# Flexible and Adaptive Life Cycle Framework for Software Development

Veysi Öztürk*
BSE University, Gulumbe Campus, Department of Information Management Systems, BİLECİK, Turkish.

* Corresponding author. Tel. +90 228 214 1887 email: veysio@yahoo.com
Manuscript submitted December 30, 2015, accepted March 30, 2016
doi: 10.17706/jsw.11.9.943-951

**Abstract:** Even though there has been a considerable amount of progress on software development tools, techniques, methodologies and SDLCs, still %68 of software development projects are unsuccessful.

In this study, Flexible and Adaptive Life Cycle (FALC) Framework is introduced to manage the utilization of different SDLCs in the projects in order to increase success rates. The FALC Framework provides opportunities to combine different SDLCs and/or suitable parts of them at appropriate phases and use them jointly according to the project characteristics and needs. By combining various SDLCs, benefits of each can be reaped from them all.

Moreover, the FALC framework can decrease development time and costs of the projects, provide a paradigm shift on using different SDLCs in parallel by different groups of the project.

**Key words:** Hybrid, SDLC, software development life cycle, FALC framework, software life cycle model.

## 1. Introduction

There are many factors affecting a software project's success. These include the capability and leadership of the project managers, the skill of the software developers, the availability of project and requirement management, the resources available, the computing environment, "the difficulty of the problem being solved", and the Software Development Life Cycles (SDLC) model being employed [1].

Only 32% projects were successful (delivered with required features, on budget, on time, and budgets) according to a survey done in 2009 [2]. %62 of IT projects fail to meet their schedule according to another research [3]. Standish Group identified that a formal methodology was used at 46% of successful projects compared to %30 of challenged and failed projects. According to this research, the use of formal methodology in a project can increase chance of success by about 16% (46%-30%) [4]. Using SDLC(s) is one of formal methodologies. SDLC is a model that defines the sequence of phases and activities that will take place during software development.

Jayaratna reported that over a thousand brand-named systems and software development methodologies (SDLCs) were in use all around the world in 1994 [5]. Some SDLCs may be more suitable for some projects than others. Depending on the selected SDLC model, one can improve quality, improve project tracking and control, decrease development time, minimize overhead, minimize risk exposure, or improve client relations [6].

Some SDLCs are more prevalently used than the others. Some of them are more known and easier to use. SDLCs are being used in projects for the following reasons; (1) understanding for ourselves and explaining

to others the various steps that we must go through before completing a project [7], (2) providing a common language for more effective communication between the project team, project manager, sponsors, and others within the organization [8], (3) providing management with a set of points (milestones), that can be examined to determine the rate of progress of the project [7], (4) focusing on the current tasks, instead of worrying about the next ones [8], (5) highlighting key activities and their interdependencies [9], (6) defining and focusing on roles and responsibilities [10], (7) managing uncertainty [11], (8) encouraging systematic evaluation of options and opportunities [11], (9) reducing development and maintenance costs due to the reasons given above [12].

Some of frequently used SDLCs and methodologies are given in [6], [13]-[15].

Some problems using SDLCs are as follows;

1) Some selected SDLCs are not suitable to the project. But project manager and team generally understand this after the project begins (sometimes in early stages, something in the middle, sometimes more late)
2) All SDLCs have advantages and disadvantages. It is necessary to select the most suitable one, which can be very difficult in some circumstances.
3) Most standards and experts advice to select SDLC before the project begins. Because of this and other reasons, project plans and agreements are done taken into account the selected SDLC. Because of this, it is generally difficult to change the selected SDLC.
4) Project needs more than one SDLC (for example, some groups of the project may want to use different SDLC )

SDLCs have paramount affects on the success of projects. “One size clearly does not fit all, particularly when the types of projects vary widely” [16]. If we take a home improvement project as an example, we may choose an equipment to begin with (e.g. a hammer) and complete the work with another one (e.g. pincers). Software development projects are no exception! Why do not we do the same for completing all phases of a project? We can begin a project with a certain life cycle and then switch to another one according to the status and requirements of the project in order to cope with dynamic nature of projects.

In many cases, it may be necessary to integrate parts and procedures from various SDLCs to support software development. The reason of this is that most SDLCs were developed for achieving success only under some circumstances. When the circumstances change beyond the limits or focus of the life cycle, there can be some problems of using it [17]. For example, the output of Software Development Life Cycle (SDLC) can be no longer predictable. In these situations, it can be necessary to alter the existing SDLC to accommodate the changes, or adapt or combine different models to comply with the new circumstances [17]. Besides, project management needs to deal with minimizing risks, which is usually managed by trying to estimate the events that will deviate from the project schedule from plans beforehand. Although a project may introduce new and unexpected events that have not been foreseen at the outset, those events may put the project at jeopardy.

In this study, a framework called Flexible and Adaptive Life Cycle (FALC) was developed. FALC framework gives opportunities to combine suitable parts of SDLCs at the right phase and time according to the project properties, circumstances and needs.

FALC framework can facilitate adaptation of project management to the dynamic nature of projects and associated risks.

## 2. Hybrid Software Development Life Cycles

Practical experience in the projects (including industry, R&D and public) indicates that “the selection of a single software development process does not provide the desired flexibility and support to business needs”

[18]. Combinations of SDLCs or hybrids are used generally in the organizations and companies [19]. Hybrid combination of traditional software engineering and agile methods can be a route to increase the quality of the developed tools and software [20]. A hybrid life cycle can be established by using a combination of different life cycles based on the changing needs of a project.

Hybrid life cycle was used in two different meanings in the literature:

A) As a new Software Development Life Cycle including and combining known life cycles totally or partly (hybrid methodology is generally used in this meaning in the literature, see ( [17], [21]-[23]). Some combined models are as follows [24] :
   - Spiral software development model= linear model + iterative model
   - Components assembly model= spiral model + OO model + 4GT
   - IBM clean room SDLC= transformational model + incremental model + Dynamic model +4GT

   Dawson suggests a meta-model (hybrid combination of some life cycles) to allow developers to adopt different model solutions as a project progresses, addressing specific problems as they arise [23].

B) Combination of known Software Development Life Cycles: This is not a new SDLC but a new approach. Rothman suggests using and combining different life cycles in projects. He thinks that life cycle combination may provide the most value to the project [25]. Farrell suggested combination of SDLCs as a very useful technique. He suggested to take parts of compatible methodologies (SDLCs) and use them together (jointly) on a single project [26]. Rothman and Farrell did not propose any technique, process and framework how to combine SDLCs and use them. In this study, we propose a framework and a process to build and use hybrid life cycles in software development projects (to take parts of compatible SDLCs and use them in parallel by different groups of the project jointly).

## 3. Flexible and Adaptive Life Cycle (FALC) Framework

There is analogical resemblance between development life cycles and home improvement apparatus such as hammer and pincers. For accomplishing some work at home, we may begin with an apparatus (e.g. hammer) and complete with another one (e.g. pincers). We can do the same for completing all phases of a project. We may begin a project with a certain life cycle and then switch to another one according to the status and needs of the project. In other words, if SDLC is a vehicle, why insist on using the same vehicle until the end of the journey? We can use different SDLCs at different phase(s) of the software development journey.

Let's remember roadmap analogy. Some maps can emphasize terrain, some of the others roadways, some of the others public transportation routes and some of them can be a combination. You can select the one that is most suited for your journey or you might want to use more than one [27]. If you are a group, then some of you may use different roadmaps from each other. Similarly, different groups of a project may use different SDLCs.

In many software projects especially long-lived ones, we may need to switch to another SDLC. The reasons of change manifest itself as follows;

a) The project deadline can be proponed by the management or the customer due to changing market conditions.
b) The change in the experience of the team (For example, if the project is planned to be executed with inexperienced personnel, the life cycle model would need to be rechecked as the personnel will be building on their experience as the project progresses).
c) The change in the status (importance, probability, etc.) of project risks
d) The risks can be mitigated according to the life cycle change.
e) Selection of an inappropriate wrong life cycle at the beginning or at any stage of the project.

f) Project personnel turnover.
g) In general, non-technical people prepare project proposals and also there are times where work packages are defined according to the development life cycle. Shortly after commencing the project, the project team is built. That team may decide that another life cycle might (would) be more appropriate to the project. For example, they may argue, "we cannot finish this project in time using the X software life cycle".

In this study, a framework called FALC (Flexible and Adaptive Life Cycle) which allows altering or changing SDLCs in a given project was developed. FALC Framework is a combination and parallelization of software and system development life cycles. FALC framework can enhance the ability of project team to adapt to the changes without losing the control of the project. The FALC Framework enables you to

a) use suitable parts of different software or system development life cycles (for each phase of the project, some phase(s) of life cycle can be more suitable. For example, if you have a big project and your team does not have enough experience, then you may select waterfall for analysis phase and decide another life cycle for the next phase)
b) change the entire life cycle or some steps of it
c) use different life cycles in parallel at the same time according to the needs of project groups in a project.
d) adapt life cycle(s) to the project instead of adapting project to the life cycle.
e) adapt life cycle(s) to your company instead of adapting your company to the life cycle.

The aim of SDLCs is to complete the project(s) successfully. Almost all SDLC models are composed of phases. A phase causes to reduce the risks and provide more manageability.

Division the project into phases enables us to control and direct the activities in a disciplined, orderly and methodical way that is responsive to changes, unexpected situations, adaptations and complications [11]. Therefore, main emphasis of FALC framework is the phases.

"No matter which life cycle model is followed, the basic activities are included in all life cycle models (SDLCs) though the activities may be carried out in different orders in different life cycle models" [28].

Project team, project manager or project management office reviews the status of the project at some decision points in FALC Framework. Decision points are milestones of the projects (the end of a phase). At decision points, it can be decided to re-plan SDLC of each project group and re-organize the project groups.

One can decide to use the FALC framework at the beginning of the project or at any phase of the project. A general form of FALC Framework is shown in Figure 1. Life Cycle X1, Life Cycle X2, … Life Cycle Xn, Life Cycle Y1, Life Cycle Y2, … Life Cycle Yn means different or the same known life cycles such as waterfall, incremental, spiral, Scrum, XP, etc. Project Group 1, Project Group m means different Project groups.

Phase 1, Phase 2, Phase n means phases (stages/steps) of the Project. Decision Point 1, Decision Point n are the decision points after each phase. For example, as shown in Fig. 1 you can plan to use Life Cycle X1 at the beginning of the project for Project Group 1, then you can change (Life Cycle X1 ≠ Life Cycle X2) or keep the same life cycle (Life Cycle X1 = Life Cycle X2) in the next phase.

In some organizations or projects, some work packages (especially big work packages) can be done by different groups. In this case, different life cycles can be used in work packages of a project. FALC framework allows using of different life cycles in work packages.

In a project, different groups may need to follow different life cycles. FALC framework allows groups to follow different life cycles. A project group is a team responsible for achieving a/some part(s) of the project or developed system.

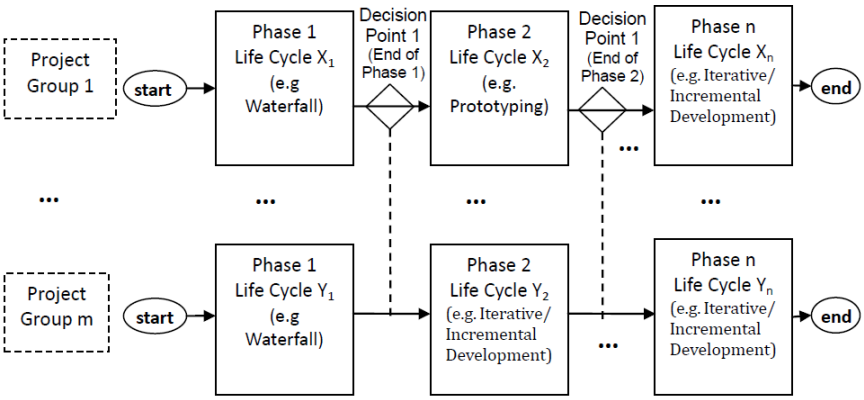


Fig. 1. FALC framework generalization.

The process for defining FALC framework phases is shown in Fig. 2. The steps of the process are as follows;

1) Decide SDLCs that will be used in the project. The following criteria can be taken into account when selecting the SDLCs;
   - Project (development) environment such as organization characteristics, the familiarity with the planned/decided technology to be used ( [29], [30]),
   - Developers and user's knowledge and experience including users' and developers' experience in application domain, users' ability to express and define software requirements and developers' software development and engineering experience [1],
   - development time, project's size, project team's skills, the IT and organizational infrastructure to support the project [8].
   - Clarity of requirements [31],
   - Number of (critical) risks,
   - Cost estimation,
   - Early functionality needs,
2) Decide the number of groups according to the SDLCs: In this step, all groups that use a different order of SDLCs will be defined. If two groups use the same order of SDLCs according to the check points, we can join them as one group. If you want to check these two groups at decision points separately or the groups are members of different companies, you can keep them. According to the project needs, a project may include only one group. A group may include only one person or more. If you decide to use only one SDLC and sure not to change it during the progress of the project, you will not need to use FALC framework. If you cannot decide from the beginning which SDLCs to use by each group, you can use the same SDLC for all groups.
3) Define phases and their details (beginning and end time of a phase, SDLCs in phases, duration time, deliverables, roles of project members, etc.) of each group. The phase for FALC is the period where only one SDLC will be used for a project group.
4) Define decision points for each phase. Decision points can be selected according to the selected SDLCs and milestones of the projects. For example, the end of a stage for a sequential SDLC or the end of an iteration for a cyclic SDLC can be decision points.
   - Check and evaluate the plan after the definition phase and at the end of each phase, and revise if needed. Note that the conditions may change at the end of each phase according to the beginning of the projects. After each phase, the life cycle of next phase can be decided using the following inputs;

the lateness in the plan, skill set of the project personnel, the change in the development time, the change in the status of the project risks and needs of the project

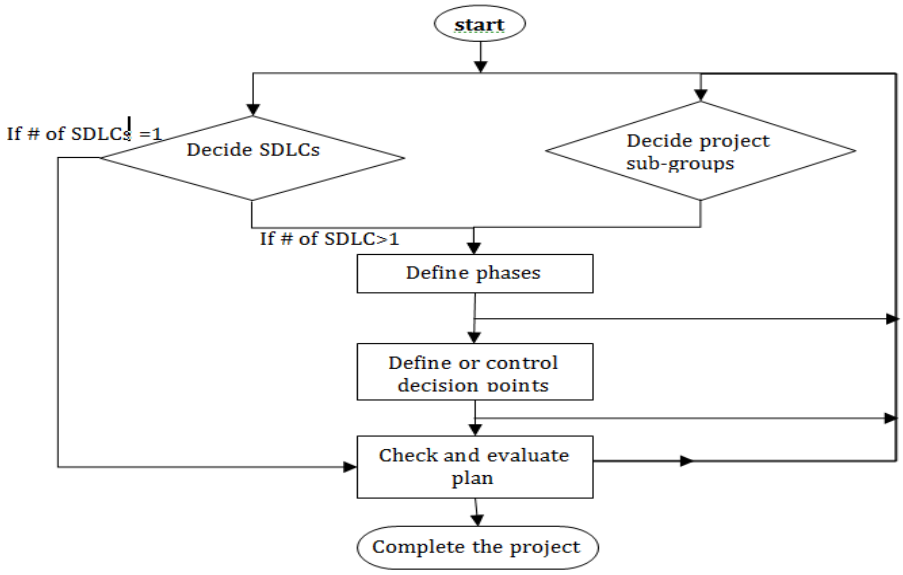


Fig. 2. The process of FALC framework.

The roles of engineers in a project may change after some phases. For example, if we use waterfall and then decide to change SDLC as agile, the role of test engineers will change. Test engineers will test the software and instead of generate test conditions and cases.

The phases must be defined according to the selected SDLC in each phase. Some SDLCs are sequential and output of the SDLC is one version of software (e.g. waterfall). Some SDLCs are cyclic (repeated cycles) and output of the SDLC can be more than one version (e.g. spiral, agile). An example of using FALC framework in a project is given in Fig. 2(here m=2). If we begin a project with a sequential SDLC, in that case we must select what phases of that SDLC will be done in the project. For example, requirements specification can be done in Phase 1 of the example given in Fig. 2. Project Group 1 can develop one of sub-systems, especially critical one(s) using prototyping SDLC in Phase 2. Project Group 2 (if m=2) can use Iterative and Incremental development until the end of the project. Project Group 1 can use Iterative and Incremental development after completing a sub-system. If we begin a project with a cyclic SDLC and then change it to another cyclic SDLC, in that case different parts of the system will be done in different cyclic SDLCs.

We must pay attention to the following during the use of FALC framework;

- The life cycles' order should be arranged correctly.
- The groups should know SDLCs which will be used in their group. It will be better if they know all the SDLCs which will be used in the project/company.

FALC framework may not be fruitful in the following situations;

- In the case where the project manager and team do not know or have very little knowledge about SDLCs (it may be very difficult to select the right life cycle for the right phase without enough knowledge and experiences about SDLCs). In such case, the project members can receive consultancy.
- Small projects with few persons (e.g. development of some web sites may not require detailed plans)
- In the projects that should be finished in a very short time (has tight schedule to follow).

FALC framework will be useful in the following situations;

- project team is large
- project members have different software or system development approach
- project manager or team have enough knowledge about SDLCs (it will help to select the right life

cycle for the right phase) or project can receive consultancy from experts.

- project seems to be failed (changing and controlling SDLCs can be an option)

## 4. Application of FALC Framework in Live Projects

FALC Framework was used partially in the development of an Electronic Record Management System (ERMS). FALC framework provided the following benefits in the project;

- Different SDLCs were used according to project groups' needs
- Suitable parts of different SDLCs were used.
- The risks at development of some parts were decreased.
- Coding time was decreased according to the initial plan.

ERMS includes the following main parts; Electronic Record Management (ERM) Software, ERM Database, Optical Character Recognition (OCR) Module, Record Sharing on Intranet & Internet Module and Scanning and Indexing Software

The application of the FALC Framework in ERMS project is shown in Fig. 3. There are four groups according to the use of SDLCs. Each group uses different SDLC sequence as shown in Fig. 3.

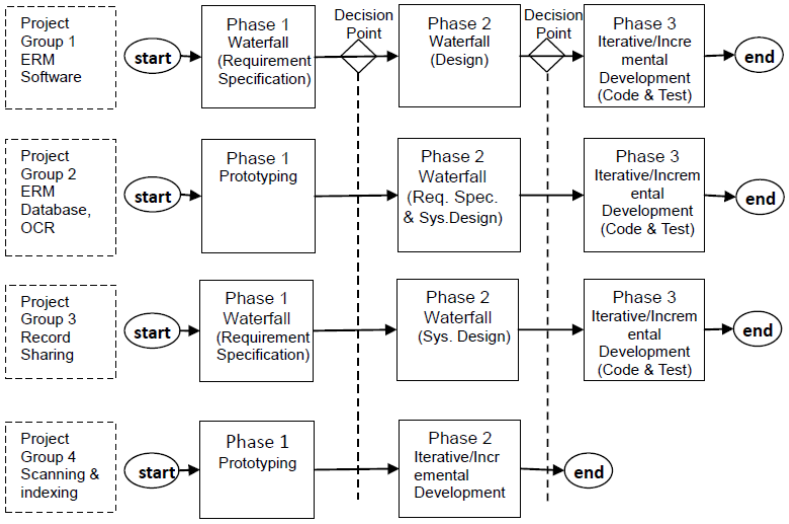


Fig. 3. Application of FALC framework in an ERMS development.

## 5. Conclusion

In this study, a framework was developed that gives opportunity to combine suitable parts of SDLCs at the right phase and time according to the project characteristics and needs. By combining several different life cycles, benefits of each can be attained. The framework was kept as simple as possible to allow easy understanding and using.

The advantages of using FALC framework are as follows;

a) Providing paradigm change that you can change the selected software or system development life cycle easily (The project or company management is encouraged about life cycle change).
b) The development time and cost can decrease by using appropriate SDLCs at appropriate phases and time according to the project properties, characteristics and needs
c) The risks can be decreased by switching to another SDLC according to the risk status.
d) Adapting SDLCs to the projects instead of adapting projects to the SDLCs.

The study does not claim that FALC framework will be useful in all situations. It can be useful if you need to use more than one SDLC. It can be useful if your project team is large and different groups of the team

have different software or system development approach. It can be helpful, especially when the project seems to be failed or things go wrong. Someone may think if the FALC framework allows an organization or a project manager too much freedom, thereby negating advantages of an imposed life cycle. In fact, organizations and project managers are generally free about what SDLC(s) to use or select. Customers or top management generally concerns about the output instead of process. FALC Framework gives opportunity to control and manage this freedom.

## Acknowledgments

The author wishes to thank Koray İnçki, Burak S. Soyer, Abdulkerim Öncü, and Ramazan Cengiz for their valuable support.

## References

[1] Alexander, L. C., & Davis, A. M. (1991). Criteria for selecting software process models. *Proceedings of the Fifteenth Annual International Conference*.

[2] The standish group international. (2009). The 10 Laws of CHAOS, Retrieved March 10, 2016, from the Department of Computer Science at the University of Chicago Web Site: https://www.classes.cs.uchicago.edu/archive/2014/fall/51210-1/required.reading/Standish.Group.Chaos.2009.pdf

[3] Dynamic Markets Limited. (2007). *IT Projects: Experience Certainty, Independent Market Research Report*. Abergavenny UK

[4] The Standish Group International. (2001). Extreme chaos. Retrieved December 24, 2015, from Centro de Informática Web Site: http://www.cin.ufpe.br/~gmp/docs/papers/extreme_chaos2001.pdf

[5] Strode, D. E. (2005). The agile methods: An analytical comparison of five agile methods and an investigation of their target environment. MSc. Thesis Massey University, New Zeland

[6] McConnell, S. (1996). *Rapid Development: Taming Wild Software Schedules*. Washington, Microsoft Press

[7] Liu, L. C., & Horowitz, E. (1989). A formal model for software project management. *IEEE Transactions On Software Engıneering.*

[8] Marchewka, J. T. (2002). *Information Technology Project Management*. Wiley

[9] Abran, A., *et al*. (2004). *Guide to the Software Engineering Body of Knowledge SWEBOK*.

[10] Government of Ontario. (2007). GO-ITS 54 Application Development Standards for SDLC. Retrieved December 24, 2015, from the Ontario Government Web Site: https://www.ontario.ca/document/go-its-54-application-development-standards-sdlc

[11] Benediktsson, O., Dalcher, D., & Thorbergsson, H. (2006). Comparison of software development life cycles: A multiproject experiment.

[12] Davis, A. M., Bersoff, E. H. B., & Comer, E. (1988). A strategy for comparing alternative software development life cycle models. *IEEE Transactions on Software Engineering.*

[13] Westfall, L. (2010). *The Certified Software Quality Engineer Handbook*. Milwaukee, American Society for Quality, Quality Press

[14] Boehm, B. W. (1988). A spiral model of software development and enhancement.

[15] Cockburn, A. (2001). *Agile Software Development*, Addison-Wesley Professional

[16] Ambler, S. W. (2013). Choose the right software method for the job. Retrieved December 24, 2015, from Agile Data Web Site: http://www.agiledata.org/essays/differentStrategies.html

[17] Center for technology in government. (1998). A survey of system development. Retrieved December 24, 2015, from The Center for Technology in Government (CTG) Web Site: http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf

[18] Al-Maharmeh, M. (2009). Applying a composite process framework (CPF) in real life software development project. *Proceedings of the IEEE Sixth International Conference on Information Technology.*

[19] Chenal, D. (2009). Matching software development lifecycle (SDLC) management tools to your environment. Retrieved December 24, 2015, from SDLCtools Web Site: http://www.sdlctools.com/index_files/Matching_SDLC_Management_Tools_to_Your_Environment_09.pdf

[20] Parsons, D., & Lal, R. (2006). Hybrid agile development and software quality. *Proceedings of the 14th International Conference on Software Quality Management (SQM).*

[21] Conigliaro, J. P. (2001). Hybrid methodologies for small scale engineering. *Proceedings of the Change Management and the New Industrial Revolution.*

[22] Shaikh, M. A. M., Morshed, A. S. M., & Mitsuru, I. (2004). Object oriented hybrid software engineering process (SEP) model for small scale software development firms, *In Proc. of 2nd International Conference on Computer Science and its Applications (ICCSA 2004)*. San Diego

[23] Dawson, C. W., & Dawson R. J. (1995). Towards more flexible management of software development using meta-models, *Software Engineering Journal.*

[24] Osama E., & Fadi P. D. (2016). Tailoring the software process model to project requirements. Retrieved March 10, 2016, from New Jersey Institute of Technology Web Site: http://eljabiri1.tripod.com/sitebuildercontent/sitebuilderfiles/paper5.pdf.

[25] Rothman, J. (2008). What lifecycle? selecting the right model for your project. Retrieved December 24, 2015, from Rothman Consulting Group Web Site: http://www.jrothman.com/articles/2008/01/what-lifecycle-selecting-the-right-model-for-your-project/

[26] Farrell, A. (2007) *Selecting a Software Development Methodology Based on Organizational Characteristics.* Master of Science Thesis, Athabasca University.

[27] Sorting out SDLC terminology. Retrieved December 24, 2015, from Testdog Web Site: http://testdog.com/knowhow/Sorting%20Out%20SDLC.html

[28] Software engineering issues. Retrieved December 24, 2015 from NPTEL National Programme on Technology Enchanced Learning Web Site: http://nptel.iitm.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Embedded%20systems/Pdf/Lesson-33.pdf

[29] Desaulniers, D. H., & Anderson, R. J. (2001). Matching software development life cycles to the project environment. *Proceedings of the Project Management Institute Annual Seminars and Symposium*.

[30] Archibald, R. D. (2003). Life cycle models for high-technology projects – Applying systems thinking to managing projects. Retrieved July 28, 2010 from Russell Archibald Web Site, http://www.russarchibald.com/files/LIFECYCLEMODELSFORHI-TECHPROJECTS.pdf

[31] Grażyna H. J., & Golińska, (2010). Decision support system for choosing a model for a software development life cycle. Retrieved December 24, 2015 from journal Operations Research and Decisions Web Site: http://www.ioz.pwr.wroc.pl/boid/artykuly/1-2010/art-4-holodnik.pdf

**Veysi Öztürk** works at BSE University as an assistant professor doctor. He received his BSc from electric and electronics engineering of Hacettepe University in 1996. He received his MSc from Science Institute of Gazi University in 1999. He received his PhD from Department of Computer Engineering at İstanbul Technical University in 2006. His research interests include are software engineering and development, artificial intelligence, expert systems, and distributed simulation. He worked at various international/national research projects related with artificial intelligence, expert systems, distributed simulation, sensor development issues as project manager or chief senior researcher. He is the author of more than 15 international/national publications.