Towards A Better Understanding Of Factors Affecting Software Project Complexity

Olga Petkova
petkovao@ccsu.edu
Department of Management Information Systems
Central Connecticut State University
New Britain, CT 06050, USA

Doncho Petkov
petkovd@easternct.edu
Department of Business Administration
Eastern Connecticut State University
Willimantic, CT 06226, USA

Abstract

We propose a model for analysis and understanding of factors affecting software project complexity based on the Analytic Network Process and problem structuring methods. We draw on past research on software project complexity and on research on software development productivity and project contexts. The paper provides an analysis of existing research on software project complexity. The proposed network model of factors affecting software project complexity is illustrated on an example. The use of problem structuring methods for assisting better understanding of software complexity is also discussed.

Keywords: software project complexity, Analytic Network Process, AHP, problem structuring methods.

1. INTRODUCTION

A well-known definition of "project" is that it is "a temporary endeavor undertaken to create a unique product or service" (Project Management Institute, 2000). Software projects are similar to other projects but at the same time they are also quite specific in nature. They are characterized with a significant degree of uncertainty and complexity. Hence understanding software project complexity is an essential precondition for better IT project management. According to Vidal et al (2011:718), today for projects the "usual parameters (time, cost and quality) are clearly not sufficient to describe properly the complete situation". They conclude further that "identifying existing project complexity sources and levels of project complexity has thus become a crucial issue in order to assist modern project management."

Traditionally, complexity in software projects is measured implicitly: either by measuring the software project product (usually based on software cost and effort estimation techniques), or by measuring characteristics of the software process (Fitsilis, 2009). There have been attempts to generate more understanding of software project complexity. Thus a recent paper by Botchkarev and Finnigan (2015) synthesized a large pool of literature to generate a systemic framework of project complexity as a system of systems (Botchkarev & Finnigan, 2015). It

provides a way to analyze software project complexity at three levels – product, internal project environment and external project environment. It provides guidelines to address different combinations of factors affecting project complexity. However the evaluation of the factors affecting complexity is left out of their work. A taxonomy of project complexity proposed in Xia and Lee (2004) was used as a framework for a systematic literature review to study the nature of complexity in IS projects and programs by Gregory & Piccinini (2013). Xia and Lee (2004) produced statistical models of the relationships between the elements of their taxonomy. Such an approach assumes that statistical findings based on one large set of data are applicable to any other situations. We argue that it has little value for the assessment of project complexity in a particular case as for successful project management we do need to understand the context of the specific situation according to Cardinal et al. (2004) since projects are often unique in nature. The potential weakness of approaches producing statistical generalizations can be addressed through the application of the Analytic Hierarchy Process (see Saaty, 1994) for evaluation of software project complexity in specific conditions as proposed by Vidal et al. (2011). They represent the factors affecting project complexity as a hierarchical tree and apply pairwise comparisons (following Saaty, 1994) to evaluate the priorities of the elements in that tree leading at the end to calculation of an aggregate measure of the complexity of a set of projects that are being analyzed. There is no clear evidence however about the details of the way the factors affecting software project complexity were identified in that research. Vidal et al. (2011) state as possible directions for future research the possibility to apply the more powerful modeling capabilities of the Analytic Network Process (ANP) (see Saaty, 1996) and that was one of the motivations for this research.

Understanding the complexity of a project depends to a large degree on the understanding of project contexts. The possibility to show how certain problem structuring methods applicable to context analysis (see Petkov et al, 2013) are suitable for understanding of software project complexity was another motivation for this research.

The purpose of this paper is to propose a better way for analysis and understanding of software project complexity based on ANP and problem structuring methods. We draw on past research on software project complexity and on some of our own research on software development productivity and project contexts. The methodology we apply is within the design science paradigm (see Hevner et al, 2004). The contribution of the paper is in the proposed artefact – a model for better understanding of software project complexity. The next section will deal with existing research on software project complexity and factors affecting software project complexity. Finally we will show how problem structuring methods (see Rosenhead and MIngers, 2001) can assist for better understanding of software project complexity which is then followed by a conclusion.

2. A REVUE OF EXISTING RESEARCH ON FACTORS AFFECTING SOFTWARE PROJECT COMPLEXITY

Baccarini (1996) proposed two types of project complexity: organizational (including types of and number of relationships among hierarchical levels, formal organizational units, and specialization) and technological (types of and number of relationships among inputs, outputs, tasks, and technologies). Williams (1999) has suggested that project complexity is viewed along two dimensions: structural (a project's underlying structure)and uncertainty. Xia and Lee (2004) propose an extension of these ideas in the form of taxonomy of project complexity that has two dimensions: organizational/technological and structural/ dynamic. The definition of these dimensions becomes clearer from the list of factors that affect them according to Xia and Lee (2004):

Structural organizational complexity

- The project manager didn't have direct control over project resources.
- Users provided insufficient support.
- The project had insufficient staffing.
- Project personnel did not have required knowledge/skills.
- Top management offered insufficient support.

Structural IT complexity

- The project involved multiple user units.
- The project team was cross-functional.
- The project involved multiple software environments.
- The system involved real-time data processing.

- The project involved multiple technology platforms.
- The project involved significant integration with other systems.
- The project involved multiple contractors and vendors.

Dynamic organizational complexity

- The project caused changes in business processes.
- Users' information needs changed rapidly.
- Users' business processes changed rapidly.
- The project caused changes in organizational structure.
- Organizational structure changed rapidly.

Dynamic IT complexity

- IT infrastructure changed rapidly.
- IT architecture changed rapidly.
- Software development tools changed rapidly.

The effect of the above factors on complexity was analyzed through statistical models using data based on 541 projects (see Xia & Lee, 2004). There is no evidence provided by these authors how they defined the parameters in each of the four groups of factors. Hence it is hard to evaluate how complete is their list of factors affecting complexity. Another criticism is about the usefulness of their findings for any specific situation. Projects are often unique and hence we agree with Cardinal et al (2004) who stress the need to reveal in every project situation the specific conditions if it is to be managed properly.

After analyzing the weaknesses of existing computational and graph-based project complexity measures, Vidal et al (2011:719) declare that in order to overcome the limits of existing measures, their paper aims "at defining a systems thinking oriented index, which is as far as possible:

- Reliable, meaning the user can be confident with the measure.
- Intuitive and user-friendly, meaning it should be easily computed and implemented, and that users must understand why it assesses project complexity.
- Independent of the project models, so that the measure is an evaluation of project complexity and not an evaluation of the complexity of a given project model.
- Able to highlight project complexity sources when building up the measure, so that the user can analyze more properly project complexity and thus make his decisions."

The above goals are quite ambitious and to the credit of these authors they are mostly achieved. We can note however that their selection of factors affecting project complexity is not comprehensive in spite of the legitimacy of the Delphi approach that was used to determine them.

A holistic systems approach to measuring project complexity is advocated also by Fitsilis (2009) who quotes Simon (1962) that "in complex systems the whole is more than the sum of parts" and "given the properties of the parts and the laws of interaction, it is not trivial to infer the properties of the whole Fitsilis (1999) is suggesting that a comprehensive model of project complexity should be based on the Project Management Body of Knowledge (PMBOK) and its knowledge areas (see Project Management Institute, 2000):

- Project Integration Management;
- Project Scope Management;
- Project Time Management;
- Project Cost Management;
- Project Quality Management;
- Project Human Resource Management;
- Project Communications Management;
- Project Risk Management;
- Project Procurement Management.

For each of the above areas then are suggested the possible sources of complexity and finally are proposed 45 metrics for measuring software project complexity. The PMBOK based model of software project complexity is indeed comprehensive. It is also one of the few attempts to address holistically this issue. However it may be argued that it is of limited practical value due to the large number of metrics and the fact that some of the proposed metrics are not sufficient or well justified. For example for the knowledge area "Project risk management" the author has proposed three metrics: Number of risks identified; Number of risks quantitatively analyzed and Number of risks where a mitigation plan is available. None of these metrics deals with the severity of the risks and with the possibility that intangible risks that cannot be analyzed quantitatively might be most serious sometimes. Hence there is a lot

work needed to make these ideas operational and useful for software development practice.

The next section will discuss the structure of a proposed network model of factors affecting software development productivity as a way to model the complexity of a software project.

3. AN ANALYTIC NETWORK MODEL OF FACTORS AFFECTING THE COMPLEXITY OF SOFTWARE PROJECT

The aim of this section is to describe a model for the prioritisation of the various quantitative and qualitative factors shaping software project complexity. It is based on research on factors affecting software development productivity (Petkova and Petkov, 2003) that can capture the richness of the links between the various software project factors. It is a qualitatively different extension of the original idea by Vidal et al. (2011) through the use of the much more powerful Analytic Network Model, capable of capturing better the complexity of software projects. It is appropriate for modeling software project complexity because similar approaches based on software cost and estimation techniques have been applied for analysis of project complexity before according to Fitsilis (2009). We will present first the essence of the multi-criteria approach that will be used for the model.

An Introduction to the Analytic Network Process

The Analytic Network Process or the Analytic Hierarchy Process for systems with feedback as it is also known aims at modelling the interdependencies between the variables in a decision problem. In our case they will be the factors affecting software project complexity. Saaty distinguishes between outer dependencies, inner - outer and inner - inner dependencies (for detailed definitions see Saaty, 1996).

The relationships between the components of the system (in our case: factors affecting the complexity of a software project) form a network. A component may interact with other components as is shown in Figure 1. In some cases it may influence only some of the elements of other components. Note that the proposed model is not aiming at measuring software project complexity like Vidal et al (2011) but is only suggested as a way of understanding the importance of the various factors affecting project complexity.

The priority of each element in the network with respect to those that are influenced by it is obtained using pairwise comparisons between factors affecting a component that are using a ratio scale from 1 to 9, and finding the eigenvector corresponding to the largest eigenvalue of the comparison matrix, just as in the traditional AHP approach (see Saaty, 1996).

The weighting procedure is performed in two phases due to the two-level structure of the network's vertices. In the first phase each component functions as a criterion for prioritizing the components influencing it. In the second, analogously, elements function as criteria for prioritising other elements when an interconnection exists between them (see Saaty, 1996).

According to Saaty (1996) the vectors of priorities of elements in the individual clusters in the hierarchy form a supermatrix. The supermatrix has to be weighted by the matrix of priority vectors of components with respect to the other components, so that it can become column stochastic.
The final priorities are obtained by calculating the limit of the supermatrix's high powers (Saaty, 1996). That limit is determined when the supermatrix has identical columns. The elements of one such column represent the values of the overall priorities of the subelements of the network components. For further details see Saaty (1996) or software documentation of one of the few software packages for ANP modelling like Decision Lens (Saaty, 1996).

Structure of the ANP Model of Factors Affecting Software Project Complexity and an Illustrative Example

The composition of the model of factors affecting software project complexity using AHP with feedback was derived on the basis of past research on software development productivity (Petkova & Petkov, 2003). The first AHP hierarchy modelling factors affecting software development productivity was suggested in Finnie, Wittig & Petkov (1993). It assumed independence between the factors within a cluster and between factors from different levels in the hierarchy. Further analysis of the influences between factors in a project may be made either using influence diagrams for individual pairs, by drawing networks for groups of pairs, or, in the case of too many variables, most appropriately by using 0,1 matrices where 1 indicates the presence of a relationship. By applying the last approach a lot more relationships between the factors can be

revealed. Those instances where there is no interaction between two factors are indicated with zeroes in the unweighted supermatrix as suggested in Saaty (1996). Each block in the supermatrix corresponds to a group of factors (or components in terms of the theory of AHP for systems with feedback according to Saaty (1996) as they are shown in Figure 1).

This ANP model of factors affecting software project complexity is broader than the taxonomy of complexity factors suggested by Xia & Lee (2004) since it includes a large number of factors affecting software development productivity considered in models such as COCOMO and others described in various published surveys like one by Conte et al. (1985), Kemayel et al. (1991) and others. It is different from the AHP model suggested by Finnie et al. (1993) as it has a network structure and includes more elements that reflect better project complexity. It contains important factors, omitted in Finnie et al (1993), like motivation, work force stability, user management commitment, user experience with the development team, code size and development schedule. On the other hand, some factors that were considered less important in the model by Finnie et al. (1993) were not included in this model. The suggested model is an example how ANP can be used for improving the understanding of project complexity factors. One can notice, however, that the network configuration can be a flexible one, and various factors can be added or dropped depending on the circumstances surrounding a particular project.

Note that some of the elements are qualitative, while others are quantitative in nature. They can be analysed together due to the use of the 1-9 ratio (See Saaty, 1996) applied for the pairwise comparisons between the factors. Thus a more balanced representation of the factors affecting software project complexity may be integrated in the proposed model.

Besides the variables included in traditional models for software cost estimation, an attempt was made to take into account issues such as user characteristics and the effect of Fourth Generation Languages' (4GLs) use in software development . In certain instances some aggregation of factors took place in order to reduce the total number of relationships. For a list of the factors see the first two columns of Table 2.

Thus, in accordance with the psychometric statistical survey by Kemayel et al. (1991), "technical attributes" in Finnie et al (1993) are replaced here with "process attributes". Programming environment is an aggregation of: use of programming tools, adherence to modern programming practices and programming standards, software reuse and power of the equipment (Kemayel et al., 1991).We may note that power of the equipment is an aggregated representation for execution time constraints and computer response time from COCOMO (see Boehm, 1981). Both of these were used as separate factors in Finnie et al. (1993).

Thus, in accordance with the results in Kemayel et al. (1991), "technical attributes" in Finnie et al (1993) are replaced here with "process attributes". Project management factors in our model represent an aggregation of the following factors considered by Kemayel et al (1991): use of a goal structure, adherence to a software life cycle, adherence to an activity distribution and usage of cost estimation procedures. Programming environment is an aggregation of the following features: use of programming tools, adherence to modern programming practices and programming standards, software reuse and power of the equipment (see Kemayel et al., 1991).

Another difference between the model we propose and the one in Finnie et al. (1993) is that no distinction is made here between analyst and programmer - instead those are replaced by the term developer (when we evaluate ability and experience) which reflects the mixed roles of developers in a modern software development environment.

In line with Kemayel et al. (1991) this model considers software management commitment and another factor, motivation as comprising such aspects as recognition, responsibility, nature of work, advancement, salary, status, interpersonal relations, and working conditions. The factor "system architecture", used in Finnie et al. (1993), was aggregated here with processing complexity. The rest of the components of the network model of factors affecting the complexity of a software project were derived from models described in Basili and Musa (1991), Conte et al., (1985), and other sources.

It can be noted that software product factors like required software reliability, processing complexity, size of code, development schedule constraints and requirements volatility are considered by many as uncontrollable. For example, they were not included by Kemayel et al. (1991) in their list of controllable factors

affecting programmer productivity. A closer investigation of the relationships of this group of factors with the rest (see Figure 1) shows that they are oriented in most cases in both directions, and hence the appropriateness of using ANP for their modelling.

A zero element in the unweighted supermatrix indicates that the corresponding factor in that row of the supermatrix does not influence the factor in the respective column. The unweighted supermatrix is formed by the priority vectors of 76 matrices of pairwise comparisons (not shown here for space reasons). These were formed by pairwise comparisons of elements of a component with respect to an element from another component using the 1-9 ratio scale for the pairwise comparisons introduced by Saaty (1994). It should be noted that the primary purpose of this example is to illustrate the proposed methodology and not to generalise findings about the relationships within the model.

The weighting matrix consists of the vectors of priorities of the components of the network (in our case, the groups of factors) which are derived on the basis of their pairwise comparisons with respect to each of the groups of factors. It is shown in Table 1.

As was pointed before, the weighted supermatrix is raised to sufficiently high powers (Saaty, 1996) until the columns in it stop changing and that is an indication for reaching a solution. These results were obtained by using a program for calculating the largest eigenvalue of a matrix and its corresponding eigenvector for network AHP models used in combination with a spreadsheet package for raising the supermatrix to high powers. Another possibility is to use the Decision Lens for ANP. In that case the mathematics of ANP remains hidden from the user. For the same reason we are not discussing here the mathematics of the Analytic Network Process, described in Saaty (1996).

The similar columns of the supermatrix represent the final limiting priorities of each factor with respect to the overall problem – understanding of the factors affecting software project complexity (shown in Table 2).

The analysis of the results and the ranking of the factors in this example show that for the project being analysed requirements volatility, project management factors, development management commitment, user management commitment and motivation are ranked respectively as first, second, fourth, sixth and seventh among all factors. Note that all of these are not technical but managerial factors. The quantitative product factors, like code size and development schedule factors, are ranked only eighth and ninth respectively, while all qualitative product factors are considered as more important according to the results. These conclusions are in line with the findings by Abdel-Hamid and Madnick (1987) that a company's managerial environment has a "significant impact on its software development costs".

Developers' experience and user involvement are ranked quite low. In spite of the illustrative nature of these results, that tends to contradict the claims of many researchers in the MIS area based on surveys that focus usually on a smaller number of factors, and requires further investigation. Some support for the results obtained here can be found in Kemayel et al. (1991), where they mention the paradox of productivity invariance with respect to experience.

It can be concluded that the network model for analysing factors affecting project complexity presented here provides greater information richness, and as such potentially deeper insight into the relationship between factors affecting software project complexity. On the other hand, it involves a lot more effort associated with the generation of the judgements compared to the use of the simpler Analytic Hierarchy Process. For that reason the use of an ANP model compared to an AHP model is justifiable only for very complex, big projects with significant complexity and large budgets in which the difference that it can make matters.

A criticism of AHP/ANP and MCDM is the assumption that a particular type of a model (in this case a network) can be readily identified from the wide range of existing ones and applied to a given problem. However that is a quite simplistic view of a problem that is being analysed. For more complex situations a number of issues need to be considered before identifying what is the problem and deciding on the relevant approach. The publication of the collection of papers in Rosenhead and Mingers (2001) showed the importance of problem structuring techniques (PSM) that help in assisting with such issues.
Rosenhead and Mingers (2001) have produced a collection of papers on the following PSMs originating in United Kingdom: strategic options

development and analysis' (SODA), 'soft systems methodology' (SSM), 'strategic choice', 'robustness analysis' and 'drama theory'. The most prominent among them is Soft Systems Methodology (see Checkland, 1999). Other related methods originating in the USA are Strategic Assumptions Surfacing and Testing (SAST) described in Mason and Mitroff (1981), Interactive Planning (Ackoff, 1993), Critical Systems Heuristics (Ulrich, 1998) and several others which are analyzed in Jackson (2003, 2006).

PSM can be used on their own for example when the goal is to understand the context of a problem or in combination with another problem solving approach like those belonging to Multi Criteria Decision Analysis (MCDA).

According to Belton and Stewart (2010:212) the role for problem structuring for MCDA may be to provide a rich representation of a problematic situation in order to enable effective multi criteria analysis or it may be to problematize a decision which is initially simplistically presented. Further analysis of the expressive capabilities of PSM for nurturing understanding of project contexts is presented in Petkov et al (2013). As an extension of the work of Petkova and Petkov (2003) on factors affecting software development productivity to the analysis of project complexity and previous research on mixing methods in a systemic intervention (see Petkov et al., 2007), we propose here that PSMs can be used for the development of initial understanding of the software project context and its stakeholders and then the factors affecting project complexity can be further explored and prioritized using the Analytic Network Process or for smaller projects, through the Analytic Hierarchy Process.

4. CONCLUSIONS AND POSSIBLE FUTURE RESEARCH

This paper demonstrated how factors affecting software project complexity can be modeled through the Analytic Network Process by Saaty (1996). The MCDM approach provides a richer modeling perspective for understanding the complexity of software projects taking into account the specific project situation. The example demonstrated here is only for illustration of the model and can be adapted for the analysis of other project situations.

Possible future work includes research on applying the proposed model to different project situations, refining for a specific situation the relevant set of factors affecting software project complexity and gathering the reflections of the stakeholders on such interventions.

This paper attempts to show how IT practitioners and researchers can build an ANP model for exploring the complexities affecting a software project and thus getting a better understanding of the factors in a complex project that is needed for improvement of its management. The results from this research aim to contribute to the wider use of MCDM for modeling and analysis in Information Technology research for improvement of software development practice.

5. REFERENCES

Abdel-Hamid, T.K. and Madnick, S.E. (1987). On the Portability of Quantitative Software Estimation Models, Information and Management, 13 (1), 1-10.

Ackoff, R. (1993). Idealized Design: Creative Corporate Visioning, OMEGA, 21, 401-410Baccarini, D. (1996). The concept of project complexity: A review. Int. J. Proj.Mgmt. 14, 4, 201–204.

Basili, V.R. and Musa, J.D. (1991). The Future Engineering of Software: A Management Perspective, IEEE Computer, September, 24(9), 90 - 96.

Belton, V. & Stewart, T. (2010), Problem Structuring and Multiple Criteria Decision Analysis, In M. Ehrgott et al. (eds.), Trends in Multiple Criteria Decision Analysis, International Series in Operations Research and Management Science 142,209-240. Springer

Botchkarev, A. & Finnigan, P. (2015). Complexity in the Context of Information Systems Project Management, Organizational Project Management, 2(1), 15-34.

Cardinal, L.B., Sitkin, S.B. & Long C.P. 2004), Balancing and Rebalancing in the Creation and Evolution of Organizational Control, Organization Science, 15 (4), July–August, 411–431.

Checkland, P. (1999). Systems thinking, systems practice. Includes a 30 –year retrospective, Wiley, Chichester.

Conte, S.D., Dunsmore, H.E. and Shen, V.Y. (1986). Software Engineering Metrics and Models, the Benjamin Cummings Publishing Co. Menlo Park.

Finnie, G.R., Wittig G.E. and Petkov D.I. (1993). Prioritizing Software Development Productivity Factors Using the Analytic Hierarchy Process, Journal of Systems and Software, 22(2), 129-139.

Fitsilis, P. (2009), Measuring the complexity of software projects. CSIE, Computer Science and Information Engineering, World Congress on, Computer Science and Information Engineering, 644-648.

Gregory, R. & Piccicini, E. (2013). The nature of Complexity of IS Projects and Programmes, Proceedings of ECIS, AIS Electronic Library.

Hevner, A. R., S.March, J. Park and S. Ram (2004) "Design Science in Information Systems Research". MIS Quarterly, 28(1), pp. 75-105.

Jackson MC (2003). Systems Thinking: Creative Holism for Managers. Wiley: Chichester.

Jackson, M C. (2006). Beyond Problem Structuring Methods: Reinventing the Future of OR/MS, Journal of the Operational Research Society, Palgrave.

Kemayel, L., Mili, A. and Ouderni, I. (1991). Controllable Factors for Programmer Productivity: A Statistical Study, Journal of Systems and Software, 16(2), 151-163.

Mason, R. and Mitroff, I.(1981). Challenging Strategic Planning Assumptions, Wiley, New York.

Petkov, D., Petkova, O, Andrew, T. & Nepal, T. (2007). Mixing Multiple Criteria Decision Making with soft systems thinking techniques for decision support in complex situations. Decision Support Systems, 43, 1615-1629.

Petkov, D., Petkova, O., & Andrew, T. (2013). On Some Lessons from Modeling Contexts in Complex Problem Solving in Information Technology, Journal in Information Technology Research, 6(4), 55-70.

Petkova O and Petkov D, (2003), Factors Affecting Software Development Productivity in an Outsourced IT Project, Journal of Information Technologies Cases and Applications, (Ivy League Publishing), 5(1), 5-22

Project Management Institute. (2000). A Guide to the Project Management Body of Knowledge (PMBOK). Project Management Institute: Newtown Square, PA, US.

Rosenhead J and Mingers J (eds) (2001). Rational Analysis for a Problematic World Revisited. Wiley: Chichester.

Saaty T. L. (1994) Fundamentals of Multiple Criteria Decision Making with the Analytic Hierarchy Process, RSW Publications, Pittsburgh, 1994.

Saaty T. (1996). The Analytic Network Process, RWS Publications, Pittsburgh

Saaty, T.L. (2008) "Decision Making with the Analytic Hierarchy Process". International Journal of Service Sciences, 1 (1), 83-98.

Simon, H.A. (1962), The architecture of complexity, Proceedings of the American Philosophical Society.

Ulrich, W. (1998). Systems thinking as if people mattered: critical systems thinking for citizens and managers, working paper No.23, Lincoln School of Management.

Vidal, L-A., Marle, F., Bocquet, J-C. (2011) Measuring project complexity using the Analytic Hierarchy Process, International Journal of Project Management, 29: 718–727.

Williams, T. (1999), The need for new paradigms for complex projects. Int. J. Proj. Mgmt. 17, 5, 269–273.

Williams, T. (2009), Decisions Made on Scant Information: Overview, In: Williams, T.M., Samset, K., Sunnevag, K.J. (Eds.), Making Essential Choices with Scant Information; Front End Decision making in projects. Palgrave Macmillan, Basingstoke, UK, 3–17.

Xia, W, & Lee, G. (2004), Grasping the complexity of IS development Projects, Communications of ACM, 47(5), 69-74.

Figures and Tables

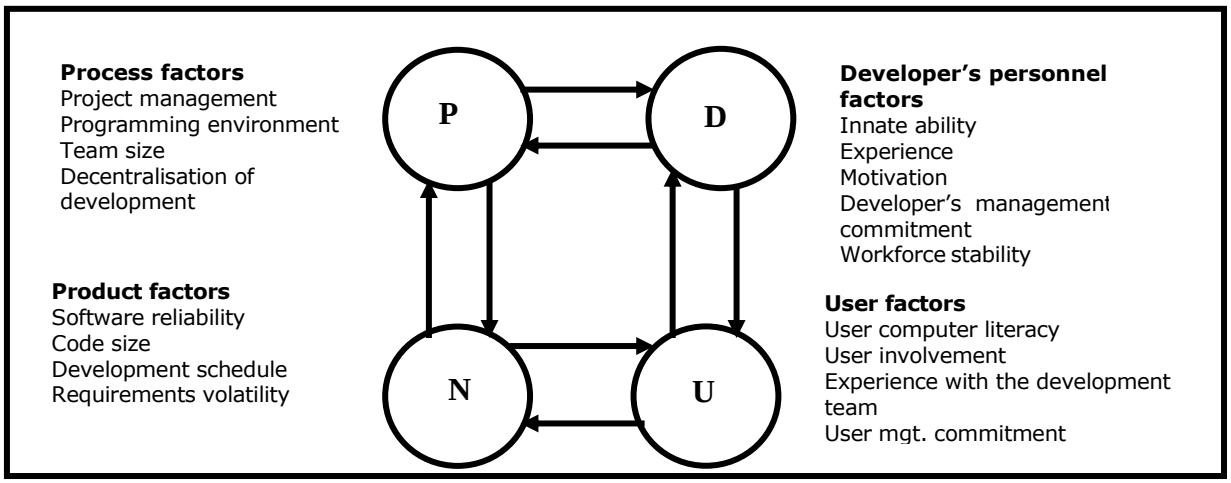

Figure 1. An Analytic Network Model of factors affecting software project complexity

Table 1. The weighting matrix, containing vectors of priorities of the groups of factors with respect to each one of them

		P	D	U	N
Process	P	0.329	0.231	0.144	0.197
Developers	D	0.203	0.424	0.270	0.119
Users	U	0.152	0.077	0.425	0.070
Product	N	0.316	0.268	0.161	0.614

Table 2. Priorities and ranking of the elements in the model of factors affecting software development productivity

Component	Element	Priority	Rank
Process factors	P1 Project management	0.103	2
	P2 Programming environment	0.044	10
	P3 Third party involvement	0.031	14
	P4 Team size	0.041	11
	P5 Decentralization of development	0.029	15
Developers factors	D1 Innate ability	0.028	16-17
	D2 Experience	0.027	18
	D3 Motivation	0.070	7
	D4 Devel. management commitment	0.088	4-5
	D5 Workforce stability	0.039	12
User factors	U1 User computer literacy	0.028	16-17
	U2 User involvement	0.016	19
	U3 Experience with the dev team	0.034	13
	U4 User management commitment	0.074	6
Product factors	N1 software reliability	0.102	3
	N2 complexity	0.088	4
	N3 code size	0.068	8
	N4 development schedule	0.066	9
	N5 requirements volatility	0.116	1