: *ProductA* and *ProductB* for self-owned products and Comparison for competitive products. The extractors review the comments and extract the features of the products. After that, the comments and the extracted feature lists are passed on to the corresponding verifiers, who are in charge of verifying them and transfer them to the Collector. For the type named Comparison, it is delivered to collector directly for data gathering without verification. The whole process lasted for nine days and finally generated 997 cases in total. Each time a person starts or completes a case there is a record in the log file.

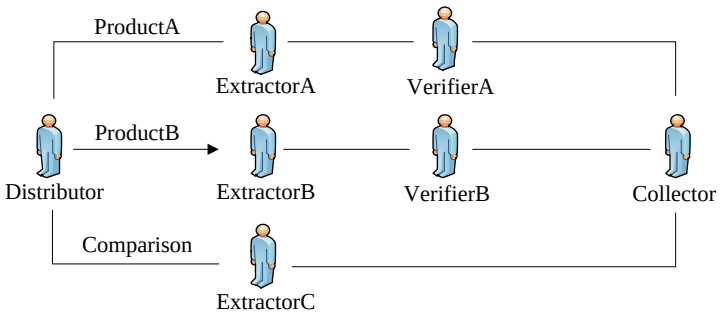

Figure 2. Workflow of the customer review process

4.2 Results and Discussions

We realized the proposed method on the Java platform and tested it based on the customer review process. The initial dependency matrixes and derived dependency matrixes are illustrated in Figure 3 and Figure 4 respectively. In this study, the real customer review process is relatively simple. Furthermore, the workflow and the role of each performer are known in advance. Therefore, the results of the proposed method were assessed by whether they captured the main structure features of original process.

4.2.1 Initial Dependency Matrixes

Figure 4 illustrates the four initial dependency matrixes which are directly mined and constructed from event logs. As can be seen, seven types of tasks and seven operators are totally discovered throughout the logs. They are given in the lists at the right and the top of the dependency matrixes. Within each matrix, the decimals out of the diagonal indicate dependency strength that the task heading in the row interacts with the corresponding column task. Particularly, the performer dependency matrix in Figure 4(d) reflects the relationships across two domains, performer domain and task domain.

4.2.2 Derived Dependency Matrixes

In our study, as shown in Figure 4, three derived dependency matrixes are further computed based on the initial dependency matrixes, so as to reflect task structure from different aspects.

- Workflow perspective - As illustrated in Figure 4(a), the causal dependency matrix reflects the dependency strength that the execution of a task listed ahead the row depends on the corresponding column task. Except the cells in the diagonal, cells with big values indicate that strong dependency relationships exist between corresponding tasks. According to the data in Figure 4(a), eight pairs of tasks are found with big causal dependency, i.e., *CT→EA*, *CT→EB*, *CT→EC*, *EA→VA*, *EB→VB*, *VA→CF*, *VB→CF* and *EC→CF*. Furthermore, comparing all the rows and columns, we found that three tasks directly follows task *CT*, which implies that *CT* is a splitting point. Similarly, the task *CF* is found as a join point. Both kinds of tasks like *CT* and *CF*

are important for task structure understanding in real applications, since they structurally identify the architecture of the entire process. In addition, the workflow dependency reflected in causal dependency matrix is consistent with the real workflow shown in Figure 2.

	CT	EB	VB	CF	EA	VA	EC
CT	0	0.479	0.205	0.092	0.463	0.199	0.457
EB	0	0	0.699	0.205	0	0	0
VB	0	0	0	0.479	0	0	0
CF	0	0	0	0	0	0	0
EA	0	0	0	0.199	0	0.700	0
VA	0	0	0	0.464	0	0	0
EC	0	0	0	0.458	0	0	0

(a) Distance Dependency Matrix

	CT	EB	VB	CF	EA	VA	EC
CT	1.0	0	0	0	0	0	0
EB	0	1.0	0	0	0	0	0
VB	0	0	1.0	0	0	1.0	0
CF	0	0	0	1.0	0	0	0
EA	0	0	0	0	1.0	0	0
VA	0	0	1.0	0	0	1.0	0
EC	0	0	0	0	0	0	1.0

(b) Data Dependency Matrix

	CT	EB	VB	CF	EA	VA	EC
CT	1.0	0.075	0.070	0.153	0.074	0.070	0.073
EB	0.075	1.0	0.645	0.321	0.962	0.611	0.668
VB	0.070	0.645	1.0	0.604	0.611	0.971	0.358
CF	0.153	0.321	0.604	1.0	0.321	0.604	0.310
EA	0.075	0.962	0.611	0.321	1.0	0.645	0.668
VA	0.070	0.611	0.971	0.604	0.645	1.0	0.358
EC	0.073	0.668	0.358	0.310	0.668	0.358	1.0

(c) Texture Similarity Matrix

	CT	EB	VB	CF	EA	VA	EC	TC	BE	BV	FC	AE	AV	CE
CheckType(CT)	1.0	0.083	0.174	0.146	0.083	0.174	0.083	1	0	0	0	0	0	0
ExtractFeaturesForProductB(EB)	0.083	1.0	0.261	0.292	0.833	0.087	0.833	0	1	0	0	0	0	0
VerifyExtractedFeaturesForProductB(VB)	0.174	0.261	1.0	0.077	0.087	0.818	0.087	0	0	1	0	0	0	0
CollectExtractedFeatures(CF)	0.146	0.291	0.077	1.0	0.291	0.076	0.291	0	0	0	1	0	0	0
ExtractFeaturesForProductA(EA)	0.083	0.083	0.087	0.291	1.0	0.261	0.833	0	0	0	0	1	0	0
VerifyExtractedFeaturesForProductA(VA)	0.174	0.087	0.818	0.077	0.261	1.0	0.087	0	0	0	0	0	1	0
ExtractFeaturesForComparison(EC)	0.083	0.833	0.087	0.291	0.833	0.087	1.0	0	0	0	0	0	0	1
Type Checker(TC)	1	0	0	0	0	0	0							
BB Extractor(BE)	0	1	0	0	0	0	0							
BB Verifier(BV)	0	0	1	0	0	0	0							
ForAll Collector(FC)	0	0	0	1	0	0	0							
AA Extractor(AE)	0	0	0	0	1	0	0							
AA Verifier(AV)	0	0	0	0	0	1	0							
CC Extractor(CE)	0	0	0	0	0	0	1							

(d) Performer Dependency Matrix

Figure 3. Initial matrixes constructed from event logs directly

	CT	EB	VB	CF	EA	VA	EC
CT	0	0.479	0.205	0.092	0.463	0.199	0.457
EB	0	0	0.699	0.205	0	0	0
VB	0	0	0	0.479	0	0.300	0
CF	0	0	0	0	0	0	0
EA	0	0	0	0.199	0	0.700	0
VA	0	0	0.3	0.464	0	0	0
EC	0	0	0	0.458	0	0	0

(a) Causal Dependency Matrix

	CT	VA	VB	EA	EB	EC	CF
CT	1.000	0.073	0.073	0.047	0.047	0.047	0.090
VA	0.073	1.000	0.937	0.272	0.209	0.134	0.204
VB	0.073	0.937	1.000	0.209	0.271	0.134	0.204
EA	0.047	0.272	0.209	1.000	0.538	0.450	0.184
EB	0.047	0.209	0.271	0.538	1.000	0.450	0.184
EC	0.047	0.134	0.134	0.450	0.450	1.000	0.180
CF	0.090	0.204	0.204	0.184	0.184	0.180	1.000

(b) Communication Dependency Matrix

	CT	VA	VB	EA	EB	EC	CF
CheckType(CT)		0.073	0.073	0.047	0.047	0.047	0.090
VerifyExtractedFeaturesForProductA(VA)	0.073		0.937	0.272	0.209	0.134	0.204
VerifyExtractedFeaturesForProductB(VB)	0.073	0.937		0.209	0.271	0.134	0.204
ExtractFeaturesForProductA(EA)	0.047	0.272	0.209		0.538	0.450	0.184
ExtractFeaturesForProductB(EB)	0.047	0.209	0.271	0.538		0.450	0.184
ExtractFeaturesForComparison(EC)	0.047	0.134	0.134	0.450	0.450		0.180
CollectExtractedFeatures(CF)	0.090	0.204	0.204	0.184	0.184	0.180	

(c) Significance Matrix

Figure 4. Derived dependency matrixes

- Information perspective - Communication dependency matrix aims to evaluate the relevance of tasks from the information aspect. As highlighted in Figure 4(b), tasks are classified into four groups, which provide engineers with a direct view of the collaboration and interaction amongst tasks. For process improvement, measures should be carried out to achieve a balance between these high related tasks. According to the data in Figure 4(b), we found that tasks *VA* and *VB* are assigned similar or closely related jobs. This is the same situation for tasks *EA*, *EB*, and *EC*. The findings are identical to the real situation, as *VA* and *VB* are both assigned to verify the features extracted by extractors, and *EA*, *EB*, and *EC* are assigned to extract product features.
- Combination perspective - Sometimes, decisions must be figured out considering from both workflow aspect and information aspect. As shown in Figure 4(c), the significance matrix allows the inter-structure of tasks to be inspected from the both aspects. Specifically, a large group with many large rectangles included plays an important role in influencing the entire performance of process. According to this point, tasks *EA* and *EB* are found high related to each other, with large rectangles corresponding to the two tasks. The similar situation exists between tasks *VA* and *VB*. Besides, from the causal dependency matrix, we also found that task *VA* is executed after task *EA* and task *VB* is executed after task *EB*, as the result of the high causal dependences between each two tasks. In this context, it can be deduced that the two task sequences have been assigned highly connected and similar jobs. In order to reduce the resource cost, a further measure could be suggested by merging the jobs of the four performers.

Based on the above discussions, our approach can performs as a useful tool to present task structure from different aspects when the intricate and complex relationships amongst items hardly hinder engineers from understanding the inter-structure of the task. Comparing the results with the real process reveals that the final dependency matrixes could capture the main structure feature of the original process. In this study, the workflow of real process was known in advance. However, in reality, the real processes are beyond the scope of cognitive more or less. Under this circumstance, our approach is helpful to support process structure understanding by visualizing the dependency amongst tasks.

5 CONCLUSIONS

This paper is our preliminary investigation of mining dependency from execution logs. It is flexible and objective for presenting the inter-structure of the tasks to support analysis questions, since the detailed information of each task are recorded when they are executed. The dependency matrixes are constructed from event logs in terms of workflow perspective, information perspective and resource perspective, making full use of the data produced by tasks. Based on these dependency matrixes, different kinds of computation can be carried out to help engineers to understand task structure from different aspects. Subsequently, the approach is illustrated and evaluated base on a real case study and some further suggestions are derived based on the analysis. The experiment of the real case study shows that our approach is useful to mine and present inter-dependency of tasks and support managers to analyze and manage task structure for different purposes.

In our future work, the method will be evaluated based on more complex and specific cases. Also, there are still some rooms to improve and extend the current method for task structure understanding. For example, time factor can be taken into account, since the task structure in different periods may differ.

ACKNOWLEDGMENTS

The work described in this paper was supported by a research grant from the National University of Singapore (Grant No: R265-000-362-133).

REFERENCES

Chun-Hsien, C., Shih Fu, L. & Wei, C. (2003) Project scheduling for collaborative product development using DSM. *International Journal of Project Management,* Vol. 21, No. 4**,** pp. 291-9.

Clarkson, P. J., Simons, C. & Eckert, C. (2004) Predicting change propagation in complex design. *Transactions of the ASME. Journal of Mechanical Design,* Vol. 126, No. 5**,** pp. 788-97.

Danilovic, M. & Sandkull, B. (2005) The use of dependence structure matrix and domain mapping matrix in managing uncertainty in multiple project situations. *International Journal of Project Management,* Vol. 23, No. 3**,** pp. 193-203.

Jarratt, T., Clarkson, P. J. & Eckert, C. (2005) Engineering change. *In:* Clarkson, P. J. & Eckert, C. (eds.), *Design Process Improvement - A review of current practice*. New York: Springer London Heidelberg, pp. 262-285.
Keller, R., Eckert, C. M., Earl, C., et al. (2006) Supporting change processes in design: Complexity, prediction and reliability. *Reliability Engineering & System Safety,* Vol. 91, No. 12**,** pp. 1521-34.
Koh, E. C. Y., Caldwell, N. H. M. & Clarkson, P. J. (2012) A method to assess the effects of engineering change propagation. *Research in Engineering Design,* Vol. 23, No. 4**,** pp. 329-351.
Kortler, S., Helms, B., Shea, K., et al. (2011) A more flexible way of modeling structure with multiple domains *13th International Dependency and Structure Modelling Conference, DSM'11, September 14, 2011 - September 15, 2011*, Cambridge, MA, United states, of Conference: Carl Hanser Verlag GmbH and Co. KG, pp. 19-29.
Li, S. (2011) A matrix-based clustering approach for the decomposition of design problems. *Research in Engineering Design,* Vol. 22, No. 4**,** pp. 263-78.
Li, S. & Mirhosseini, M. (2012) A matrix-based modularization approach for supporting secure collaboration in parametric design. *Computers in Industry,* Vol. 63, No. 6**,** pp. 619-631.
Lin, J., Qian, Y., Yassine, A. A., et al. (2012) A fuzzy approach for sequencing interrelated activities in a DSM. *International Journal of Production Research,* Vol. 50, No. 23**,** pp. 7012-7025.
Lindemann, U., Maurer, M. & Braun, T. (2009) *Structural complexity management : an approach for the field of product design.* Berlin: Springer.
Maruster, L., Weijters, A. J. M. M. T., Aalst, W. M. P. W., et al. (2002) Process Mining: Discovering Direct Successors in Process Logs. *In:* Lange, S., Satoh, K. & Smith, C. (eds.), *Discovery Science*. Springer Berlin Heidelberg, pp. 364-373.
Romesburg, H. C. (1990) *Cluster analysis for researchers.* Malabar, Fla.: Robert E. Krieger Pub. Co.
Rozinat, A., Wynn, M. T., Van Der Aalst, W. M. P., et al. (2009) Workflow simulation for operational decision support. *Data & Knowledge Engineering,* Vol. 68, No. 9**,** pp. 834-50.
Shi, Q. & Blomquist, T. (2012) A new approach for project scheduling using fuzzy dependency structure matrix. *International Journal of Project Management,* Vol. 30, No. 4**,** pp. 503-510.
Steward, D. V. (1981) The design structure system: a method for managing the design of complex systems. *IEEE Transactions on Engineering Management,* Vol. EM-28, No. 3**,** pp. 71-4.
Tang, D. (2008) Product design knowledge management based on design structure matrix *2008 12th International Conference on Computer Supported Cooperative Work in Design, CSCWD, April 16, 2008 - April 18, 2008*, Xi'an, China, of Conference: Inst. of Elec. and Elec. Eng. Computer Society, pp. 246-251.
Van Der Aalst, W. M. P., Dongen, B. F. v., Herbst, J., et al. (2003) Workflow mining: a survey of issues and approaches. *Journal of Data & Knowledge Engineering,* Vol. 47, No. 2**,** pp. 237-267.
Van Der Aalst, W. M. P., Weijters, T. & Maruster, L. (2004) Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering,* Vol. 16, No.**,** pp. 1128-1142.
Wen, L., Wang, J., Van Der Aalst, W. M. P., et al. (2009) A novel approach for process mining based on event types. *Journal of Intelligent Information Systems* Vol. 32, No. 2**,** pp. 163-190.
Xu, X., Li, C., Yan, J., et al. (2006) An analytical method based on design structure matrix for modular identification *2006 7th International Conference on Computer-Aided Industrial Design and Conceptual Design, CAIDC, November 17, 2006 - November 19, 2006*, Hangzhou, China, of Conference: Inst. of Elec. and Elec. Eng. Computer Society.
Yanjun, Q., Jun, L., Thong Ngee, G., et al. (2011) A novel approach to DSM-based activity sequencing problem. *IEEE Transactions on Engineering Management,* Vol. 58, No. 4**,** pp. 688-705.
Ying, L., Hui, Z., Chunping, L., et al. (2012) Workflow simulation for operational decision support using event graph through process mining. *Decision Support Systems,* Vol. 52, No. 3**,** pp. 685-97.
Yun, H. & Zhongchun, M. (2009) Quality function deployment method and its application in engineering project performance evaluation *2009 International Conference on Electronic Commerce and Business Intelligence, ECBI, 6-7 June 2009*, Piscataway, NJ, USA, of Conference: IEEE, pp. 184-7.
Zhang, H., Liu, Y., Li, C., et al. (2010) Deriving Event Graphs through Process Mining for Runtime Change Management. *Modelling and Management of Engineering Processes,* Vol. No.**,** pp. 127-138.