DE GRUYTER OPEN

Information Technology and Management Science doi: 10.1515/itms-2015-0001

2015 / 18

Selection of Software Development Project Lifecycle Model in Government Institution

Oksana Medvedska[1], Solvita Berzisa[2]
[1, 2] *Riga Technical University*

***Abstract* – Software development projects in government institutions have certain characteristics that can negatively impact the management process of these projects. However, the negative impact of these characteristics can be decreased by using the appropriate software development project lifecycle model, because it ensures more comprehensive and effective project management. The methodology for selecting the most appropriate model for software development projects in government institutions, where outsourcing is used, is elaborated. The impact of the characteristics of software development projects in government institutions on project execution and management process is analysed. The methodology is elaborated taking into account these characteristics and ensures that the appropriate model is selected.**

***Keywords* – Government project management, project lifecycle, public sector management, selection of software development project lifecycle model.**

I. Introduction

Wirick [1] highlights that project managers in the public sector encounter difficulties because of the constraints under which management is required to operate. These constraints are caused by wide differences that public and private organisations have in a variety of aspects, to which Boyne [2] refers as organisational environment, managerial values, etc. Software development projects in government institutions have certain characteristics that can negatively impact and complicate the management process of these projects. However, the negative impact of these characteristics can be decreased by using the appropriate software development project lifecycle model, because it ensures more comprehensive and effective project planning and management during all phases of the software development project lifecycle [3]. It is important to note that the project lifecycle model and the software development lifecycle model interact during all phases of the software development project lifecycle and both impact the project, so in this research they should be discussed conjointly as the software development project lifecycle (hereafter – SDPL) model.

Different SDPL models can be used – waterfall, iterative and incremental such as evolutionary or spiral model, agile such as SCRUM or DSDM methodologies and other SDPL models, although none is the universally appropriate model for each project. Each model has certain conditions of use, advantages and disadvantages depending on the project characteristics and circumstances; thereby, each model's effectiveness varies with project characteristics and other factors affecting the project, such as enterprise environmental factors. Because of this, the most appropriate (effective) model must be selected for every project [4].

Review of scientific literature in this field as well as practical experience shows that very often the SDPL model for the software development projects in government institutions is selected arbitrarily without any analysis of project characteristics and other factors affecting the project, so an inappropriate model can be selected [5], [6]. Use of ineffective SDPL model can adversely affect project manageability and, as a consequence, can cause a decrease in software quality [4], [5]. Therefore, model selection is a strategically important decision, especially in government institutions, where software development project management is more complex.

The research objective is to elaborate the methodology for selecting the most appropriate model for the government software development projects, where outsourcing is used. Impact of characteristics of software development projects in government institutions on project execution and management process is analysed in the present research. The methodology is elaborated taking into account these characteristics and other factors affecting the software development projects in government institutions and ensures that the appropriate SDPL model is selected.

The paper is organised in 5 sections. Section II describes the background of the research, including description of SDPL models, SDPL model selection standards as well as characteristics of the government software development projects. Section III describes the steps of the methodology for selecting the most appropriate SDPL model for the government software development projects. The results of applying this methodology to a particular software development project in the State Revenue Service of Latvia are presented in Section IV. Conclusions are drawn and recommendations to decrease the negative impact some factors have on execution and management process of the government software development projects are proposed in Section V.

II. Background

The PMBOK classification of the project lifecycle is described in *Subsection A*. SDPL models typically used in government institutions are reviewed in *Subsection B*. Major drawbacks of using standard SDPL model selection guidance are outlined in *Subsection C*. In order to elaborate SDPL model selection methodology directly applicable to the government software development projects, the impact of characteristics of these projects on project execution and management process is analysed in *Subsection D*.

A. Software Development Project Lifecycle

The project lifecycle consists of four stages (1) initiation; (2) planning; (3) execution and (4) closure [7], [9]. It provides the framework for executing and managing the project; however, large and complex projects may require more exhaustive management and control. In such instances, the work carried out to complete the project objectives within the context of the generic project lifecycle structure may be broken down into any number of phases to ease planning, management and control [8].

The PMBOK [8] classifies the project lifecycle as predictive, iterative and incremental or adaptive. In the projects with predictive lifecycle the objectives, scope, resources required for the project are determined as soon as possible, phases are sequential and plan is a more general, high-level document. In the projects with iterative and incremental lifecycle, project activities within the phase are executed iteratively to develop the deliverables (software, documentation) increment by increment until the criteria for the phase closure are met, high-level planning is executed at the beginning of the lifecycle, and more detailed planning is executed for each iteration. Adaptive lifecycle is also iterative and incremental, but iterations are very rapid. At the end of each iteration, the customer provides the feedback about the deliverable developed within the iteration to acknowledge that their current needs are reflected.

B. SDPL Models

The project lifecycle phase within which the software is developed is very substantial in the software development project and depends on the SDPL model. Although the project lifecycle model is used as a tool for planning and managing the project, the SDPL model is used for developing the software as a product of the project.

According to the project lifecycle classification proposed by the PMBOK [8] and taking into account the features of SDPL models, the following classification of SDPL models is proposed:

- Predictive lifecycle → the waterfall model;
- Iterative and incremental lifecycle → the evolutionary model, the spiral model;
- Adaptive lifecycle → Scrum and DSDM.

The classical **waterfall model** divides the SDPL into such phases: (1) feasibility study; (2) requirements analysis; (3) design; (4) development; (5) testing; (6) deployment and maintenance (number and names of phases may vary). This model assumes that all requirements are defined at the beginning of the project, all the phases are sequential, the various activities during the phase are assumed to be flawlessly done, and therefore, there is no need to re-enter the phase [3].

In the **evolutionary model**, a subset of the software requirements, which are well-understood, is initially implemented as a useable version of the software. This version of the software is then delivered to the customer to provide feedback and then is amended in compliance with this feedback. The evolutionary model is very similar to the **evolutionary prototyping model**, and the terms are sometimes used synonymously. In the evolutionary prototyping model, however, the initial software increments implement less well-understood requirements, rather than well-understood requirements [4].

The diagrammatic representation of **the spiral model** appears like a spiral with many loops. Over each loop, some features of the software are identified and analysed and the risks at that point of time are identified and resolved through prototyping. Based on this, the identified features are implemented in the next version of the software [3].

Scrum is an Agile framework and, as such, is consistent with the values of the Agile Manifesto [10]. The software is developed incrementally in a series of time periods called sprints [11]. Each sprint begins with sprint planning, produces a software increment and ends with sprint review and sprint retrospective when the developed software increment and the development process are reviewed [12].

DSDM (Dynamic Systems Development Method) consists of three phases: pre-project, project lifecycle and post-project phase. The project lifecycle phase consists of five sub-phases [13]. The Feasibility and Foundation phases are sequential, but the software then is developed iteratively and incrementally within the Exploration phase, when functional prototype is developed, Engineering phase, when design prototype is developed, and Deployment phase, when software is delivered to the customer [14].

It can be concluded that each model determines the SDPL in different ways; therefore, activities within the phases can differ, each model has several advantages and disadvantages etc. Because of this, the most appropriate (effective) model must be selected for every project [4]. However, Alexander et al. [5] state that models are often selected on an ad hoc basis using a set of unjustified undocumented criteria.

C. SDPL Model Selection Guidance

Several standards and methodologies contain guidance on the selection of the most appropriate SDPL model.

McConnell specifies a set of questions about the project and criteria to assess the answers. McConnell uses the undefined qualitative values to evaluate SDPL models against criteria [15].

Alexander et al. define a set of criteria and criteria values. These criteria are actually a set of project characteristics. To select the most appropriate SDPL model for a particular project, each criterion is evaluated according to the project characteristics, and SDPL model appropriateness is then evaluated against values of certain criteria and quantified by summarising the model evaluation of each criterion [5]. All criteria have equal importance; moreover, criteria are assessed independently of each other. Therefore, the methodology proposed by Alexander et al. does not help determine the most appropriate SDPL model in a particular set of project circumstances [4].

Davis et al. use the following metrics for comparing SDPL models: shortfall, lateness, adaptability, longevity and inappropriateness. Davis et al. do not describe how models

are evaluated against these metrics and evaluate the effectiveness of various models without considering the project circumstances [4].

ISO/IEC FDIS 1107:2007, CMMI-DEV, DO-118B, TickIT Guide and other standards can also be used to select the most appropriate SDPL model.

D. Software Development Projects in Government Institutions

In order to develop a set of criteria for selecting the most appropriate model for the government software development projects, it is necessary to identify the characteristics impacting these projects. Such characteristics are expert judgement-based (in project management at government institutions experts are experienced project managers) and are supported by the available literature in this field, so far as is possible.

The following **environmental factors** impact the execution and management processes of government software development projects.

1) A hierarchical structure of government institutions and functional matrix project organisational structure [2], [16].

▪In such organisational structure, the dispersal of decision-making power leads to the fact that nobody assumes full management responsibility for the project [9].

Impact on project execution and management process: *high*.

▪When more than one functional division is involved in project execution, the potential for conflict between functional managers and project managers exists because of resource conflict. This can cause delays in the execution of the project activities.

Impact: *high*.

▪Project managers must obtain continual cooperation from functional managers of many other divisions. Certain difficulties in the inter-divisional cooperation and information exchange can cause difficulties in managing the interdependent projects.

Impact: *medium*.

2) A role culture and autocratic management style hinder personnel from planning, organising and controlling their work on the project on their own; project execution depends entirely on the leader's competence, operational experience and personality [1], [2].

Impact: *medium*.

3) Multiplicity of the government software systems.

▪Personnel need to be aware of multiple technologies; furthermore, an average high-level IT specialist salary at government institutions is lower than in the private sector. Therefore, shortage and high fluctuation of qualified personnel is observed in government institutions [1], [17].

Impact: *medium*.

▪Considering the shortage and high fluctuation of qualified personnel, outsourcing is often used that results in a high degree of dependency on an outsourced developer. Moreover, a procurement procedure is necessary to use outsourcing software development [1].

Impact: *medium*.

4) Internal policies and procedures do not provide the framework for executing and managing the software development projects [8], [16].

▪The content of the project planning documents, milestones, approaches to risk management and quality management etc. can differ that can make it difficult to manage interdependent software development projects.

Impact: *low*.

▪Historical information and lessons learned throughout the executing and managing software development projects are unavailable because knowledge base is not maintained.

Impact: *low*.

The following **characteristics of the government software development projects** impact the execution and management process of these projects.

1) High degree of interdependence of software systems.

▪This leads to interdependency between software development projects which complicates the planning, management and control processes of these projects.

Impact: *medium*.

▪This must be considered at the time of defining software system requirements, designing functional and technical architecture, testing, and implementing the software system.

Impact: *medium*.

▪In order to provide usable functionality for carrying out core business processes, it is often necessary to implement all functionality simultaneously; however, it is difficult to break down the software system into functionally independent increments.

Impact: *medium*.

2) Business processes of government institutions supported by software systems are determined by legislation [16].

▪Legislation determines deadlines to implement software system functionality; therefore, government software development projects are often time-bounded projects constrained by hard deadlines.

Impact: *high*.

▪Because of hard deadlines, it is often necessary to specify software system requirements during developing legislation. Furthermore, several software systems may require adjustments and amendments; however, functional divisions, which operate with these software systems, may not be involved in the project. Taking into account the above-mentioned considerations, software system requirements may not be initially fully understood and specified and may also be modified during the project.

Impact: *high*.

3) Outsourced developer is responsible only for software system requirements analysis, design and development, whereas a government institution as a customer – for software system acceptance testing, implementation and maintenance. Therefore, high-quality project documentation, including software system documentation, is required in order to ensure high-quality software system testing, configuration management, change management etc.

Impact: *medium.*

4) Government software systems process restricted information (e.g. personally identifiable information). Therefore, the software system must be developed considering all security requirements in order to ensure security, confidentiality and integration of information.
Impact: *medium.*

Taking into account the characteristics of the government software development projects described above, it can be concluded that software development projects in the public sector differ from software development projects in the private sector. Although software development projects in the private sector can be extremely different, the following factors distinguish the environment of software development projects in the private sector from the environment in the public sector:

- projectized or projectized matrix organizational structure in the private sector instead of functional or functional matrix structure in the public sector;
- task culture instead of role culture;
- democratic management style instead of autocratic;
- lower level of shortage of qualified personnel.

Software development projects in the private sector are more flexible – limits of resources, amount of planning, software system development and implementing approach, amount of software system documentation etc. are more flexible. The framework for executing and managing software development projects (including framework for selecting the SDPL model) is provided by internal policies and procedures in many mature software development companies.

It follows from this that the SDPL model selection methodology directly applicable to the government software development projects needs to be elaborated.

III. SDPL Model Selection Methodology

The methodology for selecting the most appropriate SDPL model for the government software development projects is elaborated.

SDPL model selection criteria are defined taking into account characteristics of the government software development projects mentioned in *Subsection D* of Section II. Criteria are specified by the values and the relative weights. Values of criteria represent the range of values characterising a particular project. The relative weights of criteria depend on the impact each criterion has on the execution and management process of government software development projects, thus determining for which criterion the selection of the appropriate SDPL model is most important. The rating scales of SDPL models determine the appropriateness of a particular SDPL model with respect to the values of criteria.

The relative weights of criteria are defined on the basis of the authors' personal experience in government software development project management. Each rating scale is defined on the basis of the analysis of the advantages, disadvantages and conditions of use of the waterfall, iterative and incremental, evolutionary, spiral and agile models. Therefore, the relative weights of criteria, the rating scales of SDPL models and the ranges of values of criteria can be defined more precisely in accordance with the characteristics of software development projects in a particular government institution and the experience accumulated managing software development projects in this government institution.

Criteria (c_i, i = 1..20) and ranges of criterion values (v_{ij}, j = a..e), as well as the relative weight (w_i) of each criterion are defined in Table II. The rating scales (r_i) of the waterfall, iterative and incremental, evolutionary, spiral and agile models with respect to each criterion value are defined in Table III.

The methodology allows selecting the appropriate SDPL model for the government software development project in the following steps:

0. Precise the relative weights of criteria (w_i), the rating scales of SDPL models (r_i) and the ranges of values of criteria (v_{ij}).

- w_i values are the following: 1 – no impact, 1.25 – insignificant, 1.50 – significant, 1.75 – very significant impact.
- r_i values are the following: 0 – not appropriate, 1 – partly appropriate, 2 – almost appropriate, 3 – appropriate, 4 – very appropriate.
- ranges of values of criteria v_{ij} differ depending on criteria.

1. Evaluate the project against the set of twenty criteria c_i. Criteria c_i may take on the values v_{ij} defined in Table II.

2. Evaluate the appropriateness of SDPL model x with respect to the criterion's c_i value v_{ij} by determining the rating $r_i(x)$ according to the rating scale defined in Table III.

3. Multiply $r_i(x)$ by w_i and summarise the total rating of the SDPL model x.

$$r(x) = \sum_{i=1}^{20} (r_i(x) \times w_i) \quad (1)$$

4. Select the SDPL model with the highest total rating $MAX(r(x))$ as the appropriate SDPL model.

TABLE II

Criteria for Evaluating Software Development Project

c_i	Criterion	w_i	Values of criteria				
			v_{ia}	v_{ib}	v_{ic}	v_{id}	v_{ie}
c_1	Complexity	1.50	1 software system is involved	2 software systems are involved	3–4 software systems are involved	4–6 software systems are involved	> 6 software systems are involved
c_2	Size (man-hours)	1.25	<200	200–500	500–1 000	1 000–1 500	>1 500
c_3	Ability to develop software system incrementally	1.75	Cannot be broken down into increments	Increments are interdependent, cannot be brought into production apart	Increments are interdependent, but can be brought into production apart	Increments are independent, cannot be brought into production apart	Increments are independent, cannot be brought into production apart

c_i	Criterion	w_i	Values of criteria				
			v_{ia}	v_{ib}	v_{ic}	v_{id}	v_{ie}
c_4	Quality of the initially determined requirements	1.75	Only basic requirements need to be defined more precisely	Only business requirements need to be defined more precisely	Business and user requirements need to be defined more precisely	All level requirements need to be defined more precisely	All level requirements do not need to be defined more precisely
c_5	Probability of changing requirements	1.75	Very likely, extreme magnitude	Very likely, moderate magnitude	Likely, extreme magnitude	Likely, moderate magnitude	Unlikely, minor magnitude
c_6	Software security requirements	1.50	None	Very low	Low	High	Very high
c_7	Requirement for amount and granularity of documentation	1.50	Minimum, low level of detail	Small, low level of detail	Medium, medium level of detail	Large, high level of detail	Very large, very high level of detail
c_8	Time constraint and slack time	1.50	Bounded with no slack time	–	Bounded with slack time	–	No constraints
c_9	Funds availability	1.50	Bounded with no reserve	–	Bounded with reserve	–	No constraints
c_{10}	Personnel availability	1.50	Limited with no reserve	–	Limited with reserve	–	No constraints
c_{11}	Personnel qualification and experience	1.25	Low with no experience	Average with no experience	Average with experience > 2 years	High with experience > 2 years	High with experience > 5 years
c_{12}	Outsourced developer qualification and experience	1.50	Low with no experience	Average with no experience	Average with experience > 2 years	High with experience > 2 years	High with experience > 5 years
c_{13}	Communication between client and developer	1.25	Formal, informal on request	–	Formal, informal, at the site on request	–	Formal, informal, at the site regularly
c_{14}	Project organisational structure	1.25	Functional	Weak matrix	Balanced matrix	Strong matrix	Projectised
c_{15}	Amount of planning at the beginning of the project	1.25	Minimum	Small	Medium	Large	Very large
c_{16}	Risk management approach	1.50	None	–	With no documenting	–	With documenting
c_{17}	Acceptable level of developer's risk	1.25	Minimum	Small	Medium	Large	Very large
c_{18}	Verification and validation of results of the project	1.50	At the end of each phase	–	At the end of each iteration	–	Throughout the project
c_{19}	Need for project progress visibility for client	1.50	Very little	Little	Medium	Great	Acute
c_{20}	Acceptable amount of training to use the SDPL model	1.00	No constraints	Large	Medium	Small	Inadmissible

TABLE III

THE RATING OF SDPL MODELS WITH RESPECT TO CRITERIA VALUES

$r_i(x)$	x	Waterfall					Iterative and incremental					Evolutionary					Spiral					SCRUM					DSDM				
c_i	v_{ij}	v_{ia}	v_{ib}	v_{ic}	v_{id}	v_{ie}	v_{ia}	v_{ib}	v_{ic}	v_{id}	v_{ie}	v_{ia}	v_{ib}	v_{ic}	v_{id}	v_{ie}	v_{ia}	v_{ib}	v_{ic}	v_{id}	v_{ie}	v_{ia}	v_{ib}	v_{ic}	v_{id}	v_{ie}	v_{ia}	v_{ib}	v_{ic}	v_{id}	v_{ie}
c_1		4	2	1	0	0	4	4	3	2	1	4	3	2	1	1	4	3	1	0	0	4	4	3	1	0	4	4	3	2	1
c_2		4	3	1	0	0	4	3	2	2	1	4	3	3	2	1	4	3	2	1	0	4	3	2	0	0	4	3	3	2	1
c_3		4	3	2	3	1	0	1	2	2	4	2	2	3	3	4	0	1	3	3	4	1	2	3	4	4	1	3	3	4	4
c_4		0	1	2	2	4	0	1	2	3	4	2	2	3	3	4	2	2	3	3	4	2	3	3	4	4	2	3	3	4	4
c_5		0	0	1	2	3	0	1	1	2	3	1	1	2	3	4	1	2	3	3	4	2	2	3	3	4	2	2	3	3	4
c_6		4	4	3	3	2	4	3	2	2	1	4	3	2	1	1	4	3	3	2	1	4	3	3	2	2	4	3	3	2	2
c_7		0	0	3	4	4	1	1	3	3	2	1	1	3	3	2	2	2	3	3	2	4	4	3	1	0	2	3	4	3	3
c_8		0	–	2	–	4	2	–	3	–	4	1	–	3	–	4	2	–	3	–	4	3	–	4	–	3	2	–	4	–	3
c_9		0	–	2	–	4	1	–	3	–	4	3	–	4	–	4	3	–	3	–	4	3	–	3	–	4	3	–	3	–	4
c_{10}		0	–	1	–	4	2	–	3	–	4	2	–	3	–	4	1	–	2	–	4	2	–	3	–	4	2	–	3	–	4
c_{11}		3	3	4	4	4	3	4	4	4	4	3	4	4	4	4	2	2	3	4	4	0	1	3	3	4	1	2	3	3	4
c_{12}		1	2	3	3	4	2	2	3	4	4	2	2	3	4	4	2	2	3	3	4	0	1	2	3	4	1	1	2	3	4
c_{13}		3	–	4	–	4	2	–	4	–	4	2	–	3	–	4	2	–	3	–	4	0	–	2	–	4	1	–	2	–	4
c_{14}		2	2	3	3	4	2	2	3	3	4	2	2	3	3	4	1	2	2	3	4	0	1	2	3	4	1	2	2	3	4
c_{15}		0	1	1	3	4	1	1	2	3	3	1	2	2	3	3	1	2	2	3	3	2	2	3	3	3	2	2	3	3	3
c_{16}		3	–	2	–	2	1	–	2	–	3	1	–	2	–	3	0	–	2	–	4	1	–	3	–	3	1	–	3	–	4
c_{17}		4	3	2	1	0	2	2	3	3	3	2	2	3	3	3	2	2	3	3	3	1	2	2	3	4	1	2	3	3	4
c_{18}		4	–	2	–	1	0	–	3	–	4	1	–	3	–	3	1	–	3	–	4	0	–	3	–	4	1	–	3	–	4
c_{19}		4	4	3	2	1	2	3	4	4	3	3	3	4	4	3	3	3	4	4	3	1	2	4	4	4	1	2	4	4	4
c_{20}		4	4	4	4	3	4	4	4	3	3	4	4	4	3	2	4	4	3	2	1	4	3	3	1	1	4	3	3	2	1

IV. Application Example and Discussion

The methodology described in Section III was applied to select the appropriate SDPL model for a particular software development project in the State Revenue Service of Latvia (hereafter – SRS), which has almost all characteristics of the government software development projects described in *Subsection D* of Section II.

Commonly the waterfall model is used for software development projects in the SRS. However, commonly it is selected without any analysis of characteristics of software development projects; therefore, it can be inappropriate. The main results of SWOT (*Strengths – Weaknesses – Opportunities – Threats*) analysis of how appropriate the waterfall model is as the SDPL model for a particular software development project in the SRS are presented in Table IV.

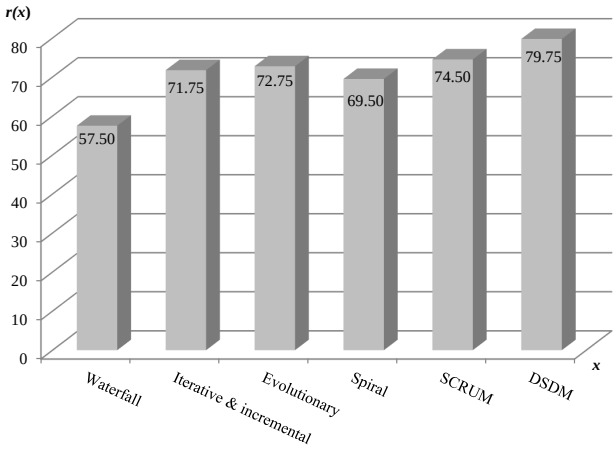

Fig. 1. The rating of the evaluated SDPL models.

TABLE IV
SWOT Analysis of Using Waterfall Model

Strengths	Weaknesses
•Document-driven approach provides comprehensive documentation for software system further development, testing and maintenance. •Detailed planning at the beginning of the project provides more coordinated inter-divisional cooperation (several SRS functional divisions are involved). •Model determines SDPL in an easy-to-understand way, so high-level experience is not necessary.	•Model requires all requirements to be defined before designing software system architecture (however, requirements are initially not well understood). •Performing phases sequentially is time-consuming (however, project is constrained by hard deadlines). •Model does not support building software system incrementally. •Late and rare delivery does not ensure visibility of progress. •Model does not cover all aspects of risk management (however, such a high-risk project requires comprehensive risk management).
Opportunities	**Threats**
•Verification and validation of specifications (e.g. design specification) allow detecting non-conformity of specifications on time. •Developing all the functionality simultaneously can ensure a higher level of the software system security. •By dividing SDPL into sequential phases, problems can also be solved sequentially one by one (important for such a complex project).	•Working software system is delivered only in the testing and deployment phase, so non-fulfilment of the requirements can occur. •Model requires to re-enter all previous phases in case of any changes (e.g. requirements changes) increasing the risk of exceeding time and budget as well as the risk of conflict with the related project (if artifacts from the previous phase have already been agreed and have already been used in the related project).

The results of applying the methodology for a particular SRS software development project are shown in Fig. 1 and according to them the DSDM model is most appropriate, whereas the waterfall model is most inappropriate.

The main results of SWOT analysis of how appropriate the DSDM model is as SDPL model for a particular software development project in the SRS are presented in Table V.

TABLE V
SWOT Analysis of Using DSDM Model

Strengths	Weaknesses
•Planning and resourcing each phase allow keeping a project on track (however, a project is constrained by hard deadlines). •Model requires only high-level requirements to be defined before designing software system architecture (however, requirements are initially not well understood). •Iterative and incremental development and delivery ensure clear visibility of progress. •Iterative risk management process helps manage project risks proactively.	•Model is complicated and difficult to understand. •Model is a relatively new approach represented with a limited amount of examples of good practice.
Opportunities	**Threats**
•By defining requirements iteratively, the software system addresses the current and imminent needs (requirements can change during developing legislation). •MoSCoW prioritisation allows avoiding development of less needed functionality of software system and keeping the project on track. •By testing early and continuous non-conformity of the software system can be detected and eliminated as soon as possible.	•Personnel have no experience in using the DSDM model. •Model requires collaboration and cooperation; however, difficulties may arise in the inter-divisional communication. •Model requires personnel to be empowered to make decisions; however, in the functional matrix structure (like in the SRS) only functional managers have full decision-making power. •Model requires active user involvement in the development process; however, it may be difficult to involve all users (users of software system are both internal and external).

As the DSDM model can provide the usable and useful 80 % of the wanted software system functionality in 20 % of the total development time [13], it can be concluded that the DSDM model is more effective for such a project as a particular software development project in the SRS, i.e. with unstable requirements and hard deadlines than the waterfall model. This conclusion is supported by SWOT analysis results and the main results are the following:

▪Software system requirements are initially not well understood as the waterfall model requires. DSDM model requires only high-level requirements to be defined initially; other requirements are defined iteratively as well as are prioritised.
Benefit: Focus on the business needs.
▪Changes in the legislation can cause changes in the requirements. Furthermore, hard deadlines for implementing the software system are determined by legislation.
Benefit: Though the DSDM model determines SDPL as an iterative and incremental approach and planning, monitoring and controlling processes also are iterative, the likelihood of the project being completed in time and on budget is increased.
▪Software system functionality supports interdependent business processes, therefore, is very complex.
Benefit: Non-conformity of the software system can be detected and eliminated as soon as possible by testing early and continuously as the DSDM model assumes.

V. Conclusion

The results of applying the methodology discussed in Section IV confirm that the methodology can be applied to select the appropriate SDPL model for the government software development projects, because the DSDM model selected by applying the methodology to a particular SRS project as the SDPL model can help decrease the negative impact some characteristics have on the execution and management processes of this project. However, the negative impact some factors have on the execution and management processes of the government software development projects cannot be decreased only by selecting and using the appropriate SDPL model. Therefore, the following recommendations are proposed:
▪Developing internal policies and procedures how to manage and execute software development projects in a particular government institution, including guidelines for selecting the appropriate SDPL model;
▪Developing a common knowledge base containing the historical information about the previous projects, e.g. the effect of decision results, risk assessment measures, using a particular SDPL model etc.;
▪Assessing the possibility of establishing the Project Management Office responsible for taking the project-related decisions in accordance with the strategy of a particular government institution in order to avoid conflict between functional managers and project managers;
▪Assessing the amount of financial resources accessible for increasing personnel motivation in order to attract and retain qualified personnel and decrease the degree of dependency on the outsourced developer.

The methodology for selecting the most appropriate SDPL model for the government software development projects can be improved by analysing the mutual dependence and impact of criterion values that can be an objective of the further research. The potential for applying the methodology for selecting the most appropriate SDPL model for software development projects in the private sector can also be further studied.

References

[1] D.W. Wirick, *Public-sector Project Management: Meeting the Challenges and Achieving Results*. New Jersey: Wiley, 2009. http://dx.doi.org/10.1002/9780470549131
[2] G.A. Boyne, "Public and Private Management: What's the Difference?" *Journal of Management Studies*, vol. 39, Issue 1, 2002, pp. 97–122. http://dx.doi.org/10.1111/1467-6486.00284
[3] R. Mall, *Fundamentals of Software Engineering*. New Delhi: PHI Learning Pvt., 2009.
[4] R.W. Boyd, *Software Lifecycle Model Selection: Criteria for Safety-critical Software*. York: University of York, 2009.
[5] L. Alexander, A. Davis, "Criteria for Selecting Software Process Models," *Proc. of the 15th Int. IEEE COMPSAC*, 1991, pp. 521–528. http://dx.doi.org/10.1109/cmpsac.1991.170231
[6] V. Massey, "Comparing Various SDLC Models And The New Proposed Model On The Basis Of Available Methodology," *Int. J. of Advanced Research in Computer Science and Software Engineering*, vol. 2, Issue 4, 2012, pp. 170–177.
[7] J. Westland, *The Project Management Life Cycle: A Complete Step-by-step Methodology for Initiating, Planning, Executing & Closing a Project Successfully*. London: Kogan Page, 2007.
[8] Project Management Institute, *A Guide to the Project Management Body of Knowledge: PMBOK Guide Fifth Edition*. Newton Square: Project Management Institute, 2013.
[9] K. Schwalbe, *Information Technology Project Management*. Boston: Course Tehcnology, 2013.
[10] The Agile Manifesto. Agile Alliance, [Online]. Available: http://www.agilealliance.org/the-alliance/the-agile-manifesto/. [Accessed: June 18, 2015].
[11] The Scrum Guide. ScrumGuides.Org, [Online]. Available: http://www.scrumguides.org/scrum-guide.html [Accessed: June 18, 2015].
[12] K. Schwaber, *Agile Project Management with Scrum*. Washington: Microsoft Press, 2004.
[13] What is DSDM? DSDM CONSORTiUM, [Online]. Available: http://www.dsdm.org/content/what-dsdm. [Accessed: June 18, 2015].
[14] J. Stapleton, *DSDM, Dynamic Systems Development Method: The Method in Practice*. Cambridge: Cambridge University Press, 1997.
[15] S. McConnell, *Rapid Development*. Redmond: Microsoft Press, 1996.
[16] Project Management Institute, *Government Extension to the PMBOK Guide Third Edition*. Newton Square: Project Management Institute, 2013.
[17] E. Pūlmanis, "Public Sector Project Management Application and Sustainability Problems, Case of EU Member State – Latvia," *PM World Journal*, vol. 3, Issue 9, 2014.

Oksana Medvedska obtained her *BSc* (2012) and *MSc* (2015) degrees in Computer Science and Information Technology from Riga Technical University (Latvia) and *BSc* degree (2014) in Economics from Daugavpils University (Latvia). Her major field of study is IT project management. From 2012 till 2013, she was an Assistant Project Manager at website design and development company, Riga, Latvia. Since 2013 she has been an IT Project Manager at the State Revenue Service, Riga, Latvia. Current research interests are IT project management in the public sector, international standards in the field of IT project management, service-oriented architecture (SOA) in the public sector.
E-mail: Oksana.Medvedska@vid.gov.lv

Solvita Berzisa holds a Doctoral Degree and is a Lecturer and Researcher at the Institute of Information Technology of Riga Technical University (Latvia). She obtained her *Dr. sc. ing.* (2012), *BSc* (2005) and *MSc* (2007) degrees in Computer Science and Information Technology from Riga Technical University. Her main research fields are IT project management, project management information systems implementation and application, as well as project data analytics. She also works as an IT Project Manager at Exigen Services Latvia. She holds a PMP certificate and was awarded the IPMA Outstanding Research Contribution by a Young Researcher 2013. She is a Member of PMI, IIBA and Latvian National Project Management Association.
E-mail: Solvita.Berzisa@rtu.lv