Modelling Collaborative Software Development Using Axiomatic Design Principles

Jbid Arsenyan, Gülçin Büyüközkan

***Abstract*—Even though software development (SD) is an intrinsically collaborative process; with the emerging importance of collaborative product development (CPD), collaborative processes gained significant consideration. Collaborative software development (CSD) encompasses an increasingly competitive position with its dynamic and innovative structure, as the complex nature of SD makes collaboration indispensable.**

This paper presents an Axiomatic Design (AD) based CSD structure exploring factors affecting SD as well as CPD dynamics, in order to offer a guideline to increase success and effectiveness of collaborative efforts in SD.

***Index Terms*—Software Development, Collaboration, Axiomatic Design.**

I. Introduction

More than ever, organizations have been facing the challenge of improving the quality of their work processes as a strategy to remain alive and competitive and many companies are struggling to reengineer, automate, and improve the way they perform their business [1]. Software Development (SD) has been a challenging task for several decades [2] and the increasing complexity of software development (SD), growing demands for different kinds of software as well as the ongoing globalization require more efficient SD processes [3].

The development of large and complex software systems is considered to be a teamwork process that requires support for coordinating cooperative activities, maintaining project control and sharing information [4]. While Collaborative Product Development (CPD) develops into a business strategy creating competitive advantage for organizations in product development (PD) by creating a collaborative environment to share risks, reduce time to market and innovate; SD collaborations also develop into a more structured activity, since mastering large SD projects becomes even more complex, not only because projects grow larger, but also because software teams are increasingly distributed across space and time due to globalization and internationalization [4]. Supporting efficient knowledge collaboration and transfer is thus essential for SD organizations to remain competitive [5]. While the software industry deals with the ever-increasing complexity of its products, collaboration among different people participating in the same development project is essential and has already been considered as an everyday part of professional SD [6].

However, CSD literature lacks of a systematic design of the process from a strategic point of view. The aim of this paper is to design a CSD structure by identifying its dynamics and developing a model through these dynamics. Literature offers many system design tools such as Quality Function Deployment (QFD), Axiomatic Design (AD), Design for X and TRIZ. Among these methodologies, AD is an appropriate tool for the design of non-engineering systems such as business plans and organizations [7]. Hence, a CSD structure with a model based on AD is proposed in this paper, while defining the dynamics, including goals, strategies and methodologies, which affect the collaborative efforts in SD. The proposed model forms a guideline for CSD from a strategic point of view, investigating different goals to attain in collaborative efforts in SD. To our knowledge, no previous study exists which explore CSD based on system design.

The paper is organized as follows: in the next section, CSD is described briefly. Third section reviews AD and presents the proposed model. The paper is concluded with a few remarks.

II. Collaborative Software Development

SD is a combined process of research, development, modification, re-use, reengineering, maintenance and similar activities that result in software products. However, SD process and projects have a long and storied history of failure, where 82% of projects today run late, while errors cost 80% of the average project budget to fix according to The Standish Group. The growing complexity of software systems and the constant extension of new requirements necessitate the cooperation of multiple persons such as analysts, developers, testers, and customers [4]. While the SE industry deals with the ever-increasing complexity of its products, collaboration among different people participating in the development project is essential and has already been considered as an everyday part of professional SD [8]. Collaboration helps SD teams to handle large software systems by knowledge sharing and communication. Also human-centric SD methods, such as Extreme Programming and other agile methods as well as internet-based multi-site cooperation tools that support remote CSD have been developed and implemented to deal with this complexity [6], which benefits its participants in time to market, reusability, robustness, extensibility,

Manuscript received March 23, 2009. This work was financially supported by Galatasaray University Research Fund.

Jbid Arsenyan is with the Industrial Engineering Department, Bahcesehir University, Istanbul, 34100, Turkey (phone: +90.212.381.0855; e-mail: jbid.arsenyan@bahcesehir.edu.tr).

Gülçin Büyüközkan is with the Industrial Engineering Department, Galatasaray University, Istanbul, 34357, Turkey (phone: +90.212.227.4480; e-mail: gulcin.buyukozkan@gmail.com).

testability, and/or adaptability [8].

Some of the most important differences between traditional SD and collaborative SD (CSD) are organizational culture, management, technical platform and development team, and social and cultural issues [9]. Therefore, CSD encloses many challenges as it involves geographically distributed teams working within different units.

Improving awareness information about work processes and about the collaboration intrinsic may help SD teams to better accept the idea of defining, standardizing and continuously improving their work [1]. In this context, this study offers a structured approach for CSD in order to implement a defined work process.

III. Modelling CSD

A. Axiomatic Design

AD is first introduced by Suh in 1990 with the goal to establish a scientific basis to improve design activities by providing the designer with a theoretical foundation based on logical and rational thought process and tools [10]. Its applications include many areas such as software design [11], quality system design [12], general system design [13; 14], manufacturing system design [15; 16], ergonomics [17], engineering systems [18; 19], and office cell design [20]. AD is generally applied to the design of physical entities. However, there exist studies that employ AD to design intangible systems such as e-commerce strategies [7] and e-commercial web-sites [21].

According to AD, the world of design has four domains: customer domain with characteristic vector of customer attributes (CAs), functional domain with characteristic vector of functional requirements (FRs), physical domain with characteristic vector of design parameters (DPs), and process domain with characteristic vector of process variables (PVs), as seen in Fig 1. The domain on the left represent "what we want to achieve" and the domain on the right corresponds to "how we want to achieve it". The transition between left to right occurs through mappings [12] as seen in Figure 1.

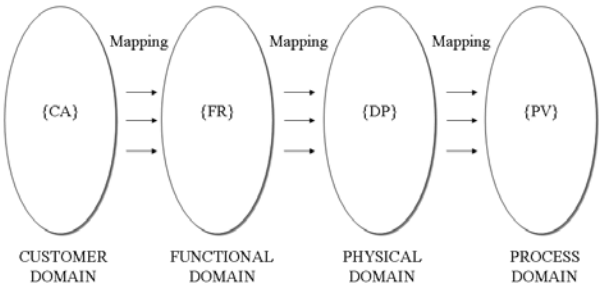

Figure 1. Four domains of design

AD consists of two axioms: independence axiom that demands to maintain the independence of the FRs and information axiom that states that the design with the minimum information content is the best design.

During the mapping process where first level CA, FR, DP and PV are decomposed into hierarchies, the independence axiom must be satisfied. Moreover, while decomposing, zigzagging between the design domains is required [12]. The independence axiom can be also defined as the case where DPs and FRs are related in such a way that a specific DP can be adjusted to satisfy its corresponding FR without affecting other FRs [7].

Mapping and decomposing represent an important phase of the AD. Sub-levels of FRs and DPs are connected through zigzagging while maintaining the independence. The relation between FRs and DPs can be expressed as follows:

$$\{FR\} = [A] * \{DP\}$$

$\{FR\}$ and $\{DP\}$ represent the functional and physical vectors, respectively; whereas $[A]$ is the design matrix that displays the relation between each FR and DP.

The independence of FRs is defined by the structure of the design matrix. To assure the independence, the design matrix should be either diagonal or triangular [10].

There exist three types of design: coupled, decoupled, and uncoupled. When $[A]$ is diagonal as seen in Figure 2, the design is called an uncoupled design and each of the FRs can be satisfied independently by means of one DP. However, this represents an ideal design and it cannot always be achieved. Decoupled design is represented by a triangular design matrix as seen in Figure 3, in which case the independence of FRs can be satisfied if and only if the DPs are determined in a proper sequence. Any other form of $[A]$ as displayed in Figure 4 is called a coupled design and it should be avoided as the design cannot guarantee the independence axiom [10].

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 \\ 0 & X & 0 \\ 0 & 0 & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix}$$

Figure 2. Uncoupled design

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 \\ X & X & 0 \\ X & X & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix}$$

Figure 3. Decoupled design

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{bmatrix} X & 0 & X \\ 0 & X & X \\ X & X & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix}$$

Figure 4. Coupled design

AD does not possess wide applications in CPD literature. AD is applied to CPD architecture to develop a teamwork platform [22]. However, literature does not offer a systematic approach for CSD requirements. The originality of the proposed work lies in the application of AD principles to determine CSD requirements and designing a structure highlighting SD requirements in order to form a CSD guideline for partners in the beginning of the collaboration.

AD approach is employed in order to develop a CSD model, given that the use of AD in strategic formulation and business planning assures a strong relationship between the goals and strategies defined [7]. AD provides a basis to define the path from what is aimed to how this goal should be achieved.

B. AD based CSD model

SD process is often introduced by considering 'why', 'what' and 'how': 'why' is defined by whoever commissions the project, the architect's primary concern in to specify 'what' must be done, and 'how' it is done is the software engineer's province [23]. In this study; 'why', 'what' and 'how' are transformed into CAs, FRs and DPs, translating goals, strategies and methodologies.

The proposed model consists of three levels. The first level of AD describes the three main domains of the CSD: customer domain, functional domain, and physical domain. The variables defined as CAs represent the goals of the CSD efforts. FRs correspond to the strategies needed to be implemented to achieve these goals. Subsequently, the methodologies and tools used to implement these strategies are symbolized by DPs. Basically, in this first level of CSD structure design, initial goal, strategy and framework are defined. Initial goal is set to be "Inter/intra-firm collaborations for SD" since the main objective of the model is to provide guidelines for successful SD collaborations. In order to have a successful collaboration, initial strategy must be to define a collaboration strategy to cover all possible collaboration areas. Consequently, CSD framework, including all methodologies and tools applied to CSD are collected to satisfy the collaboration strategy. The variables defined for the starting point are as follows:

CA_0 = Inter/intra-firm collaborations for SD

FR_0 = Define a CSD strategy

DP_0 = CSD

It is important to become aware of the three different types of goals to better understand a participative system, which are individual goal, collective goal, and project goal [24]. In this study, these goals are translated as effective partnership process, effective collaboration process, and effective SD process, respectively, in the first level decomposition of the model. It is necessary to understand what the customer wants most in supportability and to align the capability of the organization to provide it [25]. Therefore, the starting point of the model is the customer domain, where the strategic goals of the CSD system are clearly defined.

Figure 5 displays the proposed three-level AD based CSD model including mapping between functional and physical domains, as well as FR-DP zigzagging. Lines represent the decomposition into hierarchies, while arrows symbolize the zigzagging between domains. The decompositions and the hierarchical structure will be elaborated subsequently.

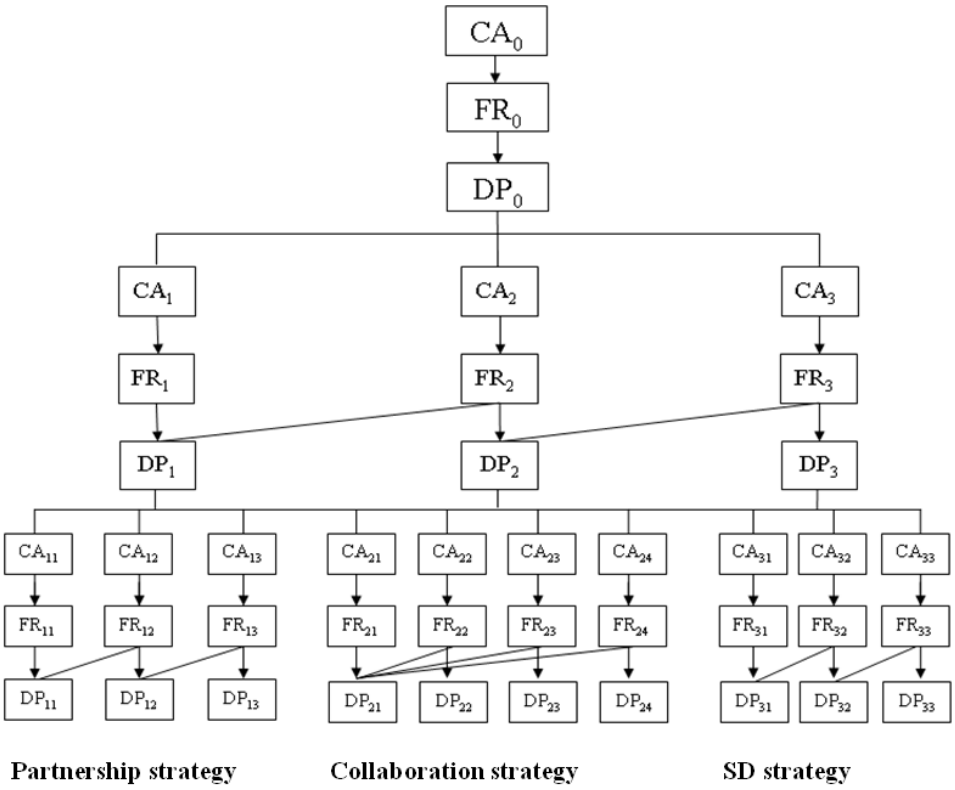

Figure 5. Proposed AD-based CSD model

The matrices presented throughout the decompositions symbolize the relationships between the strategies and the methodologies. The matrices for the relationships between goals and strategies are not demonstrated given that each strategy responds to only its corresponding goal and therefore, independence is naturally achieved.

Level 1 Decomposition

First level goal is decomposed into three sub-goals:

CA_1 = Effective partnership process

CA_2 = Effective collaboration process

CA_3 = Effective SD

Effective SD is an essential goal of CSD efforts. On the other hand, CSD includes concurrently both collaboration process and partnership process. Collaboration denotes all collaborative activities such as communication, information sharing, and interaction whereas partnership process includes phases such as identification, selection of partners and partnership sustainability.

FR_1 = Define effective partnership strategy
FR_2 = Define effective collaboration strategy
FR_3 = Define effective SD strategy

Strategies to support these goals are determined accordingly. Partnership strategy should be based upon a win-win situation whereas collaborative technologies build collaboration strategy. Some authors emphasize the win-win approach in collaboration: A win-win model assumes that the success of CSD projects depends on all involved parties to positively gain benefits from the collaboration and the different requirements to be equally represented [3].

DP_1 = Game Theory
DP_2 = Collaborative technology
DP_3 = Software engineering management

Different cooperation and competition strategies emerge according to the level of competition and cooperation between the "players" [26]. Therefore Game Theory is applied in order to model an effective partnership strategy. On the other hand, for an effective collaboration strategy, a network based on collaborative technologies must be employed in order to realize the integration of the collaboration teams. SD strategy should be defined by Software Engineering Management, which can be described as the application of management activities – planning, coordinating, measuring, monitoring, controlling, and reporting – to ensure that the development and maintenance of software is systematic, disciplined, and quantified.

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 \\ X & X & 0 \\ 0 & X & X \end{bmatrix} * \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix}$$

Figure 6. Level 1 Decomposition

As seen in Figure 6, a decoupled design is obtained in the Level 1 decomposition. This is due to the fact that implementing these methodologies requires a particular sequence, i.e. collaboration strategy is applied after the Game Theory is implemented and SD strategy is dependent of collaborative technology.

Level 2.1 Decomposition

Effective partnership process is decomposed into three sub-goals and these sub-goals present the stages of the partnership process.

CA_{11} = Identification of potential partners
CA_{12} = Formation of partnership
CA_{13}= Management of partnership

Initially, identification of potential partners is required in order to evaluate and recognize partner[s] suitable for collaboration. Subsequently, the next stage is the formation of partnership including negotiation, selection and initialization. However, effective partnership process also requires an effective management of partnership.

FR_{11} = Proactive assessment
FR_{12} = Negotiation strategy
FR_{13} = Sustainability of partnership

During the mapping process, strategies are determined accordingly. First strategy when identifying potential partners is to act proactively, rather than reacting to partnership proposals. Literature offers several studies applying decision analysis to select collaborative partners [27; 28]. A negotiation strategy has to be implemented in order to collaborate on pre-defined ground rules. The challenge of CSD is organizing the value creation process through the coordination of their respective resources across organization boundaries and consequently, contracts serve as a coordinating device, clarifying mutual expectations, enabling goal correspondence, and establishing a basis for shared common ground [29]. Next strategy is to seek sustainability of partnership and repetitiveness rather than to form ephemeral alliances. Long-term commitment can maintain a partnership relationship with competitors and can neutralize possible conflicts [26]. Partnership management is applied in order to maintain the collaboration between the parties.

DP_{11} = Decision analysis
DP_{12} = Contract management
DP_{13} = Partnership management

$$\begin{Bmatrix} FR_{11} \\ FR_{12} \\ FR_{13} \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 \\ X & X & 0 \\ X & X & X \end{bmatrix} * \begin{Bmatrix} DP_{11} \\ DP_{12} \\ DP_{13} \end{Bmatrix}$$

Figure 7. Level 2.1 Decomposition

Similarly to the Level 1; in this sub-level a decoupled design is observed as seen in Figure 7. This is essentially caused by the fact that strategies in partnership process are implemented progressively, each methodology at a time. This results in a triangular design matrix, where the each DP affects its successors.

Level 2.2 Decomposition

Effective collaboration process differs from effective partnership process given that this goal focuses on the profits acquired by CSD efforts. Four sub-goals are determined for effective collaboration process.

CA_{21} = Coordination
CA_{22} = Trust
CA_{23} = Co-learning
CA_{24} = Co-innovation

Coordination, which can be defined as making different people work together for a goal, is naturally the main goal of collaboration process. SD is a cooperative game of invention and communication [30]. CSD concerns both methods and tools supporting the communication and coordination requirements within a distributed SD process, which is essential for planning, execution, and coordination of all task-based as well as organizationally, temporally and spatially distributed activities incorporating all process- and product-related activities [3]. Information sharing is another important characteristic of collaboration process [9]. All these issues are handled within coordination goal.

In CSD, trust is required in order for developers to collaborate effectively given that without trust, developers will not normally share transient information and permanent knowledge, thus limiting their productive capacity [8]. Therefore; trust is defined as another goal to achieve in collaboration process. On the other hand, co-learning; whether corporate, individual or technical; is another goal to attain in effective collaboration process in order to benefit from the synergy emerged in collaboration. Co-learning can be defined as the knowledge, experience and expertise gathered through the collaboration process, and therefore it is

included in collaboration strategy. Another goal for collaboration process is co-innovation, as innovation is a value-adding by-product of the collaboration process.

Strategies related to these goals are as follows:

FR_{21} = Communicate

FR_{22} = Establish a culture of trust

FR_{23} = Implement a learning system

FR_{24} = Generate flexible environment

Communication, the strategy for coordination, is implemented through information technologies to support coordination of development teams. Coordination concentrates on planning, decision making, organizing, and controlling the work of others, and on the loose interaction between people and hence, a common language for communication is crucial for efficient information exchange [31]. Therefore, information technologies are implemented in order to provide the common language required.

Even though trust represents a challenging goal to accomplish between various partners, managers can expect significant rewards by establishing a culture of trust in their partnerships [32]. Therefore, establishing a culture of trust, cultivated by Information Technologies, is implemented through the application of Conflict Management. As a result of intensive cooperation with competitors, conflict inevitably occurs and hence, a good conflict management system, which enables gathering information, understanding context and participating in decision making, enhancing the capacity to deal with conflict before it escalates, is needed to maintain successful collaboration [26].

A learning system has to be implemented as a strategy for co-learning, and knowledge management is the associated methodology. Knowledge management attempts to ensure growth and continuity of performance by protecting critical knowledge at all levels, applying existing knowledge in all pertinent circumstances, combining knowledge management in synergistic ways, continuously capturing, managing, and sharing relevant knowledge, and developing new knowledge through continuous learning that builds on internal experiences and external knowledge [33].

Finally, to promote co-innovation, strategy to follow is to generate a flexible environment for collaborative team to work in. Innovation management is determined as the related methodology. Workplace flexibility constitutes a platform to base and build other levels of flexibility in the firm to deploy employees' embedded knowledge and access external knowledge needed to innovation capabilities [34].

DP_{21} = Information Technologies

DP_{22} = Conflict management

DP_{23} = Knowledge management

DP_{24} = Innovation management

$$\begin{Bmatrix} FR_{21} \\ FR_{22} \\ FR_{23} \\ FR_{24} \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 & 0 \\ X & X & 0 & 0 \\ X & 0 & X & 0 \\ X & 0 & 0 & X \end{bmatrix} * \begin{Bmatrix} DP_{21} \\ DP_{22} \\ DP_{23} \\ DP_{24} \end{Bmatrix}$$

Figure 8. Level 2.2 Decomposition

As seen in Figure 8, Level 2.2 Decomposition results also in a triangular matrix. However, this structure does not represent a process. Instead, the importance of communication strategy is highlighted within the design matrix, as the main support for the collaboration process is the information technologies implemented. Therefore, each strategy is related with DP_{21}. On the other hand, remaining FRs are independent and executed separately.

Level 2.3 Decomposition

As SD differs considerably from classical PD approach, which essentially includes design, manufacturing, and marketing; an industry-specific model is developed to design the SD process. "Effective SD" is decomposed into SD customers' needs: effective understanding of needs, effective design, and effective performance.

CA_{31} = Effective understanding of needs

CA_{32} = Effective design

CA_{33} = Effective performance

To satisfy these goals, an iterative approach consisting of three steps is adopted for designing effective SD process. The strategies to meet the customer attributes are defined as follows:

FR_{31} = Define requirements

FR_{32} = Design

FR_{33} = Maintain

The first goal in the customer domain is the effective understanding of user needs. The success of a software system is measured with how well it responds to the purpose of customer. Requirements Engineering (RE) is therefore the appropriate methodology to implement to define requirements, as RE is the process of discovering the purpose of the software by identifying customer needs and documenting these in a form that is amenable to analysis, communication, and subsequent implementation [35].

The next strategy is to design effective software to fulfil the effective design goal. The design decisions include the choice of programming paradigm, architectural style, application framework, component-based software engineering standards, and design principles, as well as assumptions that may lead to architectural mismatch [36]. In SD coding, testing and debugging, and what typically are called software design is still part of design [37].Therefore, software design framework is defined as the appropriate methodology to fulfil the design strategies.

Naturally, effective performance represents another goal for the SD process. Software maintenance is essential in achieving global software competitiveness [25]. Therefore, the appropriate strategy to follow is to maintain the software after it is implemented, through software maintenance methodologies. Software maintenance is different from hardware maintenance because software doesn't physically wear out, but often gets less useful with age and software maintenance is the process of modifying existing operational software while leaving its primary functions intact [38].

DP_{31} = Requirements engineering

DP_{32} = Software design

DP_{33} = Software maintenance

$$\begin{Bmatrix} FR_{31} \\ FR_{32} \\ FR_{33} \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 \\ X & X & 0 \\ X & X & X \end{bmatrix} * \begin{Bmatrix} DP_{31} \\ DP_{32} \\ DP_{33} \end{Bmatrix}$$

Figure 9. Level 2.3 Decomposition

A decoupled design is observed in Figure 9 for Level 2.3 Decomposition once more, since SD is represented as a process. Each FR is dependent of its predecessor, however in a proper sequence. Therefore, the achievement of each goal stimulates the realization of the next one.

IV. Conclusion

This paper presented an AD based collaboration model within the context of software industry. A three level hierarchic structure for CSD was developed defining strategic goals to attain in collaborative efforts for SD, strategies to follow to fulfil these goals, and methodologies to implement to realize the strategies. CSD does not differ from the CPD in partnership process and collaboration process. However, SD process has its specific steps and consequently, requires customized evaluation of its strategies and methodologies. The aim of this model was to present a strategic framework for implementing a collaborative structure for SD in software industry.

References

[1] Araujo, R.M., Borges, M.R.S., 2007. The Role of Collaborative Support to Promote Participation and Commitment in Software Development Teams. Software Process Improvement and Practice, 12, 229–246.

[2] Eberlein, A., Jiang, L., 2007. Description of a process development methodology. Software Process: Improvement and Practice 12(1), 101-118.

[3] Hildenbrand, T., Rothlauf, F., Geisser, M., Heinzl, A., Kude, T., 2008. Approaches to Collaborative Software Development. International Conference on Complex, Intelligent and Software Intensive Systems.

[4] Altmann, J., Pomberger, G., 1999. Cooperative Software Development: Concepts, Model and Tools. Proceedings of the TOOLS-30 conference, 1 – 5 August, USA, IEEE Society Press, 194 – 277.

[5] Ye, Y., Yasuhiro Yamamoto, Kouichi Kishida, 2004. Dynamic Community: A New Conceptual Framework for Supporting Knowledge Collaboration in Software Development. Proceedings of the 11th Asia-Pacific Software Engineering Conference APSEC'04, 472-481.

[6] Hadar, I., Sherman, S., Hazzan, O. 2008. Learning Human Aspects of Collaborative Software Development. Journal of Information Systems Education, 19(3), 311-319.

[7] Martin, S.B., Kar, A.K., 2002, Axiomatic Design for the development of enterprise level e-commerce strategies, Proceedings of ICAD2002, Second International Conference on Axiomatic Design, MIT, Cambridge, USA, June 10-11.

[8] Che, H., Zhao, D., 2005. Managing Trust in Collaborative Software Development. Software Engineering Conference APSEC 2005 workshop, Dec. 17, Taipei, Taiwan.

[9] Nayak, M.K., Suesaowaluk, P., 2008. Risk Factors that Affect Collaborative Software Development. Proceedings of the 2008 IEEE ICMIT.

[10] Suh, N.P., 2001, Axiomatic Design: Advances and Applications, New York: Oxford University Press.

[11] Kim, S.J., Suh, N.P., Kim, S., 1991, Design of software systems based on AD, Annals of CIRP, 40(1), 165–70.

[12] Suh, N.P., 1995a, Designing-in of Quality Through Axiomatic Design, IEEE Transactions On Reliability. 44 (2), 256–64

[13] Suh NP., 1995b, Design and operation of large systems. Annals of CIRP , 44 (3), 203–13.

[14] Suh NP., 1997, Design of systems. Annals of CIRP, 46(1):75–80.

[15] Suh, N.P., Cochran, D.S., Paulo C. L., 1998, Manufacturing System Design, Annals of the. CIRP, 47 (2), 627-639

[16] Cochran, D.; Campinas, J.D.; Lobo, C.; Lima, P., 2001. Using axiomatic design to support the development of a balanced scorecard. International Journal of Business Performance Management, 3(2-4), 154-166.

[17] Helander, M.G., Lin, L., 2002, Axiomatc design in ergonomics and an extension of the information axiom, Journal of Engineering Design, 13(4), 321-339.

[18] Guenov, M.D., Barker, S.G., 2005, Application of axiomatic design and design structure matrix to the decomposition of engineering systems, Systems engineering, volume 8, no1, pp. 29-40.

[19] Thielman, J., Ge, P., 2006,, Applying axiomatic design theory to the evaluation and optimization of large-scale engineering systems, Journal of Engineering Design, 17 (1), 1-16.

[20] Durmusoglu, M., Kulak. O., 2008, A methodology for the design of office cells using axiomatic design principles. Omega, 36, 633 – 652.

[21] Yenisey, M.M., 2007, Axiomatic Design Approach for E-Commercial Web Sites, Lecture Notes in Computer Science, 4550, 308-315.

[22] Hou, L., Han, D., Wen, Z., Chen, F., 2007, Integrated Management System for Product Collaborative Development Chain, IEEE International Conference on Control and Automation, May 30-June 1, Guangzhou, China

[23] Walters, H.R., 2004. Structuring professional cooperation. Information and Software Technology, 46, 415–421.

[24] Nakakoji, K. Yamada, K. Giaccardi, E., 2005. Understanding the nature of collaboration in open-source software development. Software Engineering Conference APSEC '05. 15-17 Dec. 2005, Taipei, Taiwan.

[25] O'Neill, D., 1998. Software Maintenance and Global Competitiveness. Journal of Software Maintenance: Research and Practice, 9 (6), 379 – 399.

[26] Chin, K.-C., Chan, B.-L., Lam, P.-K., 2008, Identifying and prioritizing critical success factors for coopetition strategy, Industrial Management & Data Systems, Volume 108, No. 4, 437-454.

[27] Hacklin, F., Marxt, C., Fahrni, F., 2006, Strategic venture partner selection for collaborative innovation in production systems: A decision support system-based approach, International Journal of Production Economics, 104, 100–112.

[28] Yoshimura, M., Izui, K., Kida, S., 2005, Decision Support System for Selecting Collaborative Product Development Partners, Concurrent Engineering, Volume 13 (1), 5-11.

[29] Madhok, A., Mellewigt, T., Weibel, A., 2004. Trust and Formal Contracts in Interorganizational Relationships- Substitutes and Complements. Available at SSRN: http://ssrn.com/abstract=611161.

[30] Cockburn, A., 2002. Agile Software Development. Addison-Wesley.

[31] Fagerstrom, B., Jackson, M., 2002. Efficient collaboration between main and sub-suppliers. Computers in Industry, 49, 25–35.

[32] Bstieler, L., 2006, Trust Formation in Collaborative New Product Development, Journal of Product Innovation Management, Volume 23, 56-72.

[33] Chen, Y.-J., Chen, Y.M., Chu, H.C., 2008. Enabling collaborative product design through distributed engineering knowledge management. Computers in Industry, 59, 395–409.

[34] Martínez-Sánchez, A., Vela-Jiménez, M.-J., Pérez-Pérez, M., de-Luis-Carnicer, P., 2008, Workplace flexibility and innovation: The moderator effect of inter-organizational cooperation, Personnel Review, Volume 37, No. 6, 647– 665

[35] Juran, J.M., 1998. Quality in Software Development. McGraw-Hill Professional , page 20.8-20.12.

[36] Nuseibeh, B., Easterbrook, S., 2000. Requirements engineering: a roadmap. ICSE '00: Proceedings of the Conference on The Future of Software Engineering; ACM Press: New York, NY, USA,.35-46.

[37] Reeves, J.W., 1992. What is Software Design?. C++ Journal.

[38] Kaner, C.; Falk, J., Nguyen, H.Q., 1999. Testing Computer Software, 2nd Ed.. New York, John Wiley and Sons, Inc.

[39] Hunt, B., Turner, B., McRitchie, K., 2008. Software Maintenance Implications on Cost and Schedule. Aerospace Conference, IEEE, 1-8 March, 1-6.