IMPROVING PROJECT PERFORMANCE USING DEPENDENCY CYCLE EXTRACTION AND ANALYSIS

Matin Mavaddat
University of the West of England
Matin.Mavaddat@uwe.ac.uk

Stewart Green
University of the West of England
Stewart.Green@uwe.ac.uk

Jin Sa
University of the West of England
Jin.Sa@uwe.ac.uk

ABSTRACT

To fulfill the objectives of a software development project in an organisation, project participants create different kinds of dependencies on each other. These dependencies and their types reveal a lot of dependency information about processes that are being followed to complete a project: e.g. project team structure, culture and communication patterns. Extracting these dependencies, which represent fragments of business process execution, and helping the analysts to find the root causes of some of the organizational problems in the projects is a non-trivial task; and business analysts and consultants might need to spend a lot of time doing this, using conventional analysis methods. Using a new concept called dependency cycles, this paper introduces a method for extracting these dependencies from conversation logs, and shows how they can be analyzed in order to extract patterns that could help an organization to improve its structure, communication forms, culture and business processes.

KEYWORDS

Dependency cycles, business process modeling, business process optimisation, communication patterns

1. INTRODUCTION

Large project teams usually suffer from many organizational problems such as dysfunctional team structures (e.g. dysfunctional hierarchies), cultural differences that cause misunderstanding, bad organisational culture (e.g. lack of authority, cyclic referral of responsibilities) and the complexity of the communication patterns and processes that they follow in order to fulfill the objectives of the project. These problems cause the project teams to underachieve, prevent the project from achieving its business goals, and create a lot of inefficiencies (Hoegl, M., 2005). Using conventional methods, investigating the problem with the project team and finding the actual source of problem is a non-trivial task. Usually consultants and business analysts need to spend an extensive amount of time with the team using a variety of techniques, such as interviews, questionnaires and observational methods such as ethnography, to try to find the root causes of the problems (Cadle, J., Paul, D. & Turner, P., 2010).

The paper introduces the notion of dependency cycles between roles working on a project in order to tackle these problems. It also introduces three theories (see next section) and explains how they contribute to the dependency cycle approach. The paper goes on to present principles and heuristics for extracting dependency cycles from conversation logs, like email corpora, and discusses how analyses based upon such extractions can be used to appraise a project's processes, team structure and patterns of communication.

This technique of using conversation logs, such as email messages that have been sent and received about the project's tasks and goals and objectives, is aiming to reduce the amount of time necessary for analyzing

the project team dynamics, communication patterns, processes etc. and to reveal the problems more quickly and efficiently. It also does not suffer from ethnographic approaches' shortcomings such as:

- Heisenberg principle: People usually do not work in the same way when they are being observed or, more formally, the results of the observation are affected by the presence of the observer.
- Interrupt 'business-as-usual': this means someone's watching and asking questions will interrupt peoples' usual way of work and might decrease their performance.
- It is difficult to observe knowledge workers' work without asking many questions.
- Not observing the typical working day that is a good representative of a typical work pattern.

And, being based on more documented evidence, the proposed technique is less subjective than conventional analysis methods.

The design-science research method (Hevner, A., 2007) (Hevner, A. & Chatterjee, S., 2010), and, more specifically, Peffers' et. al.'s (2006) process model for design-science research has been used for carrying out this research. Their process model consists of the following steps:

1. Problem identification and motivation
2. Objectives of a solution
3. Design and development
4. Demonstration
5. Evaluation
6. Communication

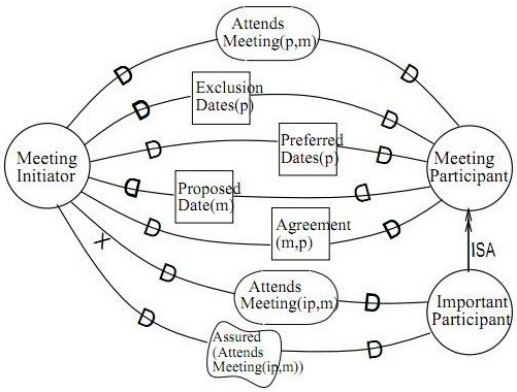
We have tried to cover the first and second steps – problem identification and motivation and objectives of a solution - in the introduction section of the paper by identifying the problem that we are trying to solve and the motivations, and also by defining in what aspects this new solution is going to supersede the previous solutions to this problem. In section 2 we will cover the third step of the process – design and development - by introducing the artifact of this research, i.e. the new technique, and demonstrating how it has been designed and developed based on sound theories. Section 3, using a case study and applying the proposed technique to it, will cover the fourth step of the process - demonstration. In section 4 the outcomes of the case study will be evaluated; which will cover the fifth step of the process - evaluation. And the sixth step – communication (publication) – is covered in part by this paper.

2. DESIGN AND DEVELOPMENT OF THE TECHNIQUE

2.1 Theoretical basis (strategic dependency models, conversation for action diagrams and episodic memory)

Dependencies between different business process participants (project team members) and the important role that it plays is not a new concept. Eric Yu (2010) in the i* framework, introduces a model called strategic dependency model (see figure 1). SD tries to capture strategic dependency of different business process participants to fulfil a business objective. In other words the model tries to capture how different business roles depend on each other to achieve a particular business goal. Eric Yu(2010) introduces four different dependency types: 1. Goal dependency 2. Task dependency 3. Resource dependency 4. Soft goal dependency. He argues that creating the SD model based on these dependency types significantly contributes to the understanding of the business processes of an organisation.

The below, very simple, strategic dependency model (figure 1) shows how different actors (agents) depend on each other in a meeting planning process. Each dependency consists of three main elements, depender, dependee and dependum. Depender is the person who depends on someone to achieve a goal, complete a task or deliver a resource. Dependee is the person who the depender has depended on and finally dependum is the subject of the dependency, the "thing" that the depender has depended on the dependee to deliver (task, goal or resource). For example, the figure 1 diagram shows that the meeting initiator (depender) has a goal dependency on the meeting participant (dependee) to attend the meeting (dependum). He or she also has some resource dependency on the meeting participant to provide him or her with exclusion dates and preferred dates. The meeting participant depends (resource dependency) on the initiator to propose a date; and, eventually, the initiator depends on the participant to agree on a date and time.



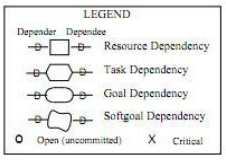


Figure 1. An example of a strategic dependency model

Moving on to the second theory, Winograd et al. (1987) put forth the conversation for action theory based on Searle's(1969) speech act theory or, in computer science world, language action perspective. They argue that business processes are networks of conversations that occur between different business process participants about achieving organisational goals. They introduce the conversation for action diagram (see figure 2) and believe "speech acts are not individual unrelated events, but participate in large conversational structure"(Winograd, 1988). The next paragraph explains the conversation for action idea and shows how, in our work, it has been combined with the strategic dependency model idea.

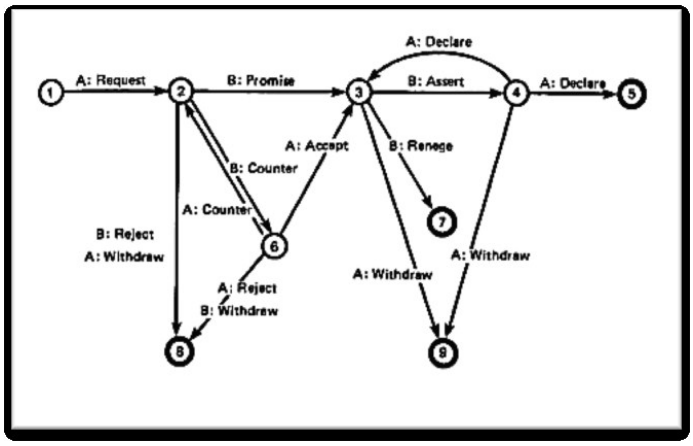


Figure 2. Conversation for Action diagram

The concept of "dependency cycles" that is inspired by the i* framework's SD model and conversation for action theory is a new concept. It combines the SD model concepts with the Winograd et al. (1987) conversation for action theory and tries to capture what drives these dependencies and what role they play in moving a business process forward. The dependency cycles represent fragments of business process instances as they follow the conversation for action diagram states. They show how people converse and, in conversing, depend on each other to fulfill an objective. A dependency cycle starts with a request and ends with either a withdrawal, a rejection of the result, or the declaration of the fulfillment of the objective of the

dependency cycle i.e. with the dependum of the dependency cycle. Dependency cycles have the same types as the dependency types in i* framework. So there is a goal dependency cycle, a task dependency cycle, a resource dependency cycle and a soft goal dependency cycle. The type of the dependency cycle depends on the conversation that contains the request concept. A request either defines whether the initiator is asking for some sort of objective to be fulfilled, resource to be delivered, task to be done; or it is a soft goal that restricts the qualities of one of the other dependency types.

Dependency cycles consist of two dimensions. In one dimension they demonstrate the sequence of occurrence of main dependencies that eventually result in meeting the business goal of the specific business process under investigation. In the other dimension they show different dependency types that are created in nested dependency cycles to fulfill a higher-level dependency cycle goal, or, in other words, to deliver a higher-level dependency cycle dependum. These dimensions reveal interesting information about the organizational structure, communication forms and organizational culture in conjunction with business processes.

Moving on again to consider how humans acquire knowledge held by other humans, there is a popular view about knowledge acquisition, which can be considered one of the main tools that the business analysts use for learning about the current situation of the business processes, that "expert's minds are filled with nuggets of information about their specialised domains. The knowledge engineer then mines these nuggets of knowledge from the head of the expert one nugget at a time" (Anthony, T. et al., 1992). This view might not be fully complete or comprehensive, but it directed this research to try to find the answer to the following questions: how can knowledge engineers or business analysts try to mine or extract the knowledge from experts' heads? What type of questions do they usually ask? Are there any commonalities between these questions?

By analysing different techniques, such as interviewing, workshops, questionnaires and even observational techniques, it has been concluded that almost all of them are trying to ask people to remember how they carry out their day-to-day jobs. They are actually posing different forms of the same question: " What have you being doing at time T in place P?". It can be argued that this question is the retrieval query for episodic memory (Tulving, 1984). The term episodic memory refers to the memory of events happening in a specific place at a specific time (Hasselmo, 2011). The answers to the episodic memory retrieval queries are "short time slices of experience with beginning and end points often related to achievement of a specific goal" (Conoway, 2009). What does this sentence brings to mind in the context of business processes, or, in other words, what are the short time slices of experience with beginnings and end points related to achievement of a specific goal? This paper argues that each is the definition of a fragment of a business process instance. It is a process because it has a beginning and an end, it shows what happens from the beginning to the end, and it is related to achieving a specific goal; and it is "process instance" because it is an experience and it is not abstract; and, finally, it is a "fragment" of the business process instance because it is a short time slice experience.

And it seems that because the question is usually asked in a generalised form, "What do you do in your day-to-day job?", respondents' minds process the outcomes of their episodic memory and create a more generalised answer. It means they remember several fragments of business process instances, try to find similar patterns between them, turn them into a more holistic process and describe it in an abstract way.

The next step in the process of developing the dependency cycle concept involves finding where to look for the fragments of business process instances, or, in other words, finding where the source of information that the fragments of business process instances model can be extracted from, the source of information that is more reliable than human memory and at the same time has been generated by humans. Conversation for Action theory was examined at this point. Winograd et al. (1987) believe that business processes are networks of conversation for action that are happening in the organisation around the organisational goals. The theory suggested that the source of information should be a history of organisational communication, or, in other words, conversation logs. Fragments of business process instances can be found inside those communications and interactions that have happened to achieve the organisational goals. Winograd et al. (1987) have introduced the conversation for action diagram (see figure 2).The authors believe that this is a good starting place for extracting the networks of conversations from any type of conversation logs.

Winograd et al. (1987) also believe that organisations are networks of commitments. This idea is quite inline with the i* framework that has been built around the intentional, strategic, autonomous actor, that in relationship with other actors, by considering different "dependency configurations", fulfils its goals and responsibilities (Yu, 2010). These two definitions, in conjunction with episodic memory and conversation for

action theory, led us to create the concept of dependency cycles. Dependency cycles show how different organisational roles depend on each other to fulfil an organisational objective. The dependency cycles represent fragments of business process instances as they follow the conversation for action diagram states. They show how people converse, and, in conversing, depend on each other to fulfill an objective.

2.2 Extracting the dependency cycles from conversation logs and analysing them

In this sub-section of the paper a set of principles and heuristics will be introduced that facilitate the extraction of dependency cycles from conversation logs. These principles and heuristics are not limited to any specific conversation corpus, but a specific method has been developed based on them for email corpora. This method has been applied for the case study be discussed later.

2.2.1 Principles and heuristics for extracting dependency cycles from conversation logs

The following heuristics and principles can be used for extracting dependency cycles:

1. Conversations that contains some form of "request" starts a new dependency cycle. This heuristic has been put forward based on the fact that some sort of dependency is usually created when we ask someone to do something for us.
2. A dependency cycle can have one or more nested dependency cycles: to deliver a dependency's dependum, the dependee might need to create more dependencies. For instance, the root dependency cycle might be a goal dependency between two actors. Then the dependee might create some resource dependencies to obtain the necessary resources for achieving the goal and some task dependencies to carry out some of the steps. In turn, the resource providers might have some task dependencies, and some soft goal dependencies with some other actors to create the resources in time and with certain quality.
3. Conversations containing some forms of reject, withdraw or declare concepts end a dependency cycle. These concepts show that the dependee has either delivered the dependum or has failed to provide the requested outcome.
4. The intermediary steps in a dependency cycle are similar to the intermediary steps in Conversation for Action diagram. The main point is that not all the intermediary states will be explicitly traversed. For instance, usually, after a request, the dependee fulfils the requested responsibility and goes to the assert state without explicitly promising to do the job, and the depender starts using the provided outcome without explicitly going to the declare state. The withdraw state is also complicated. Sometimes people withdraw from a dependency by just not responding. The main point here is that it can be argued that the intermediary steps are not as important as the first and last steps for a dependency cycle. As long as it can be found that a dependency cycle has started, and the sub-dependency cycles can be defined, and the dependency cycle can be finished by a result, then the intermediary steps can be ignored as it has been accepted that they will follow the pattern that is introduced in Winograd's(1987) conversation for action diagram.
5. The sequence of dependency cycles, nested within a main dependency cycle, can be concluded from their start and finish times. The start time of a dependency cycle is when a request has been made and, interestingly, the end time of the dependency cycle for this specific purpose is when the result has been asserted by dependee (not declared by depender), or the dependee has withdrawn or rejected to provide the dependum. It can even be inferred that dependency cycles, which have time overlap, can be considered parallel.

Dependency cycles can be modelled using BPMN choreography diagrams (Briol, P., 2010) (see figure 3). The focus of these diagrams is on the messages that are being exchanged between participants. The created diagrams have two dimensions. One that shows the sequence of choreography tasks and sub-choreographies and one that shows the way people depend on each other two carry out their tasks.

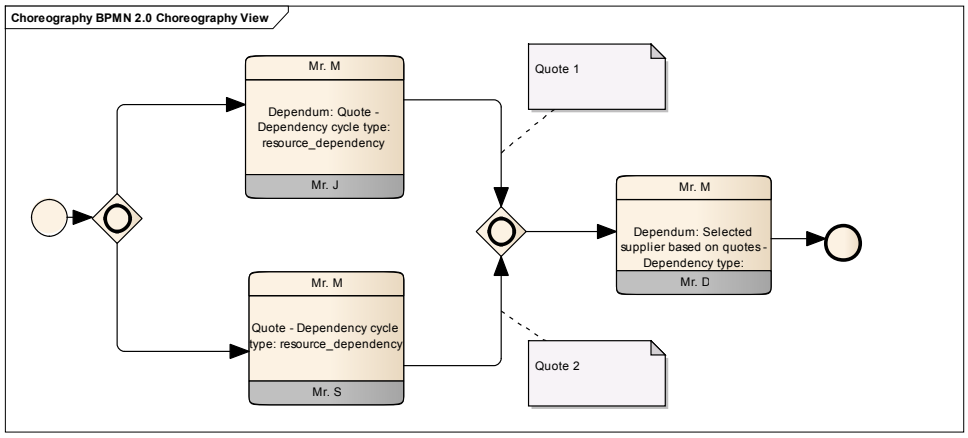


Figure 3. An example of extracted dependency cycles modeled using choreography diagrams

Figure 3 shows how a number of people have interacted (depended on each other) over selecting a supplier for a software development project. It shows Mr. M has initiated a resource dependency cycle with Mr J. for a quote, he also has initiated a resource dependency cycle with Mr. S in parallel. In the Joining gateway we have either quote 1, quote 2 or both (depending on the dependency cycle closure – if the dependum has been delivered or not). After that, Mr. M has initiated a goal dependency cycle with Mr. D to assess the quotes and select the preferred supplier. This diagram is only showing one dimension of the dependency cycle model which is their sequence to achieve the objective of the process which is selecting a supplier. The other dimension is the nested dependency cycles for each of the initiated dependency cycles. For instance, for the last dependum which is "selected supplier based on quotes" and is initiated by Mr. M, Mr. D might start initiating several other dependency cycles to deliver the dependum.

2.2.2 Dependency cycles' patterns and how they can be analyzed

Apart from business processes, dependency cycles demonstrate a dependency pattern between different project roles. This dependency pattern can reveal a lot of valuable information about team culture, suitability of assigned roles and responsibilities, and communication patterns. Analyzing successful and unsuccessful projects and teams, and extracting their dependency cycle patterns and reflecting on them

may help to identify future project problems identified by referring to those extracted patterns.

The following scenario clarifies what the dependency cycle patterns are, how they can be analysed and how they can help us find organisational and communication problems with the project team. The project team and their roles and responsibilities will be introduced first; then, what the expected dependency patterns are, based on several successful projects that have had similar team configuration, will be argued.

A project team has been built to develop a product that consists of two main sub-products: a physical book series and a software system that enables the readers to read the book online and communicate about the book content. This project team includes:

1. Product manager: the owner of the whole product idea (usually a very senior person in the publishing team).
2. Product owner: responsible for ensuring that the product that is being built is in line with the product owner's idea. The product manager is usually more involved with the day-to-day project and monitors the product development progress.
3. Non-technical project manager: responsible for the overall management of the project from physical book publishing to developing the supporting software.
4. Technical project manager: responsible for managing the software development team and the software development sub-project.
5. Editors
6. Developers and testers
7. Graphic designers
8. Suppliers: responsible for supplying any type of necessary goods for performing the project tasks.

Let us consider what we would usually expect the dependency cycle patterns to be for this specific scenario and context and then discuss what the anomalies can disclose. What is being introduced as an expected dependency pattern has been concluded from the experience of a number of successful projects, but it is not meant to be definitive at all. More patterns should be extracted from successful and unsuccessful projects to ascertain the degree of support for this preliminary idea.

The product manager is a senior publisher who has a very high level idea about what the product should be and what features and functionalities it should have, so he usually has a number of high level goals and objectives about the product that he would like to achieve. The product manager usually communicates with product owners and delegates the responsibility of the development of a specific product to the product owner by defining the high-level goals and objectives. Naturally we expect product managers to initiate a lot of goal dependency cycles specifically with product owners to make them aware of their expectation about the to-be built product.

Product owners are publishers as well. Based on what they have understood from the product manager's expectations, they try to keep the developing product close to the image they have got from the product manager in their mind. As product owners are not usually technical, they have a high level understanding of the product features and functionalities, but they are more involved with the day-to-day activities of the project. At this level of management we still expect a lot of initiated goal dependency cycles between the product owner and project management team and a number of soft goal dependency cycles that define the high level quality of the product.

The non-technical project manager is responsible for managing the publishing part of the project and also liaises between the software development and publishing teams if any communication is necessary. So we expect a lot of initiated task dependency cycles and resource dependency cycles between the non-technical project manager and the publishing team, like authors, editors, typesetters, printers etc. As the technical project manager reports the progress of the development side to the non-technical project manager, we expect to see some initiated resource dependency cycles (reports) between non-technical project manager and technical project manager.

The technical project manager works in the same capacity as the non-technical project manager but for the software development team. So we expect a lot of initiated task dependency cycles, resource dependency cycles and also soft goal dependency cycles.

Assuming that the aforementioned dependency cycle patterns show a healthy project team (as mentioned before, based on successful project experience) any deviation from these patterns might be an alarm for some sort of problem. For instance, imagine that no resource dependency cycles have been extracted between non-technical and technical project managers. It shows that there are no communications between these two teams or there are some task dependency cycles between non-technical project manager and the development team. This might show that the technical project manager is not doing his job well and non-technical project manager is taking over his role that can be dangerous at points. People on the higher level of an organizational chart should be able to delegate tasks and create goal dependency cycles and leave the task details to the people who are doing the job. If there are people in those positions that are not able to do that, initiated dependency types vividly show that and the decision maker can find a more suitable position for them. Analyzing dependency cycle configuration can reveal interesting information about dynamics of the project team. More of these anomalies will be analyzed in the case study section with a real life scenario. The interesting point here is that the more projects that are analyzed using this technique, and the more healthy patterns that are extracted and analyzed, the easier it is likely to be for the analysts and consultants to find the problems of the future projects.

3. DEMONSTRATION (CASE STUDY)

In this section, a case study will be introduced in which the proposed technique has been successfully used to find out some of the organizational problems with the project team. The simple expected-dependency cycle patterns that were introduced in the previous section were extracted by applying the proposed technique to two successful projects within the same company and same context.

The project under investigation had the exact same configuration and team set up as the hypothetical team that was described in subsection 2.2.2 The team had realized that they hadn't been working up to their potentials. Deadlines were getting missed, the product owner was not completely happy with the outcomes of the project and the team morale was very low in general. So we were invited to investigate the problem and see what we could find to help the team to perform better. In addition to other conventional methods like counseling and observation, we applied the proposed technique and produced results that were not obtainable using other techniques.

Email messages were used as conversation logs. In the first step, all the email messages that were related to the project under investigation were captured and irrelevant emails were filtered out. Then using the proposed technique's principles and heuristics, the dependency cycles were extracted and modeled using BPMN 2.0 choreography diagrams. Apart from analyzing those choreography diagrams that were instances of some of the business process fragments and finding some optimization solution to the processes they were executing, the following anomalies were discovered in the configuration of the dependency cycles. The base of comparison to detect the anomalies were those expected-dependency cycles that were discussed in the previous section (due to privacy issues the exact figures cannot be revealed):

1. A large number of "task dependency cycles" between product owner and project managers (both technical and non-technical)
2. A large number of "resource dependency cycles" between product owner and the rest of the team
3. No "soft-goal dependency cycles" extracted
4. Small number of (any type of) dependency cycles between non-technical project manager and the technical project manager.

The first interesting detected anomaly was a large number of "task dependency cycles" between product owner and project managers. Due to the role that a product owner plays in the team, more goal dependency cycles and soft-goal dependency cycles (expected quality or non functional requirements) were expected. The product owner is responsible for conveying the high level image of the product to the project team and making sure that what is being built is in line with that image. Too many task dependency cycles could be an indication of one of the following causes:

- A company culture that expect the product owner to have full control over the project, which causes a lot of problems as product owners are not necessarily good project managers.
- The personality of a product owner who would like to have full control over every single task that is being done.
- A trust issue which shows the product owner does not specifically believe in the capabilities of the project managers
- Actual lack of capability of the project managers that makes the product owner to play their roles as well.

The second interesting anomaly detected was a large number of resource dependency cycles between the product owner and the rest of the team. The product owner was asking everyone to report to her and the project managers were bypassed. This anomaly could be quite inter-related to the previous one: any of the aforementioned causes for the first anomaly could cause the second one as well. Another specific cause for this anomaly could be badly designed communication channels that again could be a sign of bad project management.

No soft-goal dependency cycle was the third anomaly found. This could be a sign of lack of communication about expected product and project qualities, which, in the end, could cause dissatisfaction for the product owner. In the product level it could again be related to some of the first causes but in the project level it is usually caused by a bad project management practice.

Although the non-technical project manager was responsible for relaying the development progress reports to the product owner, a very small number of dependency cycles had been initiated between non-technical and technical project managers. It could be an indication of lack of communication between these two team members that in turn had caused the product owner to have a very limited understanding of the software product that was being developed and making her to directly ask for reports from the development team.

After analyzing the findings, they were shared with the team and discussed. It was concluded from the findings and consultation that the following list could be the main causes of some of the project problems:

- The personality of a product owner who would like to be involved in every detail of the project and could not delegate the tasks.
- A poorly communicated high level image of the product
- Poorly defined and communicated non-functional requirements of the product that had caused dissatisfaction of the product owner that had caused her to lose trust in the development team.
- A badly designed communication channels.
- The incompetency of the non-technical project manager that had caused mistrust between the product owner and the project team.

4. CONCLUSION

This paper introduced a new concept called dependency cycles which has been inspired by i* framework strategic dependency model, Conversation for Action diagram and episodic memory theory. Dependency cycles can be extracted from conversation logs such as email conversations using the technique that has been introduced in the paper.

By extracting dependency cycles from successful and unsuccessful projects and analysing their structure and patterns, a model can be created that reveals a lot of information about team communication structure, process models that the project team is following, organizational cultures and even some aspect of the team member's personality. This information can help the project team or the consultants and analysts to find the team structural problems, communication problems, bottlenecks, and bad habits in organisational culture and even optimise the processes the project team follows.

REFERENCES

Anthony, T., Cossick, K.L. & Zmud, R.W., 1992. A Synthesis of Research on Requirement Analysis and Knowledge Acquisition Techniques. *MIS Quarterly* , 16(1), pp.117–138.

Briol, P., 2010. *BPMN 2.0 Distilled*, lulu.com.

Cadle, J., Paul, D. & Turner, P., 2010. *Business Analysis Techniques: 72 Essential Tools for Success*, British Computer Society.

Conway, M., Episodic memories. *Neuropsychologia*, 47, pp.2305–2313.

Flores, F., & Ludlow, JJ., 1980. Doing and speaking in the office. *Decision Support Systems: Issues and Challenges*, Pergamon Press, pp. 95-118.

HARRIS, L. & BROWN, G., 2010. Mixing interview and questionnaire methods: Practical problems in aligning data. *Practical Assessment Research & Evaluation*, 15(1).

Hasselmo, M.E., 2011. *How We Remember: Brain Mechanisms of Episodic Memory*, MIT Press.

Hevner, A. & Chatterjee, S., 2010. *Design Research in Information Systems Theory and Practice*, Springer.

Hevner, A.R., 2007. A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems*, 19(2), pp.87–92.

Hoegl, M., 2005. Smaller teams–better teamwork: How to keep project teams small. *Business Horizons*, 48(3), pp.209–214.

Mcgraw, K. L. and Harbison-Briggs, K., 1989. Knowledge acquisition principles and guidelines, Prentice- Hall.

Peffers, K., Tuunanen, T. & Gengler, C., 2006. *The design science research process: a model for producing and presenting information systems research*. research in information.

Searle, J.R., 1969. *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press.

Tulving, E., 1984. Precis of Elements of episodic memory. *Behavioral and Brain Sciences*, pp.223–268.

Winograd, T. & Flores, F., 1987. *Understanding Computers and Cognition: A New Foundation for Design*, Addison Wesley.

Winograd, T., 1988. A language/action perspective on the design of cooperative work. *ACM SIGCHI Bulletin*, pp.203–220.

Yu, E., 2010. Modelling strategic relationships for Process Reengineering. In *Social Modeling for Requirements Engineering*. MIT Press, pp. 11–152.