Leadership in Sustainable Infrastructure

Leadership en Infrastructures Durables

Vancouver, Canada

May 31 – June 3, 2017/ *Mai 31 – Juin 3, 2017*

WORK STRUCTURING PRINCIPLES, STRATEGIES AND TOOLS: A REVIEW OF THE LITERATURE

Kaur, Kawalpreet[1], and Mitropoulos, Panagiotis[2, 3]
[1] San Diego State University, USA
[2] San Diego State University, USA
[3] pmitropoulos@mail.sdsu.edu

Abstract: In the construction context, Work Structuring refers to the decomposition of the project's scope of work into manageable work packages and the allocation of these work packages to the project participants. Work Structuring determines the scope and complexity of each work package, as well as the dependencies between work packages and the need for coordination between the project participants assigned the different work packages. Work Structuring establishes the organizational structure of the project. As a result, it can influence the contractor's ability to win a project, and has significant implications for project performance. The long-term goal of this research is to develop ways to improve the Work Structuring of construction project organizations. Towards this goal, this study investigates the principles, strategies and tools/methods used for structuring the work in different disciplines with emphasis on construction, manufacturing and software development. The study reviews the literature with a focus on the following questions: (1) How is work structuring currently performed—what are the principles, strategies and criteria and tools that guide work structuring; and (2) What are the implications for performance, and how are these implications evaluated—what factors, metrics, and indicators are used to evaluate work structuring.

1 INTRODUCTION

In the context of construction projects, work structuring refers to the decomposition of the project's scope into work packages and the allocation of these work packages to the trade contractors. These work packages are also called "bid packages"—although sometimes they are negotiated or directly awarded to selected subcontractors. Work Structuring determines the size and complexity of each work package, as well as the dependencies between work packages. The division and relationships of the work packages can influence the cost of the work. Furthermore, the work dependencies create the demands for coordination between the project participants. Such work dependencies often result in "breakdowns" in the work process, with potential delays and added costs, and contractual friction between the project participants. For example, errors or delays in one scope of work may delay or require adjustment by following activities. The dependencies between the work items create dependencies between the project participants (the subcontractors) who are responsible for these items, and greatly influence the need for coordination (Formoso and Isatto 2009).

Work Structuring decisions establish the organizational structure of the production system and configure the project supply chain. As a result, they have significant implications for project performance. Consequently, it is important to have an effective work structure that not only achieves a competitive bid, but also helps reduce the coordination and transaction costs (costs to resolve the possible contractual problems between project participants). The long term goal of this research is to develop a systematic methodology for creating effective work structuring on construction projects. Towards this aim, this study reviews the literature on work structuring in different sectors with emphasis on construction projects, product architecture in manufacturing, and software development. The review focuses on the following questions:

- How is work structuring currently performed—what are the processes, principles, strategies and tools that guide work structuring; and
- How does work structuring influence performance.

2 BACKGROUND: WORK STRUCTURING

According to Mintzbeg (1980), organizational structuring focuses on the division of an organizational mission into a number of distinct tasks, and then the coordination of all of these tasks to accomplish that mission in a unified way. Decomposition of tasks into subtasks and assigning them to different actors is necessary to reduce the complexity and size of the task into manageable pieces, and to benefit from specialized knowledge and resources. The way tasks are assigned to different actors greatly influences the need for coordination (Formoso and Isatto 2009) in order to manage the dependencies between activities (Malone and Crowston 1994). Activity dependencies can be due to (1) workflow, where a task produces an output used by another task or (2) shared resources (Howell et al. 1989). In workflow dependencies, three coordination problems can occur: (1) the output of one task must be available at the time it is needed by the other, (2) the output must be of adequate quality, and (3) the output must be available at the right place (Crowston 1991).

March and Simon (1958) suggested two forms of specialization to guide the process of task decomposition and assignment: (1) specialization by process, grouping people who know only a particular process, and (2) specialization by purpose, grouping people who work on a particular product (or part of the product). In process-based division, a work package includes work scope that involves the same specialization and skills – for example, framing, carpeting or painting. In product-based division, one work package includes product assemblies that involve different trades, such as a wall system, or a bathroom, or the entire interior work scope of a building. Modularization involves product-based division, where entire product assemblies are awarded under one work package.

2.1 Work Structuring in Construction

During the breakdown of the project scope into work packages, there are multiple important goals and requirements (Mitropoulos and Sanchez 2016): (1) Meeting the owner's requirements, which include cost and schedule goals, requirements for contractors' prequalification, and requirements for local, small or disadvantaged contractors participation. (2) Reducing the project cost and risks by avoiding scope gaps or overlaps attracting competitive bids, avoiding double mark-up (due to second and third tier contractors), taking advantage of economies of scale. (3) Managing project risks by attracting competent, reliable trade contractors, preventing coordination problems and frictions, and establishing clear point of responsibility for product quality.

Construction researchers have emphasized the importance of work division in construction. Globerson et al., (1994) argued that the mismatch between the work breakdown structure and the organizational structure has a negative impact on the project progress. According to Vrijhoef and Koskela (2000), a major part of the problems in construction projects originate at the interfaces of different parties or functions. Draper and Martinez (2002) highlight the fragmentation of supply chain in construction and how it constrains the production process of the physical product. Tommelein and Beeche (2001) emphasized the importance of managing the interfaces between components and activities and suggest that decoupling between activities as a result of shared resources helps increase work flow. In recent years, there has been an increased attention to the management of the interfaces between project participants (Kuprenas and Rosson 2000, Chua and Godinot 2006, Chen et al. 2008, Chen et al. 2010, Noteboom 2004, Pavitt and Gibb 2003). These researchers highlight the severity of interface issues and the necessity of interface management. However, they focus on the management of the interfaces rather than the determination of the interfaces. The work structuring decisions are directly related to these issues, as the work packaging determines to a large extent the project interfaces. Poor division of work among these work packages leads to work disruptions and reworks (Brotherton et al, 2008).

One way to address the problem of managing the interfaces is procurement of work “clusters.” Nicolini and colleagues (Nicolini et al. 2001) reported two pilot projects where the work was divided and procured in “clusters” such as “Building frame and envelope,” “Internal finishes” and “Mechanical and electrical services.” These clusters included actors who have high interdependence so they could internalize interfaces and coordination. Cluster members were selected during schematic design and participated in design development. Tsao (2004) found that work structuring affects not only the coordination costs, but also the production costs of the tasks. For example, if the same subcontractor performed the interdependent work packages (concrete walls and door frames in the specific case), that subcontractor had incentives and opportunities to use different methods and avoid the quality issues at the interfaces and reduce the costs of the combined work.

The project management literature recognizes the importance of bid packaging for creating interest in bidding, and attracting low bids (Gould and Joyce 2009) but provides minimal guidance on how to develop effective bid package structure. General recommendations include conforming with local practices and availability of subcontractors, and avoiding second tier contractors to reduce bid costs due to double mark-up. The above discussion highlights the importance of work structuring and the need to better understand how the work structuring decisions are made and how they impact both the bid cost (and the likelihood of winning the project) and project coordination and performance.

In order to develop more systematic methodologies for generating effective work structures in construction projects, this study reviews the literature in two sectors with extensive experience in work structuring: (1) product architecture in manufacturing, and (2) software architecture. The review explores how these disciplines make the decisions to breakdown their “product” into components or modules. An initial brief review is presented here due to space limitations.

3 PRODUCT ARCHITECTURE IN MANUFACTURING

3.1 Definitions

Product architecture is defined as the way in which the functional elements of a product are configured into physical units and the way in which these units interact (Ulrich and Eppinger 1995). Product architecture involves the allocation of product function to physical components and includes (1) the arrangement of functional elements, (2) the mapping from functional elements to physical components, and (3) the specification of the interfaces among interacting physical components (Ulrich 1993). Fixson (2005) defined product architecture as a comprehensive representation of the fundamental structure of a product that includes product characteristics, type of components as well as components’ interfaces. The word component is defined as a physically distinct portion of the product that embodies a core design concept (Clark 1985). The subsystem is defined as the relative hierarchy between a currently focused system and its subordinate systems (Liu et al, 2016). Sometimes the word modules are used instead of components or sub systems. The term module is defined as the structurally independent unit from other units within the system, which when all the modules perform together to accomplish the common goal of the system (Kim et al. 2016). Ultimately, a product architecture is a model of an engineering artefact in terms of components linked by relations (Wyatt et al, 2012).

Product architecture involves the allocation of product function to physical components. During the first step of product architecture design the overall function of the product divided into subsystems of lower complexity. These subsystems are integrated through functional chains in a meaningful way to fulfill the overall function of the system (Emmatty and Sarmah 2012). These functional needs help to identify the system, subsystems, and modules to creating the product architecture. During this process, the conceptual design helps to identify and optimize the number of modules (Emmatty and Sarmah,2012).

Two critical aspects of successful product architecture are the knowledge of the components and the way in which the components are integrated and linked together into a coherent whole (Henderson et al. 1990). The linkage between the different product components plays a very important role for the efficient product architecture. Product architecture aims to reduce the interdependencies between the components by clustering of the components. According to Kim et al. (2016) the complexity of the system depends upon the number of the elements or systems.

3.2 Approaches to product architecture development

Different manufacturing industries have different approaches to product architecture. The automobile and airplane industries have the most complex and challenging product architecture because the product development process involves thousands of engineers spending years in designing, testing, and integrating hundreds of thousands of parts (Gokpinar et al 2010). Thus, the product architecture is a deliberate process in the product development involving hundreds of decisions, many of which can be usefully supported by knowledge and tools (Krishnan and Ulrich 2001). Krishnan and Ulrich (2001) divide the product development into four phases (1) concept development, (2) supply-chain design, (3) product design, and (4) production ramp-up and launch. The product architecture is considered part of concept development.

Wyatt and Wynn (2012) identify four phases in the product architecture development process: Exploration, Generation, Evaluation and Communication. They also identify formal and informal methods for developing the architecture. Informal methods depend on human creativity, previous experiences and brainstorming and include paper-based methods for conceptual design (Helms et al. 2012). In product architecture, the main goal is the identification of modules or subsystems from the functional models.

Cutherell (1996) et al., presented a method for defining a module-based product architecture in which product is decomposed into possible modules based on functional schematics. However, this method lacks a systematic technique for defining the clustering and interaction of the components. Stone et al. (2000) et al. described a “heuristic method” for identifying modules for product architecture, which requires gathering the customer needs, mapping the needs to the functional model of the product, identifying product architecture using heuristics, and generating and selecting modular concepts. The subsystems or modules are identified using heuristics regarding the grouping of the sub-functions.

Formal methods provide a systematic and potentially more reliable approach to development and optimization of product architectures (Kurtoglu and Campbell 2009). In engineering there are several methods for formal design synthesis (Wyatt et al. 2012). In morphological analysis (Zwicky 1969), a design problem is decomposed into partial problems, solutions are identified to each partial problem, and combinations of those solutions are considered. The Subjective–Objective System (Ziv-Av and Reich 2005) combines morphological analysis with an evaluation of the solution quality as a weighted sum of selected characteristics of the solution. An agent-based system (Campbell et al. 2000), uses software ‘agents’ that modify an emerging architecture according to their objectives and preferences. Another approach is Knowledge-Based Engineering (KBE) that involves capturing the steps that an expert designer would take to solve the problem in computer executable form. Product configuration systems are tool that assist with product architecture as they can guide the selection of combinations of predesigned components for a particular customer’s needs (Wyatt et. al 2012).

Helms et al., (2012) provided a general approach for computational design synthesis of product architecture to generate alternatives of product models based on Function – Behavior - Structure representation. At the Function level, the design problem is mapped and decomposed into three levels of details, i.e. overall function, high-level functions and sub-functions. At the Behavior level, a network of the sub-functions is synthesized from the physical viewpoint. At the Structural level the concrete components are connected to produce the product architecture. At that point, the modularization of components can be evaluated to reduce the complexity of the product architecture. This method generates alternative design solutions iteratively and automatically. Wyatt et al. (2012) developed a method of computational design synthesis with network structure constraints. In this method, the product architecture is expressed as a network of components linked by connections. Alternative product architectures are synthesized using constraints on the structure of the network. This enables the development of high quality solutions based on specified quality metrics.

3.3 Effects of product architecture on performance

Product architecture can affect many aspects of product and process quality, from technical performance to production costs and satisfaction of later lifecycle requirements. Product architecture has important implications for organizational performance, as it impacts five areas of managerial importance (Ulrich 1995): (1) the effort and cost of making product changes; (2) the ability to economically produce product variety; (3) the ability to use standardized components for different products; (4) the ability to optimize the product performance, or component performance; and (5) product development management, including the organizational division of the work, the managerial complexity and skills needed (planning and systems engineering vs. integration and coordination skills), and the ability to innovate (Henderson and Kim 1990). The alignment of product architecture and organizational structure can affect the performance of the product as well as the organization. Gokpinar et al (2010) used networks to characterize both product structure and communication patterns, and found that misalignment between product architecture and organizational structure results into product quality problems.

4 SOFTWARE ARCHITECTURE

4.1 Definition

Software architecture is defined as the structure of a software system, which is made up of software components, connectors, and the properties of those components (Patidar and Suman, et al,2015, Perovich & Bastarrica 2009, Bass et al. 2003). As part of the software architecture, Boer and Vliet, (2009) include the set of architecturally significant requirements and design decisions that led to the structure. The software architecture in a more general definition, is the product development that gives the highest return on investment with respect to quality, schedule, and cost (Bass et al 2003). Software architecture is considered as a decision-making process that addresses the goals, issues and concerns expressed by the system stakeholders (Pedraza-Garcia et al 2014). In the software industry, the software architecture's decision making is basically evolving the design decisions.   However, software architecture decisions are not limited to the design phase only, as they also play an important role in all stages of software development, evolution and integration (Dasanayake et al., 2015).

The goal of software architecture is to capture and meet the stakeholders' requirements and fulfill the quality attributes of the software system (maintainability, reliability, testability, performance and security). The complexity of the system is a critical consideration of the software architecture (Bass et al 2003). Thus, the interaction between different modules or components is the main concern during the software architecture process. Research in the field of software architecture aims at developing principles and models to improve not only the performance of the software, but to also make the overall development projects more cost efficient.

4.2 Approaches and methods in software architecture

Hoefmeister et al (2005) introduced a general software architecture design approach by comparing five industrial software architecture design methods. According to his findings, formation of the software architecture involves three main activities, architecture analysis, architecture synthesis and architecture evaluation. In architectural analysis, the problem is defined based on the requirements of the stakeholders. This problem is solved by selecting, prioritizing and analyzing the requirements. During the architectural synthesis, the main architectural decisions are made and the possible solution is given to the defined problem. During evaluation, the proposed solution is analyzed based on the requirements and further improvements are suggested. Different methods are adopted for the software architecture evaluation.

Kruchten (1995) developed the “4+1” view model of software architecture. This model is based on the multiple or concurrent views and allows to address separately the concerns of the various stakeholders of the architecture: end-user, developers, system engineers, project managers, etc. The Attribute-Driven Design (ADD) method was developed by the Carnegie Mellon Software Engineering Institute to design the software architecture based on quality attributes (Wojcik and Clements, 2006). Lattanze (2005) developed the Architecture Centric Design Method (ACDM). ACDM is an iterative architecture design method in which the architecture is refined until the stakeholder needs are achieved. This method documents and captures architectural risks and trade-offs, which also helps in creating better estimates and schedules of the project.

Garcia suggested a strategy to develop the activities of software architecture design by using Business Process Management Notation (BPMN). Their work mainly focuses on the systematic consideration of the decision-making activities required for the software architecture design. Barbacci et al., (2003) proposed the Architecture Trade-off Analysis Method (ATAM). ATAM was developed to record any risk, sensitive points and trade-off points within the architecture of a complex software intensive system. ATAM helps an organization develop a set of analyses, rationale and guidelines for ongoing decision making about the architecture. The Quality Attribute Workshop (QAW) is a compliment to the ATAM process, where the system stakeholders are engaged at the initial stage of the project life cycle to identify the driving quality attributes (Barbacci et al., 2003).

Managing the requirements throughout the life cycle of the project is very important. This can be done by communicating the stakeholder requirements by the operational management who aid all the project stakeholders, project managers, end users, developers and testers to continually keep noticing the requirement status and recognize the impact of changing requirements to schedules, functionality and cost (Wyatt et al. 2012). According to Babu & Ram (2016), the software architectural model is developed on the basis of risk evaluation and prediction method followed by the risk assessment stage. The module prediction strategy is applied to split up the entire requirement into sub-groups to handle the large number of stakeholder requirements. As the software architecture is the initial stage of the project planning, the risk assessment helps to identify the potential risk factors to minimize the impact of risk at the initial stage (Babu and Ram 2016).

Dasanayake et al (2015) identified two types of architecture design approaches—the Up-front design approach and the Continuous design approach. In the Up-front design approach, considerable time was spent on the design phase and the detailed architecture was developed during design, with minor changes later in the project. In the continuous design approach, the design is not fully developed at the initial stage and the project starts with the minimal design and later expanded during the project progress. This approach is followed by the smaller and agile-like projects. Nevertheless, the practitioners follow a hybrid of the above two approaches utilizing considerable time in up front design with continuously modifying the system based on the requirements change or learning during the project. A recent survey of architecture decision-making held by Ven and Bosch (2016) claimed that the role of the software architect has changed from the traditional approach of creating modules and documentation, to more of an advisory role in the current agile project. Now, for more agile projects the architectural decisions are made JIT (Just in time) by the development team itself (Konemann, 2009).

4.3 Effects of software architecture on performance

Software architecture has two important implications for performance: (1) It impacts the performance of the software, and (2) it impacts the performance of the development effort. For software, key performance metrics include performance (such as speed) reliability (errors), maintainability, testability, and security. As discussed previously, software architecture is very concerned with the trade-offs between the performance criteria. The complexity of the system is an important consideration of the software architecture (Bass 2007). Such complexity also affects the management complexity of the development project and consequently the risks of budget and schedule overruns (Fairbanks 2010, Poort & van Vliet 2012).

5 CONCLUSION

This brief review of the literature of work structuring in construction, manufacturing and software sectors identifies some starting points for further investigation and research. First, it indicates that work structuring decisions have important implications for product and project performance. Both the manufacturing and software sectors recognize the significance of product architecture and software for the performance of the product as well as the performance of the development project in terms of budget, schedule and meeting the requirements. The construction literature also provides evidence regarding the importance of bid packaging, however the influence of work structuring has not been systematically examined.

Second, the timing of work structuring decisions requires further consideration. In manufacturing, product architecture decisions are made during the design process. In software development, there is a debate about how much structure should be decided early on, as it can limit flexibility and adaptability later on. In construction, bid packaging decisions are typically made during the procurement process, and rarely at the design phase, unless there are special requirements, such as modularization.

Third, the manufacturing and software sectors have developed systematic methods to support product architecture decisions. The process of work structuring involves 3 phases: (1) Analysis of the system (product or software) into components, (2) Synthesis of the components into modules or units, and (3) Evaluation of the architecture. In construction, the analysis is performed using the Work Breakdown Structure. However, the synthesis of work scope elements into bid packages depends exclusively on the practitioners' experience without any systematic decision-support.

Finally, in software and product architecture, the evaluation of the structure is based on systematic methodologies, including scenario-based, mathematical modeling, and simulation (Patidar and Suman 2015, Roy and Graham 2008), as opposed to construction where the evaluation of bid packaging is experience-based.

In conclusion, construction research needs to develop systematic methodologies to evaluate alternative work structuring configurations for construction projects, and their implications for project performance. Currently, construction practitioners make such decisions on every project based on their experience and skill. Thus, a systematic methodology could make a significant contribution to project management theory and practice.

6 REFERENCES

Babu, R.T.K.S. and Ram, N.S. 2016. Improvement of the dependency structure of the software architecture model with risk estimation. Journal of Statistical Computation and Simulation, 86(5), 908-921, DOI: 10.1080/00949655.2015.1042379

Barbacci, M.R., Ellison, R., Lattanze, A., Stafford, J.A., Weinstock, C.B. and Wood, W.G. 2003. Quality Attribute Workshops (QAWs). Technical report, 3rd ed., School of Computer Science, Camegie Mellon University, CMU/SEI-2003-TR-016, ESC-TR-2003-016.

Bass, L., Clements, P. and Kazman, R. 2003. Software Architecture in Practice. 2nd edition reading, Addison-Wesley, MA, USA.

Boer, R.C. de and Vliet, H.N. 2009. On the similarity between requirements and architecture. *The Journal of Systems and Software*, 82: 544-550.

Brotherton S.A., Fried R.T., Norman E.S. 2008, Applying the Work Breakdown Structure to the Project Management Lifecycle , PMI Global Congress Proceedings, Denver, Colorado, USA.

Campbell MI, Cagan J, Kotovsky K (2000) Agent-based synthesis of electromechanical design configurations. J Mech Des 122:61–69.

Chen, Q., Reichard, G., & Beliveau, Y. (2008) Multiperspective Approach to Exploring Comprehensive Cause Factors for Interface Issues" Journal of Construction Engineering and Management, ASCE, 134(6): 432-441.

Chua DK and Godinot M (2006) "Use of a WBS Matrix to Improve Interface Management in Projects" J. of Construction Engineering and Management, ASCE 132(1): 67–79.

Clark, K. B. 1985. The interaction of design hierarchies and the market concepts in the technological evolution. Research Policy, Elsevier Science, 14: 235-251.

Crowston K. (1991) Towards a coordination cookbook: Recipes for multi-agent action. Cambridge, EUA, MIT, Sloan School of Management, PhD Dissertation.

Cutherell, D. 1996. Product Architecture' in The PDMA Handbook of New Product Development. John Wiley and Sons, New York, USA.

Dasanayake, S., Markkula, J., Aaramaa, S. and Oivo, M. 2015. Software Architecture Decision-Making Practices and Challenges: An Industrial Case Study. 24th Australasian Software Engineering Conference (ASWEC), IEEE, Adelaide, Australia, 88-97.

Draper, J.D. & Martinez, J. 2002, 'The Evaluation of Alternative Production System Designs With Discrete Event Simulation' In:, Formoso, C.T. & Ballard, G., 10th Annual Conference of the International Group for Lean Construction, Aug. 6-8, Gramado, Brazil. http://www.iglc.net/Papers/Details/181

Emmatty, F. and Sarmah, S. 2012. Modular product development through platform-based design and DFMA. Journal of Engineering Design, Vol. 23(9):696–714.

Fairbanks, G. (2010). Just Enough Software Architecture. Marshall & Brainerd.

Fixson, S.K. 2005. Product architecture assessment: A tool to link product, process, and supply chain design decisions. Journal of operation management, 23(3):345-369.

Formoso, C. T., and Isatto, E. L. (2009) "Production Planning and Control and the Coordination of Project Supply Chains" in O'Brien, W. J., Formoso, C. T., Vrijhoef, R., and London, K. A (eds) Construction Supply Chain Management Handbook, pp. 3.1–3.25, CRC Press, Boca Raton.

Globerson, S. 1994. Impact of various work-breakdown structures on project conceptualization, *International Journal of Project Management*, 12 (3): 165-171.

Gokpinar, B., Hopp, W.J. and Iravani, S.M.R. 2010. The Impact of Misalignment of Organization Structure and Product Architecture on Quality in complex Product Development. Management Science, 56(3): 468-484.

Gould, F., and Joyce, N. (2009) Construction Project Management" Third Edition, Pearson Prentice Hall, New Jersey.

Helms, B. and Shea, K. 2012. Computational Synthesis of Product Architectures Based on Object-Oriented Graph Grammars. *Journal of Mechanical Design*, ASME, 134(2): 1-14.

Henderson, R.M. and Clark, K.B. 1990. Architectural Innovation: The reconfiguration of existing. *Administrative Science Quarterly*, 35(1): 9-30.

Hofmeister, C., Kruchten, P., Nord, R.L. 2005. Generalizing a Model of Software Architecture Design from Five Industrial Approaches. *Proceedings of the 5th Working IEEE/IFIP Conference on Software* Architecture (WICSA5), Pittsburgh, PA, 1-10.

Kim, G., Kwon, Y., Suh, E.S. and Ahn, J. 2016. Analysis of Architectural Complexity for Product Family and Platform. J. Mech. Des. 2016; 138(7):071401-071401-11.

Konemann, P. 2009. “Integrating decision management with UML modeling concepts and tools,” Joint *Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, Cambridge, 297–300.

Krishnan, V. and Ulrich, K. 2001. Product Development Decisions: A Review of the Literature. *Management Science*, 47(1): 1–21.

Kuprenas, J. A., and Rosson, M. (2000). “Interface consideration on multiple prime contractor construction projects.” In Proceedings of Construction Congress VI: Building Together for a Better Tomorrow in an Increasing Complex World, Walsh K (ed.), ASCE, Feb 20-22, Orlando, Florida, pp. 1093–1102.

Kurtoglu, T., and Campbell, M. I., 2006. “A Graph Grammar Based Framework for Automated Concept Generation,” Proceedings of 9th International Design Conference, DESIGN, pp. 61–68.

Lattanze, J.L. 2005. The Architecture Centric Development Method (ACDM). Technical report, School of Computer Science, Camegie Mellon University, CMU-ISRI-05-103.

Liu, C., Hildre, H.P., Zhang, H. and Rølvag, T. 2016. Product architecture design of multi-modal products, *Res Eng Design*, 27:331–346.

Malone, T. W. and Crowston, K. (1994) “The Interdisciplinary Theory of Coordination” ACM Computing Surveys, 26(1): 87-119.

March, J. G. and Simon, H. A. (1958) Organizations. John Wiley & Sons, New York, NY.

Mintzberg, H. (1980) “Structure in 5s: A Synthesis of the Research in Organization Design” Management Science, 26(3): 322-341.

Mitropoulos, P. and Sanchez, R. (2016) “Criteria and Considerations for Project Work Structuring” Management, Procurement and Law, ICE, Vol 169(3), pp. 124-130.

Nicolini, D., Holti, R., Smalley, M. (2001). “Integrating project activities: The theory and Practice of Managing the Supply Chain Through Clusters.” Construction Management & Economics, 19(1):37-47.

Nooteboom, U. (2004). “Interface management improves on-time, on-budget delivery of megaprojects.” JPT Online, 56(8), Society of Petroleum Engineers, http://dx.doi.org/10.2118/0804-0032-JPT

Patidar, A., and Suman, U. 2015. A Survey on Software Architecture Evaluation Methods. *2nd International Conference on Computing for Sustainable Global Development,* IEEE, 967-972.

Pavitt, T. C., and Gibb, A. G. F. (2003). “Interface management within construction: In particular, building façade.” Journal of Construction Engineering and Management, ASCE 129(1): 8–15.

Pedraza-Garcia, G., Astudillo, H. and Correal, D. 2014. Modeling Software Architecture Process with a Decision-Making Approach. *33rd International Conference of the Chilean Computer Science Society,* 1-6.

Perovich, M. Bastarrica, M.C. Rojas, C. 2009. Model-Driven approach to Software Architecture design. *ICSE Workshop on Sharing and Reusing Architectural Knowledge*, 1-8.

Poort, E. R.; and van Vliet, H (2012). "RCDA: Architecting as a risk- and cost management discipline". The Journal of Systems and Software. Elsevier. 85 (9): 1995–201.

B. Roy, B. and Graham, T.C.N. (2008) “Methods for Evaluating Software Architecture: A Survey,” School of Computing TR 2008-545, Queen’s University.

Stone, R.B., Rolla, M., Wood, K.L. and Crawford, R.H. 2000. A heuristic method for identifying modules for product architectures. *Elsevier Science*, 21: 5-31.

Tommelein, I. and Beeche, G. (2001). "De-coupling cladding installation from other high-rise building trades: A case study". Proceedings of the 9th conference of the international group for lean construction, Singapore, 6-8 August.

Tsao C. C. Y., Tommelein, I D; Swanlund E. S.; and Howell, G. A. (2004) "Work Structuring to Achieve Integrated Product–Process Design" Journal of Construction Engineering and Management, ASCE130(6): 780–789.

Ulrich, K. 1995. The role of product architecture in manufacturing firm. Research Policy, Elsevier Science, 24:419-440

Ulrich, K.T. and Eppinger, S.D. 1995. Product design and development, McGraw-Hill, NewYork, NY, USA.

Ven, J. S. v. d. and Bosch J. 2016. "Busting Software Architecture Beliefs: A Survey on Success Factors in Architecture Decision Making, 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Limassol, 2016, pp:42-49.

Vrijhoef, R., and Koskela, L. (2000) "The four roles of supply chain management in construction" European Journal of Purchasing & Supply Management 6 (3): 169-178.

Wojcik, R., Bachmann, F., Bass, L., Clement, P., Merson, P., Nord, R. and Wood, B. 2006. Attribute-Driven Design (ADD). Technical report, CMU/SEI-2006-TR-023, ESC-TR-2006-023, School of Computer Science, Carnegie Mellon University.

Wyatt, D.F., Wynn, D., Jarrett J. and P. Clarkson, J. 2012. Supporting product architecture design using computational design synthesis with network structure constraints. *Res Eng Design*, 23

Wyatt, D.F., Wynn, D., Jarrett J. and P. Clarkson, J. 2012. Supporting product architecture design using computational design synthesis with network structure constraints. Res Eng Design, 23:17–52.

Ziv-Av A, Reich Y (2005) SOS–subjective objective system for generating optimal product concepts. Des Stud 26:509–533

Zwicky F (1969) Discovery, invention, research–through the morphological approach. Macmillian, Toronto