Formalizing the ISDF Software Development Methodology

Mihai Liviu DESPA
Bucharest University of Economic Studies
mihai.despa@yahoo.com

The paper is aimed at depicting the ISDF software development methodology by emphasizing quality management and software development lifecycle. The ISDF methodology was built especially for innovative software development projects. The ISDF methodology was developed empirically by trial and error in the process of implementing multiple innovative projects. The research process began by analysing key concepts like innovation and software development and by settling the important dilemma of what makes a web application innovative. Innovation in software development is presented from the end-user, project owner and project manager's point of view. The main components of a software development methodology are identified. Thus a software development methodology should account for people, roles, skills, teams, tools, techniques, processes, activities, standards, quality measuring tools, and team values. Current software development models are presented and briefly analysed. The need for a dedicated innovation oriented software development methodology is emphasized by highlighting shortcomings of current software development methodologies when tackling innovation. The ISDF methodology is presented in the context of developing an actual application. The ALHPA application is used as a case study for emphasizing the characteristics of the ISDF methodology. The development life cycle of the ISDF methodology includes research, planning, prototyping, design, development, testing, setup and maintenance. Artefacts generated by the ISDF methodology are presented. Quality is managed in the ISDF methodology by assessing compliance, usability, reliability, repeatability, availability and security. In order to properly asses each quality component a dedicated indicator is built. A template for interpreting each indicator is provided. Conclusions are formulated and new related research topics are submitted for debate.

Keywords: *Software Development Methodology, Innovation, Project Management*

1 Introduction

The research efforts and results presented in the current paper apply exclusively to web applications. Though they might apply to other categories of software applications or to other fields altogether, they were validated only in the context of web applications. From the end-user's point of view, a web application is considered to be innovative if it's easier to use, faster, cheaper, more reliable or more secure than other applications that accomplish the same results or if it fulfils a need that has yet to be address in the online environment. In the context of the end-user, innovation targets the fulfilment of a specific need.

From the project owner's point of view a web application is considered innovative if it:

- includes at least a functionality that generates added value for the end-user and the functionality is not found in other web applications that target the same market;
- includes a combination of functionalities that generate added value and the combination of functionalities is not found in the same configuration in any other web application that targets the same market; functionalities can be found separately in other web applications but not in the same configuration;
- provides access to a graphic interface that includes elements or element combinations which improve user experience and are not found in other web applications that target the same market.

In the context of the project owner innovation focusses on market characteristics and targets novelty and added value. From the project manager and from the development

DOI: 10.12948/issn14531305/19.2.2015.07

team's point of view a web application is considered to be innovative if it includes functionality that they have never implemented before. In the context of the project manager and the project team, innovation focusses on the degree of novelty of the current application compared to previously implemented applications.

This paper focuses on the perspective of the project manager and the project team regarding innovative web applications. Research and the author's own experience in the field of software development lead to the conclusion that innovative web applications are characterized by frequent change of specifications, high dynamics of technology and standards, higher than usual risks, proprietary cost structure and custom testing scenarios. Thus the research hypothesis of the current paper is the fact that building an innovative web application requires a dedicated software development methodology.

A software development methodology is an effort to standardize the set of methods, procedures and artefacts intrinsic to the software development life cycle [1]. The software development methodology illustrated in the current paper is called Innovative Software Development Framework and will be referred with the acronym ISDF. The methodology was developed based on practices employed by the author in innovative IT projects he personally managed in the last 5 years. The initial methodology was built empirically based on the development life cycle and was refined and formalized by integrating additional elements identified by reviewing scientific papers. The resulting methodology was tested and validated in the successful implementation of three innovative software development projects. The ISDF methodology is depicted in the current paper by presenting a case study performed on one of the above mentioned projects. In order to comply with confidentiality contract clauses and to protect the project owner's identity data is anonymized and project will be referred to with the acronym ALPHA. The results and scientific output presented in the current paper have been presented at the 14th International Conference on Informatics in Economy, Education, Research and Business Technologies.

2 Literature Review

Current software development methodologies are branched into heavyweight and lightweight. As part of the literature review process, heavyweight and lightweight methodologies were analyzed with an emphasis on epitomizing their overall structure, positive attributes, negative attributes and the type of project they are suitable for.

Heavyweight methodologies follow the waterfall model and rely on detailed planning, exhaustive specifications and detailed application design. The waterfall model is predictable, generates comprehensive software artefacts and diminishes the risk of overlooking major architectural problems [3]. Waterfall model is typically described as a unidirectional, top down [6] as every phase begins only after the previous phase has been completed [7]. The output of one phase becomes input for the next phase [7]. The central figure of the waterfall model is the project plan [11]. Waterfall development entails high effort and costs for writing and approving documents, difficulties in responding to change, unexpected quality problems and schedule overrun due to testing being performed late in the project and lack of project owner feedback [3]. Other issues proprietary to the waterfall model is the fact that systems often do not reflect current requirements and lead-time is often generated by the need to approve software artefacts. Also the waterfall model pushes high-risk and difficult, elements to end of the project, aggravates complexity overload, encourages late integration and produces unreliable up-front schedules and estimates [4]. Waterfall works best for projects with little change, little novelty, and low complexity [4].

Lightweight methodologies follow the agile model and emphasize working software, responding to change effectively and user feedback. Agile model was built to be adaptive, flexible and responsive with an emphasis on collaboration and communication. The

DOI: 10.12948/issn14531305/19.2.2015.07

agile model embraces conflict while encouraging exploration and creativity [5]. Agile model relies on iterative and incremental development [9] and focuses on people not on technology or techniques [8]. The central figure of the agile model is the project owner [11]. The downside of agile model is the fact that it relies on inadequate architectural planning, over-focusing on early results, generates weak documentation and low levels of test coverage [2]. There is a powerful negative correlation between the size of the organization and the successful implementation of the Agile model, thus the larger the organization the harder it is to employ agile methods [10]. Also the Agile model offers limited support for globally distributed development teams, reduces the ability to outsource and narrows the perspective of generating reusable artefacts [12]. Agile model works best for small teams as in large teams the number of communication lines that have to be maintained can reduce the effectiveness of practices such as informal face-to-face communications and review meetings [12].

The need for formalizing a software development methodology dedicated to innovative projects is generated by the fact that traditional heavyweight methodologies are unable of delivering fast development without compromising quality whereas agile lightweight methodologies are characterized by inadequate documentation, weak architecture and lack of risk management [2]. A software development methodology has to be described quantitatively and qualitatively, has to lead to similar results if used repeatedly, has to be applied with a reasonable level of success and has to be relatively easy to explain and teach [13]. A software development methodology should include people, roles, skills, teams, tools, techniques, processes, activities, standards, quality measuring tools, and team values [12].

3 Developing the ALPHA Application

The core of every software methodology is its development life cycle. The development life cycle formalized in the ISDF methodology and used in the ALPHA project consists of the following stages: research, planning, design, prototype, development, testing, setup and maintenance. Research, planning, development, testing and setup are common stages in most software development methodologies. Building a prototype, design and maintenance are also employed in other software development methodologies but are not regarded as distinct development life cycle stages. Innovative software development projects though, enforce prototyping as a distinct stage because it plays an important role in reducing risk, refining specifications and validating the innovative idea that initially lead to the inception of the project. As part of the research process development, life cycle stages of the ALPHA project were analyzed as independent entities highlighting, people and roles.

Research stage in the ALPHA project methodology was dedicated to gathering and exchanging information and it involved the project manager, the project owner and the project team. The project owner's role was to formulate requirements and communicate them to the project manager. The project manager's role was to evaluate requirements and assemble a team with the necessary set of skills, professional values and experience required to implement the project. Including the project manager, 8 people were involved in developing the ALPHA application. Previous experiences lead to the conclusion that the ISDF methodology is effective on teams that do not exceed 9 team members. When selecting the team members, the project manager took into account the fact that implementing innovative projects requires strong associating, questioning, observing, experimenting, and networking skills [14]. The project team's role was to evaluate requirements from a technical perspective. In the ALPHA project, the project manager together with the project team also had the role of converting requirements into actual specifications. As part of the research process, the project owner analysed applications that were similar or complementary to the ALPHA application.

Planning stage in the ALPHA project was

DOI: 10.12948/issn14531305/19.2.2015.07

dedicated to formalizing the main characteristics of the web application and it involved the project owner, project manager and the project team. The project owner had the role of providing feedback on software artefacts. The project manager's role was to plan activities, set standards and assign responsibilities to team members. The project manager together with the team members had the role of defining the overall flow of the application. The flow was broken down into smaller, easier to manage subassemblies. For each subassembly a comprehensive set of functionalities was defined. Based on the required functionality the technical team members designed the database structure. The project manager together with the project team also chose the tools, technologies and processes that were going to be employed in the ALPHA project.

Design stage in the ALPHA project was dedicated to creating the graphic component of the application and it involved the project owner, the project manager and the project team. The role of the project owner was to provide feedback on the layout. The project manager had the role of ensuring that the graphic component is consistent with the functionality and the target group of the web application. The only team member involved in the design stage was the graphic designer. His role was to create a layout in accordance with specifications received from the project manager.

Prototype stage in the ALPHA project was dedicated to building a functional proof of concept and it involved the project owner, the project manager and the project team. The role of the project owner was to provide feedback on the prototype. The role of the project manager was to refine specifications in accordance with the project owner's feedback. The role of the project team was to build the prototype. Innovative web development projects are characterized by a considerable degree of uncertainty. Building the prototype had the role of validating the idea that lead to the inception of the ALPHA project. The prototype also acted as a basis for delivering consistent feedback and refining specifications.

Development stage in the ALPHA project was dedicated to actually building the functionality part of the application and it involved the project manager and the project team. The role of the project manager was to monitor progress, motivate team members and report to the project owner. The role of the development team was to write code and debug.

Testing stage in the ALPHA project was dedicated to identifying programming, design, and architectural issues and it involved the project manager and the project team. The role of the project manager was to insure that the testing scenarios were exhaustive. The role of the project team was to identify and fix security, functionality, design and architectural issues and fix them. Also the project team had to ensure that the web application is doing everything it was design to do and nothing that it wasn't design to do.

Setup stage in the ALPHA project was dedicated to installing the web application on the live environment and it involved the project team. The role of the project team was to configuring the live environment in terms of security, hardware and software resources.

Maintenance stage in the ALPHA project was dedicated to ensuring that the application is running properly on the live environment and it involved the project team. The role of the project team was to monitor the traffic, and the firewall, mail, database and network protocols error logs.

Next step in the research process was to analyses the succession, connections and interaction of the software development life cycle stages highlighting resources, activities and tools. Figure 1 presents a schematic representation of the development life cycle used in the ALPHA project. The development life cycle presented in Figure 1 is also representative for the ISDF methodology.

DOI: 10.12948/issn14531305/19.2.2015.07

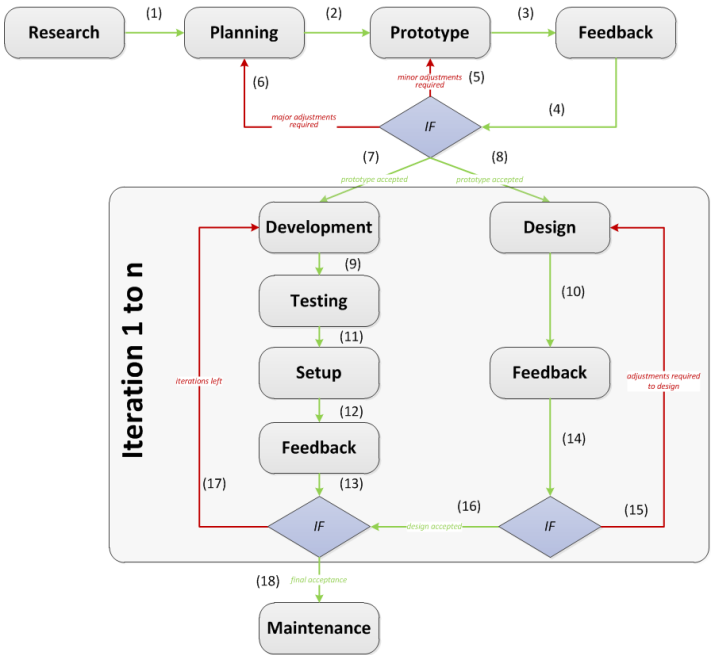

Fig. 1. Development life cycle for the ALPHA application

Research for the ALPHA project started with a series of meetings between the project manager and the project owner. The project owner presented his vision on the application and detailed on the initial set of requirements. The project manager then analysed similar web application already operating in the online environment. The project team performed a technical review of the requirements. The Research stage ended with the project manager and the project team drafting the specifications for the ALPHA application. In the Planning stage the project manager and the project team defined the overall flow of the ALPHA application and broke it down into manageable subassemblies. The overall flow and the subassemblies were built with the help of use case diagrams, UCD. Figure 2 presents the UCD diagram for the Register – Login- Logout process of the ALPHA application.

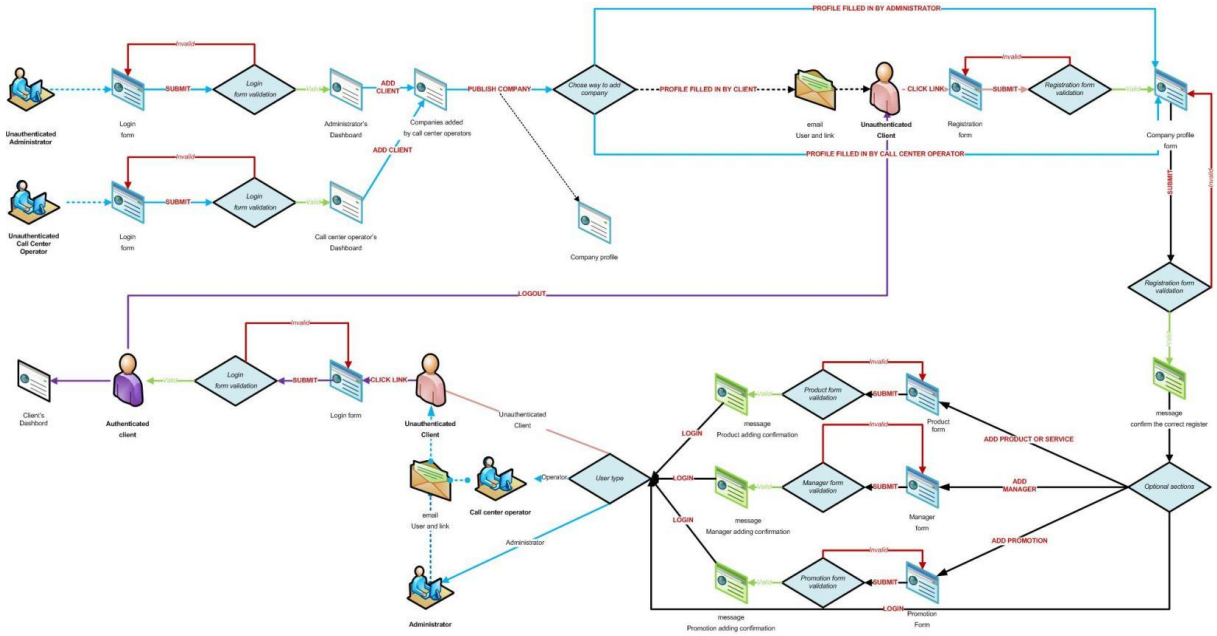

Fig. 2. UCD diagram for the ALPHA project's Register – Login – Logout process

DOI: 10.12948/issn14531305/19.2.2015.07

Building UCDs is an important process in understanding the structure of the application and it is also one of the first deliverables that the project owner comes into contact with. The ISDF methodology does not rely heavily on UCDs because building an innovative application is a very dynamic process and initial planning will change multiple times until the application is completed. The role of the UCD diagrams in the ISDF methodology is to help the project team gain a deeper understanding of the application and also provides the project owner with a preview of what the development's team is going to implement. In the ALPHA project a restriction was enforced of building a maximum of 10 UCD's and allocating a maximum of 2 hours for building each UCD. The *Planning* stage continued with building the database structure. Figure 3 presents a sample of the database structure built for the ALPHA application.

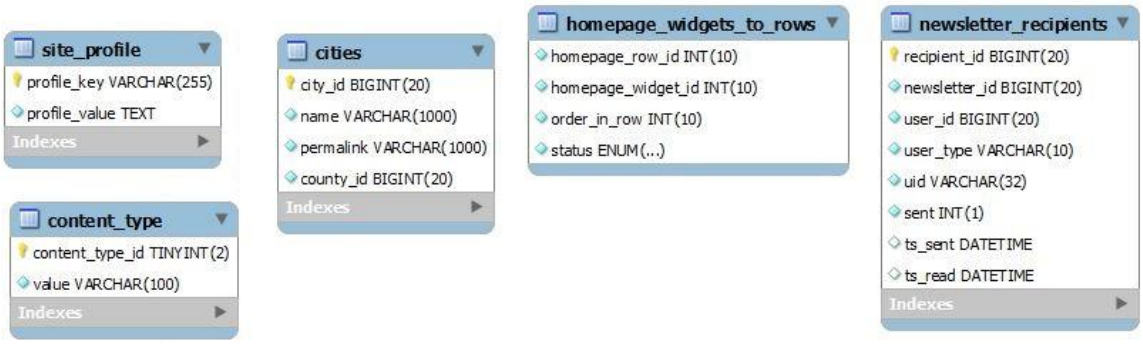

Fig. 3. Sample of the ALPHA application database structure

The role of the database structure in this stage of the ALPHA project was to help the project team gain a deeper understanding of the application. The database structure built in the Planning stage was not a mandatory requirement for the final application. The database structure changed significantly in three separate occasions by the time the project was finished. The Planning stage ended with the project manager and the project team deciding on what tools, technologies and process to employ in the development process of the ALPHA application. In terms of code versioning tools the project team decided to use Tortoise SVN. For the overall planning, resource allocation, budgeting and activity planning the project manager decided to use Microsoft Project. In terms of bug tracking, task assignment and progress monitoring the project manager and the project team decided on using Pivotal Tracker. In terms of technology the project team opted for the LAMP stack with CentOS as the Linux distribution. HTTP server of choice was Apache, SGBD system was MySQL and programming language PHP. In order to facilitate building on a MVC architecture the PHP Zend framework was chosen. The Planning stage ended with defining standards and quality measuring techniques. The ALPHA application was designed to be W3C, Yslow and Page Speed compliant. Data regarding quality was collected using the web application GTmetrix.

In the Prototype phase the project team built a mock-up of the application in order to validate the assumptions made in the Research and Planning stages. The mock-up was built using Prototyper. The prototype was built based on the UCD's developed in the Planning stage and acted as a proof of concept. The prototype of the ALPHA application was presented to the project owner for feedback, process represented in Figure 1 by transition 3. The prototype was not in accordance with the project owner's vision on the final application so the project team completely rebuilt the prototype, process represented in Fig. 1 by transition 6. After rebuilding the prototype the feedback received form the project owner required only minor adjustments to the prototype, process represented in Figure 1 by tran-

DOI: 10.12948/issn14531305/19.2.2015.07

sition 5. After the adjustments were implemented the prototype reflected accurately the project's owner vision on the final application. The prototype had to be built fast and it did not require any programming skills. In the ALPHA project the maximum time allocated for building a prototype was 3% of the total estimated project time and there were a total of 2 prototypes built. The Research, Planning and Prototype stages were executed in the spirit of the waterfall model and generated artefacts that are valuable in the context of innovative projects. After the prototype was approved by the project owner the Development and Design stage started simultaneously.

The Design stage consisted of a series of layout iterations were the graphic designer created a layout and made adjustments according to feedback received from the project owner, process represented in Figure 1. by transitions 10, 14 and 15. Building the functionality for the ALPHA application consisted of a series of iterations that were organized according to timeboxing technique. Each iteration was planned to last two weeks and ended with a functional version of the application. Deadlines were non-negotiable. Each iteration was built by adding functionality to the previous iteration. The ALPHA project was built in 8 iterations. An iteration included the Development, Testing and Setup stages. Development was performed in the spirit of the Agile methodologies with self-organizing teams and daily meetings to assess progress and to identify issues. Developers worked in pairs, with only one of them codding while the other was observing. Roles were exchanged daily. Pair programming reduces the number of bugs and increases the likelihood of delivering innovative solutions. Functionality was built following priorities set by the project owner. Functionality prioritisation was performed using the MoSCoW model.

Testing was performed using the testing scenarios defined in the Planning stage. Scenarios needed adjustments as the requirements for the ALPHA applications changed during actual implementation. The testing scenarios included all the instances of the ALPHA application. Figure 4 presents a sample of the testing schema used in the ALPHA application.

ID	type	UCD	UCD aproved	priority	development estimate (h)	start	finish	development	debugging	tested (D)	tested (G)	layout	estimate HTML	duration HTML	content	tested (D)	tested (G)	Overall OK
ADMIN																		
AP1	admin																	
AP2	admin																	
AP3	admin																	
AP4	admin																	
AP5	admin																	
AP6	admin																	
AP7	admin																	

Fig. 4. Sample of the testing schema used for the ALPHA application

The testing schema was designed for two testers. Each tester was involved in the development of the application starting from the Planning stage, when they contributed to building the UCDs, and ending with the Setup stage when they tested the application on the live environment.

The Setup stage entailed installing the applications on the live environment and adding proper content. Data was imported into the application's database in order to generate proper content. The first versions of the ALPHA application was installed on the live environment after the first development iteration, which was 5 weeks into the project, including research, planning, prototyping and design. After the first version of the application was installed on the live environment feedback was collected from the end-user and project manager. The role of the end-user was to provide feedback regarding usability, design, and functionality. In the ALPHA project after the code from the first iteration was installed and tested on the live environment application was tested by a sample batch of potential end-users. End-user testing was performed after each iteration. The Maintenance stage started after the code from the last it-

DOI: 10.12948/issn14531305/19.2.2015.07

eration was setup on the live environment. In the ALPHA project the Maintenance stage focussed on adding new functionality and improving existing functionality. Also an important part of the maintenance process was fixing design, architecture and functionality issues that were not identified in the Testing stage.

4 Quality Control in the ALPHA Application

Quality was handled in the ALPHA project by assessing the application's attributes in terms of compliance, usability, reliability, repeatability, availability and security.

Compliance is the extent to which the application's functionalities follow architecture, graphic design and user flows defined in the planning stage [15]. In order to assess the ALPHA application in terms of compliance the *Compliance degree*, indicator was used. The indicator is referred in the current paper using the *CD* acronym and is defined as follows:

$$CD = \frac{\frac{Fl+Fg}{Fp}+\frac{Fe}{Fi}}{3} \quad (1)$$

where:
Fl – number of missing functionality;
Fg – number of flawed functionality;
Fp – number of designed functionality
Fe – number of additional functionality;
Fi – number of actual functionality.
The *CD* indicator ranges in the [0,1] interval where an application with *CD* = 0 has an ideal compliance degree and an application with *CD* = 1 is an application with a very low compliance degree.

Table 1. Interpretation of the *CD* indicator

Interpretation of the *CD* indicator	***CD* indicator level**
High compliance degree	0
Acceptable compliance degree	(0 - 0,1]
Moderate compliance degree	(0,1 - 0,2]
Low compliance degree	(0.2 – 1]

Usability is given by how easy a user can access and use the application's functionality [15]. In order to assess the ALPHA application in terms of usability the *Usability degree*, indicator was used. The indicator is referred in the current paper using the *UD* acronym and is defined as follows:

$$UD = \frac{\sum_{k=1}^{nfa} \min(r_1 a_k, r_2 a_k, r_3 a_k ... r_{na_k} a_k)}{\text{nfa}} \quad (2)$$

where:
ra – number of actions required to reach the *a* functionality by using route r;
ak – *k* functionality;
nfa – number of application functionality;
na_k – number of possible routes to reach the a_k functionality.
The *UD* indicator ranges in the [1,100] interval where an application with *UD* = 1 has an ideal usability degree and an application with *UD* = 100 is an application with a very low usability degree.

Table 2. Interpretation of the *UD* indicator

Interpretation of the *UD* indicator	***UD* indicator level**
Optimal usability degree	1
High usability degree	(1 - 3]
Moderate usability degree	(3 - 5]
Low usability degree	(5 – 10]
Extremely low usability degree	> 10

Reliability is the speed of page loading, response times of different features and behav-

DOI: 10.12948/issn14531305/19.2.2015.07

ior when the application is accessed using low-speed Internet connections [15]. In order to assess the ALPHA application in terms of reliability the *Reliability degree*, indicator was used. The indicator is referred in the current paper using the *RD* acronym and is defined as follows:

$$RD = \frac{\sum_{k=1}^{nia} Vm_k}{nia} \quad (3)$$

where:
Vm_k – average loading speed for the k instance; variable expressed in seconds;
nia – number of application instances.
The *RD* indicator ranges in the [0,30] interval where an application with *RD* = 0 has an ideal reliability degree and an application with *RD* = 30 is an application with a very low reliability degree.

Table 3. Interpretation of the *RD* indicator

Interpretation of the *RD* indicator	***RD* indicator level**
Optimal reliability degree	[0 - 2]
High reliability degree	(2 - 5]
Low reliability degree	(5 – 10]
Extremely low reliability degree	(10 – 30]

Repeatability is determined by the degree of predictability, when seeking a specific outcome [15]. In order to assess the ALPHA application in terms of repeatability testing scenarios are used.
Consider *T* as the set of project testers defined by:

$$T = \{t_1, t_2, t_3 \dots t_i \dots t_{ntst}\} \quad (4)$$

where:
t_i – i tester;
ntst – number of project testers.
Consider *S* as the set of project test scenarios defined by:

$$S = \{s_1, s_2, s_3 \dots s_j \dots s_{nsct}\} \quad (5)$$

where:
s_j – j scenario;
nsct – number of test scenarios.
Consider *R* as the set of test results defined by:

$$R = \{r_1, r_2, r_3 \dots r_j \dots r_{nrez}\} \quad (6)$$

where:
r_j – j result;
nrez – number of test results.

The ALPHA application has a high repeatability degree if:

$$t_i(s_j) = r_j \; \forall \; i \in (1,2,3 \dots ntst), \forall \; j \in (1,2,3 \dots nsct) \quad (7)$$

Availability is the extent to which the application is accessible [15]. In order to assess the ALPHA application in terms of availability the *Availability degree*, indicator was used. The indicator is referred in the current paper using the *AD* acronym and is defined as follows:

$$AD = \frac{Taa}{Ttf} \quad (8)$$

where:
Taa – uptime; variable expressed in hours;
Ttf – total exploitation time; variable expressed in hours;
The *AD* indicator ranges in the [0,1] interval where an application with *AD* = 1 has an ideal availability degree and an application with *AD* = 0 is an application with a very low availability degree.

Table 4. Interpretation of the *AD* indicator

Interpretation of the *AD* indicator	***AD* indicator level**
Low availability degree	[0-0,94)
Medium availability degree	(0,94 – 0,97]
High availability degree	(0,97 – 0,99]
Ideal availability degree	(0,99 – 1]

Security represents the extent to which data and personal information are protected [15].

DOI: 10.12948/issn14531305/19.2.2015.07

In order to assess the ALPHA application in terms of security the *Security degree*, indicator was used. The indicator is referred in the current paper using the *SD* acronym and is defined as follows:

$$SD = \frac{Vi}{Vc} \quad (9)$$

where:

Vi – number of identified vulnerabilities;
Vc – number of known vulnerabilities.
The *SD* indicator ranges in the [0,1] interval where an application with *SD* = 0 has an ideal security level and an application with *SD* = 1 is an application with a very low security level.

Table 5. Interpretation of the *SD* indicator

Interpretation of the *SD* indicator	***SD* indicator level**
Low security degree	[1, 0,3]
Medium security degree	(0,3– 0,1]
High security degree	(0,1 – 0)
Ideal security degree	0

Interpretation for the quality indicators used in assessing the ALPHA application was performed empirically based on previously implemented software development projects.

5 Formalizing the ISDF Methodology

The development of the ALPHA application was performed using the ISDF software development methodology. By analysing the development of the ALPHA application, the ISDF methodology was formalized and presented in a structured manner. Table 6 presents a concise view on the ISDF software development methodology.

Table 6. ISDF software development methodology characteristics

Methodology characteristic	**ISDF Specific**
Roles	project owner; project manager; project team; end-user
Skills	associate; question; observe; experiment; networking
Team	9 individuals; self-organizing; emphasize informal and face-to-face communication
Tools	prototyping; code versioning; bug reporting; progress tracking; graphic design and workflow applications
Techniques	pair programming; timebox approach; MoSCoW prioritisation of tasks
Routines	30 minute meetings; daily written reports; weekly one hour meetings for planning or adjusting the current iteration
Artefacts	use case digammas; wireframes; prototypes; test case scenarios; database schemas
Processes and Activities	create artefacts; build prototypes; extend prototype using iterative development; collect continues feedback; developed testing scenarios before actual coding
Standards	W3C compliant; B grade by Yslow and Page speed standards; page size under 2 MB; less than 100 HTTP requests; average page load time under 5 seconds
Quality control	compliance; usability; reliability; repeatability; availability; security
Restrictions	no more than 30 minutes per daily meeting; no more than 10 UCDs; no more than 2 hours per UCD; no more than 3 prototypes; no more than 1% of the total estimated time allocated to building a prototype

DOI: 10.12948/issn14531305/19.2.2015.07

Core principles	early delivery of working software, welcome change, explore multiple implementation scenarios, non-negotiable deadlines, writing code over writing documentation

Roles of core importance for the ISDF methodology are project owner, project manager, project team and end-user. The role of the project owner is to provide accurate and detailed application requirements to the project manager and to provide continuous feedback. The ISDF methodology requires the project owner to be involved in every stage of the development life cycle. Project owner must provide feedback on all aspects concerning the application but most important components are: feedback on the prototype, feedback on each development iteration and feedback on design. The role of the project manager is to compile specifications based on requirements provided by the project owner, assemble the project team, design the overall flow of the application, define the implementation timeframe, design testing schemas, track progress and report to the project owner. The role of the project team is to plan the architecture of the application, choose the technologies required to build the application, design the database structure, design the graphical layout, implement functionality, test the application and setup the application on the live environment. The role of the end-user is to provide feedback on the functionality, design, security and usability of the application.

Skills required in developing innovative software and by that matter required in ISDF teams, are the ability to associate, observe, experiment, network and question. In the context of innovation, the ability to associate means being able to make connections across areas of knowledge. Transferring knowledge and ideas from other fields into software development is an abundant source of innovation. Sharp observation skills are a key element of innovation as it facilitates gathering data and information that eludes most people. When building a team the project manager should look for individuals with a network of vast connections. Being exposed to people with different backgrounds and perspectives increases your own knowledge. ISDF requires people with experimenting skills that build prototypes and pioneer new concepts and technologies. Questioning is essential for innovation as it is the catalyst for associating, observing, experimenting and networking skills [14].

Teams employed in innovative projects built using the ISDF methodology consist of maximum 9 individuals including the project manager. ISDF teams rely heavily on face-to-face communication. Empirical trials determined that teams larger than 9 individuals have issues with effectively conducting the daily and weekly meetings. Also project managers find it hard to properly go through more than 9 reports a day. ISDF teams are self-organized in terms of assigning tasks and building functionality. The project manager acts as a mediator to balance workload and solve conflicts.

Tools used in the ISDF methodology include prototyping, code versioning, bug reporting, progress tracking, graphic design and workflow applications. There are countless tools that can be used for the above mentioned tasks. Each team should choose tools that they are familiar with, that suit their budget and comply with their company culture. For instance in the ALPHA project Prototyper was used for building the prototype, code versioning was performed using Tortoise SVN, bug reporting and progress tracking was performed using Pivotal Tracker, graphic design was performed in CorelDraw and workflows were performed using Microsoft Visio. ISDF is not a methodology that focuses on tools but it definitely tries to exploit them as much as possible. Using the same tools over and over will allow the project manager to reuse artefacts from past projects.

Techniques used in the ISDF methodology concern programming, tasks prioritisation and time management. ISDF relies on pair programming technique to reduce the number

DOI: 10.12948/issn14531305/19.2.2015.07

of bugs, increase solution diversity, build collaboration networks and stimulate learning. ISDF uses the timebox approach for project planning in order to increase focus and avoid missing deadlines. In the ISDF methodology prioritisation of tasks is accomplished using the MoSCoW technique in order to ensure early delivery of the most valuable functionality.

Routines enforced by the ISDF methodology consist of daily 30 minute meetings, daily written reports, weekly one hour meetings for planning or adjusting the current iteration. Every morning team members meet together with the project manager and share progress on their work. A special emphasises on these meetings is to identify and eliminate factors that inhibit progress on tasks. Daily written reports are sent by the team members to the project manager at the end of each working day. Reports contain details on the tasks performed that particular day and also allow the team members to transmit more sensitive information to the project manager; information that they are not comfortable sharing with the rest of the team in the daily meetings. Weekly meetings are for planning or evaluating the overall progress of the iteration. Each iteration begins with a weekly meeting where tasks are assigned to team members. Task assignment is a collaborative process as ISDF teams are self-organized, the project manager only intervenes to mitigate conflict or to help overcome deadlocks.

Artefacts generated by the ISDF methodology consist of use case digammas, wireframes, prototypes, test case scenarios and database schemas. In innovative software development application artefacts are very important because they are required in the process of protecting intellectual property rights like obtaining patents. Innovative software development projects often result in applications that incorporate valuable new technologies or processes that are subject to intellectual property laws. Artefacts are also valuable assets when new team members join the project. In the ISDF methodology all artefacts, except database schemas, are generated by the project manager. The database schema is generated by the project team.

Process and activities critical to the ISDF methodology are represented by creating artefacts, building a prototype, codding and extending the prototype using iterative development, collecting continues feedback and developing testing scenarios before actual coding. ISDF is a methodology focused on coding but creating software artefacts is a critical process in implementing innovative applications as it facilitates protecting intellectual property rights and it helps mitigate risks. Innovation is based on an idea. In order to tests the feasibility of the idea building a prototype is required. Prototype can also help secure additional funding for an innovative project. Codding and extending the prototype is performed by using iterative development. Building an application in multiple iteration allows for better tolerance to changing requirements as is the case in innovative projects. A critical process of the ISDF methodology is collecting feedback from the project owner and from the end-user. Feedback from the project owner is collected in every stage of the development lifecycle. Feedback from the end-user is collected after the first iteration code is setup on the live environment. The testing process begins after codding for the first iteration is finished. Testing scenarios are written by the project manager and by the testers before the actual codding process begins.

Standards within ISDF methodology regard codding best practices, page size, HTTP requests and average page loading time. ISDF requires that all pages be W3C compliant unless breaking best practice guidelines was performed intentionally in order to boost performance. Also requires a B grade by Yslow and Page speed standards for all pages. ISDF enforces page size under 2 MB and less than 100 HTTP requests to load a page. To optimize user experience average page loading time should be below 5 seconds.

Quality control in the ISDF methodology concerns compliance, usability, reliability, repeatability, availability and security. Compliance is assessed by the degree in which functionality architecture, graphic design and

DOI: 10.12948/issn14531305/19.2.2015.07

user flows adhere to project owner specifications. Usability is determined by the ease with which a user accesses and uses an application's functionality. Reliability is determined by loading speed and response times. Reliability also requires for applications developed with ISDF methodology to take into account users that have access to low-speed Internet connections. Repeatability of a web application is determined by the degree of predictability, when seeking a specific result. Availability is determined by the extent to which the application is accessible. Security is determined by the extent to which data and personal information are protected [15].

Restrictions enforced by the ISDF methodology concern time and resources allocated for activities. Imposing restrictions ensures that project does not stray from its original goals, follows the planned timeframe and does not exceed initial budget. In the ISDF methodology the maximum length of an iteration is two weeks and the minimum length is one week. The daily meetings must not exceed 30 minutes. No more than 10 UCD's are created per project and building a UCD should not take more than 2 hours. No more than 3 prototypes are built per project and building a prototype should not take more than 1% of the estimated project timeframe.

Core principles characterizing the ISDF methodology consist of early delivery of working software, welcoming change, exploring multiple implementation scenarios and actively involving project owner into all project stages. ISDF values writing code over writing specifications. ISDF emphasizes design over documentation. Though planning is not overlook development is always prioritized. The project owner decides the priority of tasks and deadlines are non-negotiable.

6 Conclusions

Research results presented in the current paper are confined to the web application development field and were not tested on projects with a timespan larger than 14 months or on project teams consisting of more than 10 individuals. Innovative software development projects require a dedicated software development methodology that accounts for frequent change of specifications, high dynamics of technology and standards, higher than usual risks, proprietary cost structure and custom testing scenarios. The ISDF methodology was developed empirically by trial and error in the process of implementing multiple innovative projects. The current version of the ISDF methodology was refined by reviewing scientific literature and incorporating valuable elements from the waterfall and agile development models. The waterfall model provides support for generating software documentation which is valuable in the case of innovative software development. The agile model provides a process capable of coping with frequent change of requirements as this is frequently the case in innovative software development projects. The roles enforced in the ISDF methodology are project owner, project manager, project team and end-user. The ISDF methodology employs tools for prototyping, code versioning, bug reporting, progress tracking, graphic design and workflow applications. The routines proprietary to the ISDF methodology are daily 30 minute meetings, daily written reports and weekly one hour meetings. The artefacts generated by the ISDF methodology consist of use case digammas, wireframes, prototypes, test case scenarios and database schemas. In terms of software development techniques ISDF methodology relies on pair programming, timebox approach and MoSCoW prioritisation of tasks. The following are processes and activities proprietary to the ISDF methodology: creating artefacts, building prototypes, extending prototypes using iterative development, collecting continues feedback and developing testing scenarios before actual coding. Standards of the ISDF methodology enforce W3C compliance, Yslow and Page speed B grades, less than 100 HTTP requests to load a page, page size under 2 MB and page loading time under 5 seconds. Quality control regards compliance, usability, reliability, repeatability, availability and security. As a future research topic, ISDF methodology can be scaled in order to accommodate software development projects

DOI: 10.12948/issn14531305/19.2.2015.07

that require larger teams.

Acknowledgment
This paper was co-financed from the European Social Fund, through the Sectorial Operational Programme Human Resources Development 2007-2013, project number POSDRU/159/1.5/S/138907 "Excellence in scientific interdisciplinary research, doctoral and postdoctoral, in the economic, social and medical fields -EXCELIS", coordinator The Bucharest University of Economic Studies.

A shorter version of this paper, titled Software Development Methodology for Innovative Projects - ISDF Methodology, has been presented at the 14th International Conference on Informatics in Economy, Education, Research and Business Technologies.

References

[1] T. DeMarco, "The role of software development methodologies: past, present, and future", Proceedings of the 18th international conference on Software engineering, 25-30 Mar. 1996, Berlin, Germany, Publisher: IEEE, ISBN: 0-8186-7246-3, pp. 2-4

[2] M. R. J. Qureshi, "Agile software development methodology for medium and large projects", IET Software, vol.6, no.4, pp.358-363, doi: 10.1049/iet-sen.2011.0110

[3] K. Petersen, C. Wohlin and D. Baca, "The Waterfall Model in Large-Scale Development", Proceedings of the 10th International Conference on Product-Focused Software Process Improvement, 15-17 Jun. 2009, Oulu, Finland, Publisher Springer Berlin Heidelberg, ISBN 978-3-642-02151-0, pp. 386-400

[4] S. H. VanderLeest and A. Buter, "Escape the waterfall: Agile for aerospace", Proceedings the 28th Digital Avionics Systems Conference, 23-29 Oct. 2009, Orlando, USA, Publisher: IEEE, doi: 10.1109/DASC.2009.5347438, pp. 6.D.3-1- 6.D.3-16

[5] T. Dyba and T. Dingsoyr, "What Do We Know about Agile Software Development?", IEEE Software, vol.26, no.5, pp. 6-9, doi: 10.1109/MS.2009.145

[6] B. V. Thummadi, O. Shiv and K. Lyytinen, "Enacted Routines in Agile and Waterfall Processes", Proceedings of the 2011 Agile Conference, 7-13 Aug., Salt Lake City, USA, Publisher: IEEE, 2011, doi: 10.1109/AGILE.2011.29 pp. 67-76

[7] P. Trivedi and A. Sharma, "A comparative study between iterative waterfall and incremental software development life cycle model for optimizing the resources using computer simulation", Proceedings of the 2nd International Conference on Information Management in the Knowledge Economy, 19-20 Dec. 2013, Chandigarh, India, Publisher: IEEE, pp. 188-194

[8] D. Duka, "Adoption of agile methodology in software development", Proceedings of the 36th International Convention on Information & Communication Technology Electronics & Microelectronics, 20-24 May 2013, Opatija, Croatia, Publisher: IEEE, ISBN: 978-953-233-076-2, pp. 426-430

[9] S. Zhong, C. Liping and C. Tian-en, "Agile planning and development methods", Proceedings of the 3rd International Conference on Computer Research and Development, 11-13 Mar. 2011, Shanghai, China, Publisher: IEEE, doi: 10.1109/ICCRD.2011.5764064, pp. 488-491

[10] J. A. Livermore, "Factors that impact implementing an agile software development methodology", Proceedings of the 2007 IEEE SoutheastCon, 22-25 March 2007, Richmond, USA, Publisher: IEEE, doi: 10.1109/SECON.2007.342860, pp.82-86

[11] T. J. Lehman and A. Sharma, "Software Development as a Service: Agile Experiences", Proceedings of the 2011 Annual SRII Global Conference, 29 Mar. - 2 Apr. 2011, San Jose, USA, Publisher: IEEE, doi: 10.1109/SRII.2011.82, pp. 749-758

[12] A. Cockburn, "Selecting a project's methodology", IEEE Software, vol.17, no.4, pp. 64-71, doi: 10.1109/52.854070

[13] R. Klopper, S. Gruner and D. G. Kourie, "Assessment of a framework to compare software development methodologies", Proceedings of the 2007 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries, Sunshine Coast, 30 Sep. - 03 Oct. 2007, South Africa, Publisher: IEEE, doi: 10.1145/1292491.1292498, pp. 56-65

[14] C. M. Christensen, J. Dyer and H. Gregersen, The Innovator's DNA: Mastering the Five Skills of Disruptive Innovators, Publisher: Harvard Business Review Press, pp. 304, ASIN: B0054KBLRC

[15] M. Despa, I. Ivan, C. Ciurea, A. Zamfiroiu, C. Sbora, E. Herteliu, "Software testing, cybernetic process", Proceedings of the 8th International Conference on Economic Cybernetic Analysis: Development and Resources, 1-2 Nov. 2013, Bucharest, Romania ISSN 2247-1820, ISSN-L 2247-1820.

Mihai Liviu DESPA and has graduated the Faculty of Cybernetics, Statistics and Economic Informatics from the Bucharest Academy of Economic Studies in 2008. He has graduated a Master's Program in Project Management at the Faculty of Management from the Bucharest Academy of Economic Studies in 2010. He is a PhD Student at the Economic Informatics PhD School and he is currently Project Manager at GDM Webmedia SRL. His main field of interest is project management for software development.

DOI: 10.12948/issn14531305/19.2.2015.07