Transdisciplinary Lifecycle Analysis of Systems
R. Curran et al. (Eds.)

doi:10.3233/978-1-61499-544-9-142

Mechanisms of Dependence in Engineering Projects as Sociotechnical Systems

Bryan MOSER[a,1], William GROSSMANN[b] and Phillip STARKE[c]
[a] *Massachusetts Institute of Technology*
[b] *Virginia Polytechnic Institute and State University*
[c] *Technical University of Munich*

Abstract. With coordination defined as the management of dependencies, complex engineering projects are well coordinated when teams are aware and able to respond to demands for interaction across product and organization systems. Classic representations of dependence in project management standards and practices emphasize sequence: a single dimensional consequence of dependence. However, underlying mechanisms of dependence which drive remain assumed or hidden, preventing analysis of systemic consequence on scope, quality, schedule, and cost. This paper begins with a review of dependence as viewed commonly in system engineering and project management. Building on our recent work to consider engineering projects as sociotechnical systems, we propose dependence characteristics which more meaningfully capture underlying project activity dynamics. Mechanisms are proposed for dependence which are satisfied by the interplay of demand for interaction and the supply of coordination. Attention allocation and exception handling behaviors in the project organization influence the extent of local satisfaction of dependence. Project architectural characteristics lead to emergent and systemic impacts on cost, schedule, and quality. Our next step in this research is introduced, the instrumentation of teamwork experiments to observe and validate the demand and satisfaction of dependencies by project team during complex project execution.

Keywords. Project Design, Complexity Management, Complex Dependencies, Interdependence, Concurrent Engineering, Program Management

Introduction

In response to increasing sociotechnical complexity of engineering projects, an emphasis has been placed on standards and practices for systems engineering and project management. While there have been successful improvements, examples of failure in large engineering projects are sufficiently alarming to focus attention on the efficacy of these practices. We have argued that today's engineering environment of dispersed teams, concurrent work, and increasingly complex subsystems has led to a decline in teams' abilities to anticipate and respond to needed coordination. Information capture, archive, and search have become abundant and inexpensive, yet performance has not necessarily improved. Wasteful attention to information of little value can itself create risk of poor quality, schedule delays, and budget overruns.

Surprisingly, existing project management (PMBOK, Prince2, P2M, etc.) and systems engineering (SEBOK) standards treat task dependence in a narrow way – as sequence. Often the activities driven by dependence are characterized as overhead or "soft" activities, with only a few consequences of dependence treated through network or topological analyses on top of these narrow representations. Yet we continue to see

[1] Corresponding Author, E-Mail: bry@mit.edu

that activity driven by dependence in engineering can reach up to half of a project's attention, often in unexpected positions and timing across a project. Thus existing handling of dependence assuming no, little or uniform coordination is inaccurate and calling for better coordination a truism. Instead we consider design of an engineering project to reflect – within limited capabilities and capacities – when, where, and why coordination to satisfy dependence is valuable and feasible. In this paper we begin by exploring an improved definition of dependence suitable to capture the dynamics of complex engineering systems projects.

1. Context: Instability of Experience and System Complexity lead to Surprises

What we produce and *how we work* combine as a sociotechnical system in which products, processes, and people interact and evolve. Over time, if system characteristics and work styles align, then improved performance is promoted and predictable. If a product system changes quickly and work styles are unable to adapt, the relevance of past practices diminishes. Likewise, if ways of working change quickly without consideration of the systems produced, performance may also be at risk. In an environment when both ways of working and complexity of systems change simultaneously, the emergent performance characteristics of an engineering project become quite difficult to anticipate. For this reason the thoughts leaders of scientific management a century ago promoted standard work and reduction of variation in both parts and people.

Teams in the past commonly worked on new product systems located in the same buildings, corporation, and work culture. Development phases were more likely to have been sequential, the workforce likely to be experienced with the previous generation of product. Relationships – both formal and informal – existed amongst teams over time and across multiple projects. In contrast, recent complex engineering development projects are marked by meaningful changes in how product, process, and organization architectures are organized and linked. These characteristics of recent projects are beset by fundamental trends which exacerbate the instability and complexity leading to surprise.

In this paper we view interactions across boundaries as activity, with ability and experience relevant to performance. Since demand for coordination changes in a sociotechnical system due to the overlap of dependence and teams, then instability in product and teaming naturally leads to changed patterns of demanded interaction. We argue, consistent with Nonaka [1], that interaction patterns over a career are an aspect of tacit knowledge. Organizations may be highly capable to coordinate given past experience, through both formal and informal relationships across system and organization. The teams become experienced and able to recognize critical interactions in tradeoff with all of the demands on their time.

We witnessed the contrary in a recent multi-billion dollar aerospace initiative, cancelled after years of planning, designing, and engineering across a highly distributed organization and product system. A lead engineer lamented that the performance of teams was lacking, even though requirements had been carefully mapped, interfaces listed, work packages defined and assigned. In her words, the teams had no “feeling for the dependencies.” The I.T. automated workflow, meant to reinforce attention to critical matters may have had the opposite effect, instead preventing development of tacit knowledge and experience at uncertain interactions. The teams were focused on

their own work and their side of each interface, rather than the back and forth interactions across dependencies.

2. Related Work

Literature on dependencies within engineering projects can mainly be found in the domains of portfolio management, process management and the study of team performance. In all of these fields much is based on the work of Thompson [2] on organizational dependencies. It is commonly stated that project activities are dependent due to resource sharing and other factors such as time constraints, project outcomes or risk profiles [3]. Understanding and managing the dependencies between projects is considered to be a critical issue [4].

Dependence of specific tasks is often discussed centered on improving product development processes [5]. The links are shown to be a result of work outputs that to be passed from an upstream to a downstream task, thus making it an issue of task sequence. Only few see task dependencies as a broader concept in need of further investigation beyond precedence, functional, and probabilistic dependencies [6], [7]. Task dependence is also central in many works concerned with team performance [8]. It is generally stated that high task dependencies in teams produces positive effects [9]. Research by Shea and Guzzo [10], for instance, suggests that task dependence positively influences team efficacy while Campion [11] finds that it correlates with employee satisfaction.

Dependence also plays a large role on interpersonal relationships [12]. Thus, insights gained by analyzing the psychology and sociology of interpersonal relationships can be adapted and applied to the study of dependencies in product development. Two main theories serve as a basis for the research of dependence in interpersonal relationships, *Interdependence Theory* [13] and The theory of *Cognitive Interdependence* [14].

2.1. Dependencies in Textbooks and Project Management Standards

Tasks are the lowest level of activity unit, an atomic element to the models of projects. Dependence modeled as precedence is therefore a relationship amongst milestones [15], [16]. A task dependent on another task is characterized by these precedent constraints: e.g. Finish to Start (FS). These commonly used practices also categorize dependence as *Internal vs. External* (do the two activities fall *within or across* some defined system or boundary?) and *Hard vs. Soft* (the dependence *cannot or can change* within the horizon of the project?) We note that the mathematical relationship describing the dependence contains very little information, only the expected schedule consequence rather than any of coupled characteristics of the activities. Importantly, the underlying driver of dependence – the essential meaning – is not expressed.

2.2. Dependence as Process

The Critical Path Method (CPM) and PERT were born of industrial and military projects in the 1950s. Dependence in these models is taken as discrete sequence constraints amongst tasks. PERT differs in that uncertainty in duration is considered.

Typically these models are used to control critical elements of development. PERT and IDEF, task-based charting methods, model the process of tasks. As in CPM, tasks are related through time-based relations such as precedence constraints.

Rather than a Gantt-like display against calendar time, both PERT and IDEF portray the flow of a tasks as a network. Each task is a box, with dependencies shown as lines connecting the boxes. Other modeling languages (IDEF, UML, SysML, OPM) and tools have since emerged. Enterprise architectures, such as PERAM, CIMOSA, and GERAM, are used to represent a total product development process.[17]

The design structure matrix (DSM) is an N-squared matrix which emphasizes the dependencies and their pattern across a set of functions or tasks. teward, published a description of DSM in 1981 [18] with the tasks as the element along the rows and columns of the matrix. The problem addressed by DSM is to find a sequence of the tasks avoiding a delay, rework, or poor quality. In some cases in which a subset of tasks are tightly coupled these tasks are

partitioned; acting as a subset which as a whole satisfies the desire to have all dependencies in the lower left triangle.

Dependence has also been defined as a demand in one activity for information from/in another activity. Traditionally one might refer to the "source" of the information as the upstream activity, and the demanding "sink" activity as downstream. In a network or PERT diagram which shows activities on nodes and arrows as dependence, the arrow can be considered a pipe for directional flow of needed information, from upstream to downstream. A measure of the dependency strength (or depth) can be represented as the amount of information available upstream that is needed downstream.[6]

2.3. Coordination as the Management of Dependence

Clark and Fujimoto [19], in their characterization of the development process as a "system of interconnected problem solving cycles", observe that in theory shorter lead times can be achieved through "Integrated Problem Solving". Integrated problem solving increases the dependency amongst tasks, with increased overlap and mutual communication. However, case study data showed that the shorter lead times expected theoretically from such concurrency are difficult to realize. When teams work on dependent activities they may need to coordinate. "Different types of coordination result from different kinds of dependencies, which in turn are dependent upon different kinds of products, services, actors, work specializations, efforts, tasks and purposes." [20] Malone and Crowston, after a review of many related terms, defined coordination as "the management of dependencies". [21]

3. Definition and Characteristics of Dependence in Engineering Projects

In our previous work we have presented a modelling framework for continuous, concurrent, and mutual dependence. [7],[22] We quickly summarize this previous work and expand the definition to describe the underlying mechanisms of dependence driving dynamics of engineering projects as sociotechnical systems.

3.1. Dependence as Essential Need to Interact

In project management, systems engineering, computer science, information theory, linguistics and other fields "dependence" is a common term. Here we explore a definitions across these different disciplinary uses of the word, beginning with a dictionary definition of dependence: "the quality or state of being dependent; *especially*: the quality or state of being influenced or determined by or subject to another" [23]

The nature of a system determines the meaning of *influence and determination* in the definition of dependence amongst elements within the system. The significance of a dependency is that the essence of an element – a task, a system, an organization, a phrase – cannot be realized without the awareness of some other element, which itself exists with a meaning of its own. Therefore, in contrast to definitions of project dependence as task sequence, our definition is tied a broader view of *dependence as need*. A meaningful representation should go beyond a measure of consequence and be tied to the root cause of the need. For engineering projects, "dependence is defined as need for interaction that matters – a demand for coordination – so that an activity's outcome is successful" [7].

3.2. Dependence as Demand for Interaction during Concurrent Progress

In addition to a demand for some amount of information, one can also consider the timing of the flow. The information from an upstream task may be needed completely before another task begins, which is the meaning of classic Finish to Start (FS) dependency. However, the tasks could also be dependent while proceeding in parallel, creating a continuous, concurrent demand for interaction. This continuous, *concurrent dependence* is common amongst critical design tasks, yet unable to be easily represented in the classic approaches.

This same representation of dependence may be subdivided to allow further granularity as stages of concurrent progress (shown on in Figure 1.) The concurrent progress diagram is divided into sections which each may have a depth and shape to express the need from upstream to downstream. Likewise, in a view of concurrent progress, there may be a demand for interaction through flow of information and resources amongst tasks in both directions.

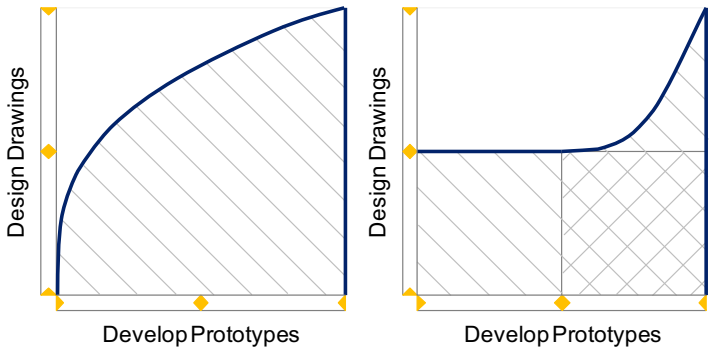

Figure 1. Concurrent Dependence with Stages from [13].

3.3. Dependence as Exception Trigger

In Figure 2 taken from our recent paper, concurrent and mutual dependency characteristics are shown in one diagram. If the state of progress is in an unconstrained area, and dependencies have been satisfied through coordination, the two tasks at that moment are effectively independent.

Finally, our representation of project dependence includes not only dependence as continuous flow, but also the exception handling behavior of the project organization. Other researchers have used network diagram representation of dependence with related exception handling [25]. The concurrent and mutual dependence diagrams treat the "shaded" areas as zones of exceptional activity. Why? If a downstream task proceeds without the needed upstream information, or perhaps the received information is defective, then the progress and quality of the downstream task is at risk. In this way, the dependence if poorly coordinated is a source of exception propagation.

It is possible that mutual progress is positioned in the concurrent dependence that should otherwise be constrained: in a shaded area. Two dependent tasks which touch and cross the boundary (referred to as the "exception boundary") will trigger errors and exception handling behavior.

The dependency in Figure 2 is mutual, driven by need in both directions, from Design to Prototype activity, and inversely from Prototype to Design activity (the shaded area in the upper left). Thus the open area -- when the two tasks can operate as long as coordination is effective -- is a zone of nominal activity. Crossing the boundary into a zone of exceptional activity triggers a demand for the organization's exception handling behavior.

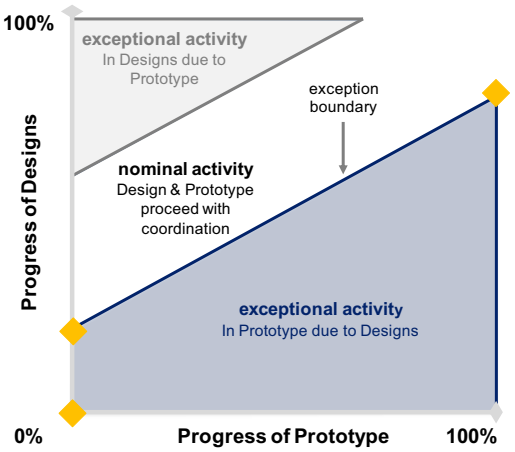

Figure 2. Mutual, concurrent dependence with exception boundaries [7].

4. A Framework of Dependence Mechanisms

4.1. Characteristics of a Practical Dependence Representation

In summary this representation of project dependence is *continuous, concurrent and mutual.* A dependence captures a concurrent (and possibly mutual) need for information and results as a continuous function of progress of activities, generating demands for coordination activity. If a dependence boundary is crossed, exceptional activity is triggered.

The dependence is a constraint on independent performance. This constraint is mitigated through effective coordination which satisfies the dependence by addressing an underlying need. Coordination itself is real activity, requiring awareness, attention

allocation, abilities, and experience to be well performed. Therefore, coordination takes time, cost, and impacts quality. We've seen across tens of projects that this simple, visual representation becomes readily useful in practical, industrial settings.

By examining the mechanisms of dependence more closely, one can analyze the local and systemic impacts of the dependence on project performance. To overcome the limits of the classic view of dependence as sequence, the consequences of coordination (better or worse) should be multi-dimensional: one should forecast the systemic effect of the dependence being satisfied on project cost, schedule, and quality.

4.2. Mechanisms of Dependence

A close view of the mechanisms for satisfaction of dependence is shown in Figure 3 below. Dependence is driven by two types of causes, sources of need we term *Flow* and *Pool* causes. A flow cause of dependence is a need for results or information from another task.

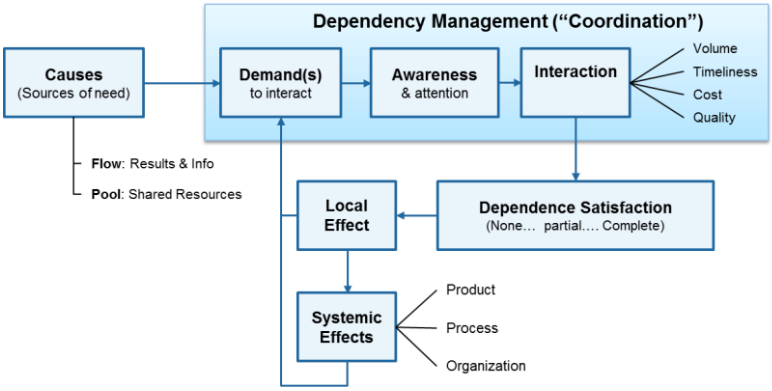

Figure 3. Mechanisms of Dependence from Cause to System Effects.

A pool cause of dependence is a need for a resource shared by another task. They both lead to a demand for interaction. Awareness of the dependence and allocation of attention are the major factors influencing how or if any interaction takes place. The volume, timeliness, cost, and quality of the interaction all have consequences regarding the satisfaction of the dependence.

Dependency management, or coordination, may influence the demand itself, the awareness and the allocation of attention, as well as the interaction. Many classic dependency management techniques aim at improving the awareness of the dependence (e.g. CPM or DSM) or improving the interaction (e.g. action plans or standardization).

The extent to which the dependence is satisfied determines the local effects, which in turn influence the systemic effects. Local effects are the immediate consequences for the tasks (e.g. delay, costs, and rework) and the individuals (e.g. frustration or establishment of trust) whereas the systemic effects influence the significance of the local effect on product quality, the process as a whole, and the organization. These effects in turn can lead to a change in the remaining demand to interact. If the dependence is fully satisfied the demand is effectively eliminated and thus no demand to interact remains. If the dependence is only partly satisfied or not at all satisfied through insufficient interaction, demand to interact may decrease or – in some cases – increase.

4.3. System Consequences of Timing and Quality of Dependence Satisfaction

Given these characteristics and mechanisms of a project dependency, how do these combine with behaviors of teams to drive systemic results in cost, schedule, and quality? Below in Figure 4 we show a concurrent dependency with two stages. For the first 20% of the upstream design work the progress of downstream prototypes is prevented,. e.g. information from that initial 20% is needed to begin the first prototype. After this point, in a second stage of dependence, as long as designs proceed (at pace) ahead of the progress of prototypes, and coordination ensures transfer of upstream results, then the prototypes can proceed without exception.

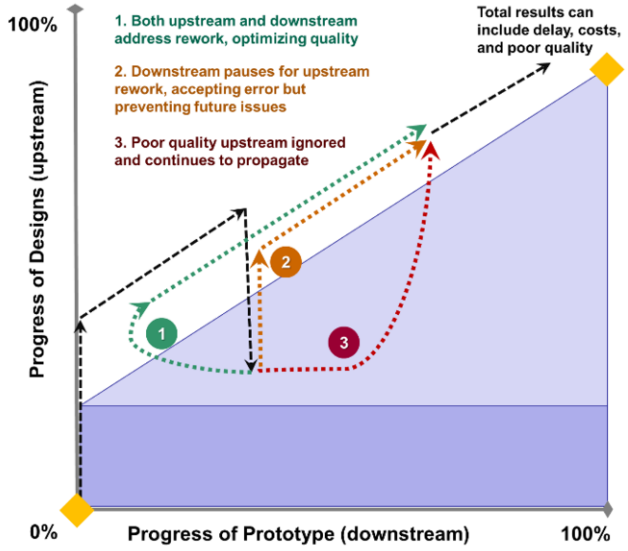

Figure 4. Systemic Consequence of Complex Dependence.

A dark dashed line shows a hypothetical mutual progress path, beginning in the lower left. The path shows designs proceeding to about a third of progress before the prototypes begin. Until about a quarter of the prototype scope is completed the mutual progress seems nominal, far from the dependency exception boundary. However, the progress upstream is reversed (a bug discovered), and the mutual progress is no longer in the open, unconstrained area but instead placed in the middle of the area of exceptional activity.

In order to analyze the cost, schedule, and quality implications of this condition, the behaviors of the teams come into play. The mutual progress from that point, now shown in the middle of the exceptional activity area, is driven by the combination of team behaviors. Have the teams prioritized and paid attention to quality? Is this issue even noticed? If so, how does each team respond and make a decision on how to proceed? Given these teams' allocation attention and exceptional handling behaviors the resulting paths could lead to a recovery of quality through rework at some delay and increased cost. It is also possible that the quality issue be missed or ignored, leading to undetected quality issues both in the Designs and the Prototype.

Significantly, in contrast to a singular dimensional analysis of schedule effects such as the critical path method, this representation and the mechanisms of dependence allow one to forecast total results as emergent and a mix of total cost, duration, and

quality. Choices by teams amongst many demands on their attention, driven by behaviors and priorities, changed the paths of progress within and across tasks. Typically in complex engineering programs the awareness of teams to the systemically meaningful dependencies is challenged and combined with natural limits in capacity and ability.

5. Next Steps and Validation

The validation of our research into engineering projects as sociotechnical systems will require the instrumentation of performance during complex project planning and execution. For this paper's representation of dependence, we will observe projects in progress to test the dependency model's practicality and usefulness. We are preparing a platform to measure the demands on and the attention of teams across product, process, and project organization. The responses to dependence will be correlated to local and systemic performance. Also, we have begun a series of experiments to test the effect of increased awareness of concurrent and mutual dependence on local and systemic performance of the engineering project.

6. Conclusions

Why do some teams, even in the face of complexity, perform with excellence? The situation faced by product development initiatives today includes changes in our product, our teams, and how we work together. Judgment and embedded practices – built on decades of traditions and standards -- have lost relevance as new, product, process, and team architectures emerge and overlap. This trend will continue; future engineered system projects will continue to increase in complexity technically and organizationally. We are driven in our thinking and experiments to better understand the nature of teamwork across boundaries, to uncover performance in these complex sociotechnical systems and significantly improve schedule, cost, and quality.

We have seen in industrial practice that techniques which rest on traditional planning significantly misrepresent dependence. Coordination activity requires effort, time, and cost, and for modern engineering projects can be one-third to half of the total attention by teams. Some assume that coordination is a qualitative rather than real activity, and therefore do not recognize the limited capacity of teams to both work and interact with others.

This paper began with a review of dependence as viewed commonly in system engineering and project management. Building on recent work to consider engineering projects as sociotechnical systems, we described dependence characteristics which more meaningfully capture underlying project dynamics. We proposed mechanisms for dependence and their satisfaction through the interplay of demand for interaction and the supply of coordination. We've shown how attention allocation and exception handling behaviors in the project organization influence the extent of local satisfaction of dependence. In turn, project architecture leads to emergent and systemic impacts on cost, schedule, and quality. Our next step in this research focusses on the instrumentation of teamwork experiments to observe and validate the demand and satisfaction of dependencies by project team during complex project execution.

References

[1] I. Nonaka, *The Knowledge-Creating Company : How Japanese Companies Create the Dynamics of Innovation*, Oxford University Press, 1995.

[2] J. D. Thompson, *Organizations in action: Social science bases of administrative theory*, Transaction Publishers, New Brunswick, 1967.

[3] D. Verma and K. K. Sinha, Toward a theory of project interdependencies in high tech R&D environments, *Journal of Operations Management,* Vol. 20, no. 5, pp. 451–468, 2002.

[4] A. de Maio, R. Verganti, and M. Corso, A multi-project management framework for new product development, *European Journal of Operations Research*, Vol. 78, no. 2, pp. 178–191, 1994.

[5] T. R. Browning, E. Fricke, and H. Negele, Key concepts in modeling product development processes, *Systems Engineering*, vol. 9, no. 2, pp. 104–128, 2006.

[6] A. D. Christian, K. J. Grasso, and W. P. Seering, Validation studies of an information-flow model of design, In: *Proceedings of the 1996 ASME Design Engineering Technical Conferences*, 1996.

[7] B. R. Moser and R. T. Wood, Design of Complex Programs as Sociotechnical Systems, In: J. Stjepandić, N. Wognum and W. J.C. Verhagen (eds.): *Concurrent Engineering in the 21st Century*, Springer International Publishing Switzerland, pp. 197–220, 2015.

[8] S. M. Gully, K. A. Incalcaterra, A. Joshi, and J. M. Beaubien, A meta-analysis of team-efficacy, potency, and performance: Interdependence and level of analysis as moderators of observed relationships, *Journal of Applied Psychology*, Vol. 87, no. 5, pp. 819–832, 2002.

[9] G. Van der Vegt, B. Emans, and E. Van de Vliert, Effects of Interdependencies in Project Teams, *Journal of Social Psychology*, Vol. 139, no. 2, pp. 202–214, 1999.

[10] G. P. Shea and R. A. Guzzo, Groups as human resources, In: K. M. Rowland and G. R. Ferris (eds.) *Research in personnel and human resource management, Vol. 5*, JAI Press, Greenwich, 1987.

[11] M. A. Campion, G. J. Medsker, and A. C. Higgs, Relations between work group characteristics and effectiveness: Implications for designing effective work groups, *Personal Psychology*, Vol. 46, no. 4, pp. 823–847, 1993.

[12] C. E. Rusbult and P. A. Van Lange, Interdependence, interaction, and relationships, *Annual Review of Psychology*, Vol. 54, no. 1, pp. 351–375, 2003.

[13] H. H. Kelley and J. W. Thibaut, *Interpersonal relations: A theory of interdependence*, Wiley, New York, 1978.

[14] C. R. Agnew, P.A.M. Van Lange, C. E. Rusbult, and C. A. Langston, Cognitive interdependence: Commitment and the mental representation of close relationships, *Journal of Personality and Social Psychology*, Vol. 74, no. 4, pp. 939–954, 1998.

[15] S.J. Mantel, *Project management in practice*, 4th ed, Wiley, Hoboken, 2011.

[16] H. R. Kerzner, *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, Wiley, Hoboken, 2013.

[17] P. Bernus and L. Nemes, A framework to define a generic enterprise reference architecture and methodology, *Computer Integrated Manufacturing Systems*, Vol. 9, no. 3, pp. 179–191, 1996.

[18] D.V. Steward, The design structure system: a method for managing the design of complex systems, *IEEE Transactions on Engineering Management* , no. 3, pp. 71–74, 1981.

[19] K.B. Clark, *Product Development Performance: Strategy, organization and management in the world auto industry*, Harvard Business Press, 1991.

[20] R. Müller, Coordination in organizations, In: *Cooperative Knowledge Processing*, Springer-Verlag, London, 1997, pp. 26–42.

[21] T.W. Malone and K. Crowston, The interdisciplinary study of coordination, *ACM Computing Surveys CSUR*, vol. 26, no. 1, pp. 87–119, 1994.

[22] B. Moser, F. Kimura, and H. Suzuki, Simulation of distributed product development with diverse coordination behavior, In: *Proc. of 31st CIRP International Seminar on Manufacturing Systems*, 1998.

[23] M.-W. Dictionary, *Dependence*, 2008. Accessed: 03.06.2015. [Online]. Available: http://www.merriam-webster.com/

[24] H.H. Baligh, R.M. Burton, and B. Obel, Organizational consultant: Creating a useable theory for organizational design, *Managment Science*, Vol. 42, no. 12, pp. 1648–1662, 1996.

[25] Y. Jin and R. E. Levitt, The Virtual Design Team: A computational model of project organizations, *Computational and Mathematical Organization Theory*, Vol. 2, no. 3, pp. 171–195, 1996.