A Systematic Approach to the Comparison of Roles in the Software Development Processes

Murat Yilmaz[1], Rory V. O'Connor[2], and Paul Clarke[1]

[1] Lero Graduate School in Software Engineering, Dublin City University, Ireland
[2] Lero, the Irish Software Engineering Research Centre, Dublin City University
{murat.yilmaz,roconnor,pclarke}@computing.dcu.ie

Abstract. The vision of building a successful software product requires teams of individuals equipped with a wide range of social and technical skills. Furthermore, by combining these skills with appropriate job roles, we should be able to improve the productivity of a software organization. In order to identify and compare different roles in software development activities, we conduct a systematic comparison of software development models, covering traditional approaches through to agile techniques. To compare the roles in the literature with industrial software landscapes, we use data from a survey conducted on 266 software practitioners to ascertain job roles in two middle size software companies, one of which uses traditional methods and in particular ISO/IEC 12207 for managing their software development activities while other uses a tailored agile methodology. In light of our interviews, we found that based on project specific needs, the roles used in industry vary significantly from the roles defined in literature.

1 Introduction

Software development is a complex socio-technical activity, which relies on teams of individuals working harmoniously. Therefore, individuals should be able to cope with challenges embedded in software development tasks. These tasks, however, should be performed as teamwork to accomplish a particular contract with stakeholders [1]. During these activities, the socio-technical skills of individuals are an important consideration when forming teams. As mentioned in every software development methodology, there are job roles for individuals to be assigned. A role is a series of expectations from an individual mostly for team-based activities that are defined in a social context or a situation.

Furthermore, from an industrial perspective, the actual success of customizing a methodology not only depends on the methods we choose but also the roles that are included in a software development method. Therefore, understanding these roles and systematically selecting a set of suitable roles for a proposed methodology has several merits. Firstly, the role selection process helps us to control the flow of information to manage the activities in a software company. Consequently, roles convey a value to the development methodology [2]. Software development is not easy, it needs dedicated personnel. However, evidence suggests that individuals should be more effective in settings such that roles are well-defined [3]. Secondly, role selection can be used for

A. Mas et al. (Eds.): SPICE 2012, CCIS 290, pp. 198–209, 2012.

stimulating individuals. Agile methods cause alterations in several roles or job titles previously defined in traditional software development. This realignment has weakened some of the traditional roles to some extent: therefore even some practitioners think that agile reduces the ability of managers to command their teams [4]. Thirdly, organizing the roles for the software development methodology can be considered as a form of software quality assurance activity in order to improve the product quality [5].

In this paper, we constitute a systematic comparison framework based on actualized roles and defined roles in the software development processes. We formalize our research question as: "*In practice, do software development roles differ from the role definitions provided by the software development process methodologies?*" To this end, we review the literature to single out the set of defined roles for the selected software development processes and systematically compare them with the roles that are used in industrial settings. Based on a case study with two middle size software companies, we first use the data collected on our surveys to understand the working roles or titles in an industrial software organization, and secondly we interview software practitioners to validate our results.

The remainder of this paper is structured as follows: In section two, we introduce our research viewpoint, which defines our systematic approach that enables the comparison of different roles. The following section reviews the roles identified in literature for the different software development processes. The next section evaluates our approach by analyzing of data gathered from the case studies we conducted in two middle size software companies. The last section will conclude the paper with a brief summary of contributions.

2 Research Overview

The first part of our systematic approach starts with constructing our research goal to evaluate whether there is a significant amount of difference in previously identified roles and their actualizations especially when tailoring a role-based task assignment in software development. Next, we survey the literature for the roles for both traditional and agile methodologies that are mentioned in software development literature. We selectively chose software methodologies and processes and work on the roles that are defined by these approaches. In technical terms, we conduct a thematic content analysis (i.e. descriptive presentation of this literature review) based on roles as the units of analysis. After identifying software development roles in the literature, secondly we conduct a focus group study with one of our industrial partners, where we seek opinions about actual roles that are used in their company. We initiate the focus group conversation by using some parts on our previously conducted survey, in which we ask participants about their organizational roles and experience levels on that role (see figure 1). Secondly, we interview team leaders and development managers about how accurate the actualization of the job roles.

Content analysis is an organized study of characteristics found in a content of any type of communication, such as books, websites, newspapers, etc [6]. Our approach uses the content analysis technique for making interpretations to create a role

selection schema based on literature of roles in software development methodologies. Based on the survey data collected previously, these roles will be systematically compared to their industrial actualizations. To this end, we first collect data from literature and consult industry about the defined roles frequently used in software engineering settings. Secondly, we conduct a focus group, where we record the session and a content analysis was performed on participants' definition of roles that are actualized in software development landscapes.

We form a number of acronyms based on the roles that are found from the literature. Here, we are making partial use of a coding mechanism to construct a role-based schema with the defined roles from the literature. The coding aims to create variables based on the roles defined in software development. It is done for easy comparison of roles by constructing a unique key for each role found from the literature. Our coding schema allows us to observe the commonalities and differences between software engineering roles. It helps us to investigate cause-effect relationships, interrelationships, and situational conditions for each role category. Here, we design several questions to seek validity for our coding in the defined categories, and analysis of identified roles from the literature.

- Is this role the same as a role in the other categories?
- Are there any duplicated role codings in a category?
- In which context do these roles emerge?
- What kind of roles have changed or evolved in emerging methods?
- Is there any observable change for other roles when a role evolved to an other form (i.e. covariance between categories)?

The *objective coding* [7] is a technique to review a collection of documents for extracting and indexing the information so as to form a new perspective on representing the data. We use an objective coding scheme on the collected information of roles. This coding should be helpful for visually comparing actualized roles systematically with the ones cited in the literature. In addition, a diagram is drawn to support the development of the relationship among roles (see Figure 1).

Finally, we aim to formulate a framework for software practitioners, which enable them to select proper roles for their software development methodologies. Consequently, by using such a framework, a software practitioner may easily choose or customize the necessary roles for his or her development activities.

3 Roles in Software Development Processes

Many different variants of development models and methodologies have been created. In this section, we survey the roles that are defined in the literature starting from traditional software development and working through ISO/IEC 12207, and agile methodologies such as extreme programming (XP), scrum and feature driven development (FDD).

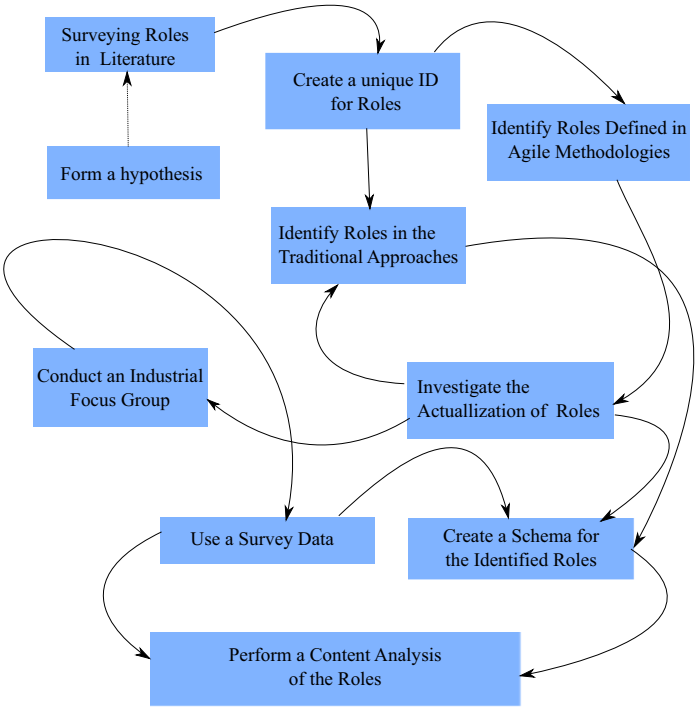

Fig. 1. Our Systematic Approach for Investigating the Roles in Software Development Environments

3.1 Roles in Traditional Software Development

Software engineering teams address the complex problems of software development by sharing the tasks among its members with respect to their roles. Roles are the descriptions of duties or assignments and competence for participants that are required to achieve a defined tasks and activities of software development [8]. In his essay, *The Cathedral and the Bazaar*, Raymond states that because of the strict roles defined in traditional software development, traditional approach is similar to building a cathedral, where a small team of people working in an isolated environment [9]. Therefore, this could be considered as a drawback because several artifacts are only visible for a limited number of individuals in this setting.

Traditional roles include: *Project manager* who is responsible for allocation of resources, project expenditures, and responsible from the general objectives of a software project. Another typical role in the development processes is the role of a developer. A *software developer* is responsible for designing and maintaining the software programs, whereas a *software tester* is responsible for creating test plans and testing the developed programs. In many cases *user interface designers* (design screen interfaces), *database designers* (design database schema) and the *software architects* (design technical blueprints) are also included as a generic software practitioner category.

Table 1. Traditional Software Development Roles

Code	Role Name	Primary Type of Value
PM	Project Manager	Resource Allocation and Budgeting
SD	Software Developer	Development Activities
UID	User Interface Designer	Design Screen Interfaces
DD	Database Designers	Data Modeling
SA	Software Architects	Software Modeling
BA	Business Analyst	Stakeholder Management
RE	Requirement Engineer	Gathering Requirements
SQA	Software Quality Assurance	Creating and Maintaining Quality
SAN	System Analyst	Construction of a System

A business analyst is not only responsible for solving the problems by regulating the connections between the business and the technical people but also for documenting several parts (e.g. requirement documents) of a software project. In addition to these roles some others can also be seen regarding several needs; e.g. requirements engineer, systems analyst, software quality assurance engineer (see Table 1) .

Table 2. Systems Engineering Roles and their values from [10]

Code	Role Name	Primary Type of Value
RO	Requirements Owner	Understanding Need
SD	System Designer	Accomplishing work
SA	System Analysis	Reducing Risks
VV	Validation & Verification	Mitigating Risks
LO	Logistics and Operations	Understanding need
G	Glue among the subsystems	Accomplishing work, Reducing Risks
CI	Customer Interface	Understanding the Need
TM	Technical Manager	Technical Management
IM	Information Manager	Knowledge Management
PE	Process Engineer	Managing and Understanding Needs
CO	Coordinator	Organizational Management
CA	Classified Ads SE	Accomplishing Work (assumed)

Sheard [11] identifies twelve roles (see Table 2) of development from system engineering viewpoint while investigating the relationship between the roles and their importance for creating a value. This work not only suggests that the value is asserted in qualitative terms and it should be quantified in further research but it also claims that it should be observed as a requested improvement within a product by better (i) definition of the requirements, (ii) management strategies, (iii) ways for mitigating risks, (see [10] for details).

3.2 Roles in ISO/IEC 12207

ISO/IEC 12207 [12] has three main groups of roles for its participants. The first group consists of the principal roles are the *acquirer*, who is a form of stakeholder that obtains products or services from *supplier*, who is an individual or another organization agree on providing a software products or services. *Implementer* executes development tasks, while the *maintainer* can be either an organization or an individual who performs the upkeep of developed software), and *operator* is responsible for the execution of a

system [12]. The second category consists of configuration and supporting roles; the *configurator* is responsible for the establishment and transformation of the information needed by an individual or a group, *evaluator* tests and measure a software process or a product by using the data collected during the actual tasks that are performed, the *auditor* investigates the products and processes are compatible with the agreements, the *usability specialist* deals with the demands and needs of the stakeholders such as the design activities based on human factors and skills and their fulfillment [12].

Table 3. Roles in ISO/IEC 12207 (adapted from [12,13])

Code	Role Name	Primary Type of Value
AC	Acquirer	Software Client or User or Product Owner
SU	Supplier	Software Producer, Product Seller
IMP	Implementer	Realization of Development Tasks
MN	Maintainer	Maintain the Software
OP	Operator	System Execution
CON	Configurator	Accomplishing Work, Reducing Risks
EV	Evaluator	Test & Measure a Process or a Product
AU	Auditor	Contract Management
US	Usability Specialist	Problems Regarding to People Factors
MA	Manager	Managing
AM	Asset Manager	Managing Assets
CM	Knowledge Manager	Knowledge Management
RA	Reuse Administrator	Seeking for Reusable Parts

The third group has the organizational roles, the *manager* identifies and manages the state of the play (i.e. condition and progression of the project) with respects to project constraints (e.g. objectives, budget, schedules), the *asset manager* is a type of manager deals with the management and optimization of the assets regarding to the plan he or she prepared, the *knowledge manager* role works on the collection of particular knowledge and skills throughout the organization and used for improvement for the products and services. The *reuse program administrator* seeks to find favorable or advantageous circumstances for reusable parts of a product or a service. Unlike the other two subfields of software engineering (i.e. requirements engineering and software development), *domain engineer* is a form responsible for designing the domain models (i.e. software models) and domain descriptions for a software system (see Table 3).

3.3 Roles in Extreme Programming

According to Beck [14], the participants and their roles are as follows; *Programmers* are the individuals who need to have good communication and collaboration skills for both team and individual levels. They are responsible for developing, maintaining and testing the software. One of their main responsibilities is to ensure that their work is clean and lean. The technical decisions are made by programmers. *Customers* form the steering teams in business terms and in particular in requirement satisfaction decisions. *Testers* help customers to write functional test cases. Business decisions are made by customers [14]. The *tracker* role composes a trace and feedback mechanism in XP. The estimations, goals and iterations made by teams are controlled by a tracker, who provides feedback. The *tracker* is also responsible for measuring constraints such as scarce

resources and delivery times versus goal evaluation. The *coach* is the role which is accountable for XP project who needs to understand the problems occurring during the process to instruct team members and transfer the information or sometimes experience among teams and individuals. Finally, the *manager* is responsible for final decisions, and also an aim of this role is to recognize problems likely occur during the development life-cycle (see table 4).

Table 4. Roles in XP (adapted from [14,15])

Code	Role Name	Primary Type of Value
PRG	Programmers	Maintaining and Testing Software
CU	Customers	Managing Business Decisions
TST	Testers	Helps Costumers for Functional Test Cases
TRC	Tracker	Feedbacks and Estimations
CO	Coach	Supervise Team
CON	Consultant	Guides the Team for Problem Solving
MA	Manager	Management

3.4 Roles in Scrum

Schwaber and Beedle [16] single out six roles for the participants of Scrum. The *Scrum Master* is a type of management role specific to Scrum, who is responsible for the alignment of practices and rules as they have organized. This role interacts not only with project team but also customer and management. Its aim is to maximize productivity by practicing the agile and scrum values and monitoring the team to avoid any kind of complications. The *Product Owner* is the role which is responsible for exercising the project management and control activities. Additionally, this role is also responsible for transforming the product backlog into product features. *Scrum Team* should be considered as a self organizing structure to produce a working piece of a product, where its main goal is to achieve time targeted objectives of each sprint. The *customer* role will continuously evaluate the backlog items, and helps the selection for a sprint. The *management* role is responsible for implementing the proper standards for the software development process. Additionally, this role encompasses decision making activities and finalizing them at different stages of development process such as evaluating goals, gathering requirements, etc. (see Table 5).

Table 5. Roles in SCRUM (adapted from [16])

Code	Role Name	Primary Type of Value
SM	Scrum Master	Managing Scrum Team
PO	Product Owner	Product Management Decisions
CUS	Customer	Evaluation of backlog items
ST	Scrum Team	Organized itself for time boxed goals
MNG	Management	Evaluate Decisions and Goals
USR	User	Evaluate System Functionalities

3.5 Roles in FDD

FDD has the most comprehensive role description with a flexibility of roles [17]. For example, an individual can play multiple roles, or either a role can be shared by multiple persons [15]. The three main categories of roles, which are: *key, supporting and additional* roles. The key roles are *project manager*, who administers the entire project and maintains the work settings of the software team, the *lead software architect* is the role which makes the appropriate decisions for software development, the *software development manager* is a role which focuses on daily activities and team negotiations during the software development activities. The *lead programmer, the class owner and the domain expert* are the three roles used in FDD. The supporting roles includes; *manager (release), knowledge expert, build process engineer, toolsmith* and *system administrator*. Moreover, *testers*, *technical document expert* and *software deployment personnel* are the other roles used in this practices [17](see table 6).

Table 6. Roles in FDD (adapted from [17,15])

Code	Role Name	Primary Type of Value
PM	Project Manager	Resource Management
LSA	Lead Software Architect	Architectural Decisions
DEM	Development Manager	Evaluation of backlog items
LP	Lead Programmer	Organized itself for time boxed goals
CO	Class Owner	Form Teams for Implementing Features
DE	Domain Expert	Inform Teams for Adequate Features
RM	Release Manager	Managing the development process
DM	Domain Manager	Managing Domain Experts
LG	Language Guru	Acquiring a Knowledge on Technology
BE	Build Engineer	Executing a Build Process
TA	Toolsmith	Creating Utilities for project
SYA	System Administrator	Administration of Work Systems
TE	Testing	Verifying the Actualization of a System
DEP	Deployer	Release of Feature Deployment
TEW	Technical Writer	The Documentation for Users

4 Evaluation of Roles from Industrial Settings

As a part of a survey, we asked 266 participants from two different software companies about their roles in their applied settings in order to identify the commonality of meaning in the different roles. One of the software companies (with a staff about 400 personnel) is working in telecommunication sector, which composes solutions for large-scale e-government projects. The other company supplies turn key software solutions to telecommunications operators and mobile service providers. It has a staff of about 40 personnel. By creating a list of roles based on the roles mentioned in the literature, we conduct a focus group in one of the companies about the actualization of roles in development environments. This brings individuals together to debate about software development roles in their company and their actualizations with respect to their experiences. Next, we ask our research question to a selection of people mostly to the individuals from the management teams.

Company A is using the traditional software development approaches to define the roles: PM, SD, UID, SA, BA, SQA, where DD is embedded in SD, and RE role is somehow split with BA and SD. The role of system analyst provides the requirement engineering processes.

> ***Interview quotation:*** *"During our development activities, we observe lots of overlapping roles, which sometimes hinder our ability to handle some development tasks. For example, some of our teams have key players with overlapping roles and some individuals perform more than one role by the nature of our development process. We found it interesting to have a big picture of the roles in the different software development processes."*

Company A uses ISO/IEC 12207 combined with an iterative development schema and a customized role selection based on the traditional viewpoint for developing and maintaining software project. However the roles defined by ISO/IEC 12207 are not fully used to profile the personnel. Instead, they use the role names (see Table 1) that are traditionally used in software development.

> ***Interview quotation:*** *"We use approximately 14 out of 43 processes, 60 out of 95 activities, 180 out of 406 tasks from ISO/IEC 12207. We believe that assigning suitable roles to teams and individuals is very important for our success. A review of roles in different methodologies is useful from an industrial perspective. All type of roles should be visible to everyone in the company, and they should be defined in a simple language to provide a way of ensuring everyone understands them. Therefore, we are not using the role names provided by ISO/IEC 12207. I would say, we mostly use the classical role names you have mentioned."*

According to the management team of Company A, the role of team leader should not dictate anything to teammates but communicated the vision of a company or a project. Therefore, maintaining a friendship and trust is more important than dictating the facts to software teams.

> ***Interview quotation:*** *"People usually trust other people to some extent. There are always problems, when it comes to role assignment as well as delegations based on these roles. I personally observed several situations, where improper delegation did cause lots of conflicts and tensions. I would strongly suggest that role tailoring should not be taken lightly."*

Company B uses a customized agile methodology, which relies on XP and Scrum. They use agile methodology so as to cope with dynamically challenging requirements and to fulfill the request of their customer for continuous integration with small increments. They use all roles defined by scrum (i.e. SM, PO, CUS, ST, MNG, USR) and a tester role (TST) and a progress tracker (TRC) role from XP.

> ***Interview quotation:*** *"There is the notion of tailoring methodologies, how about the roles? It is always a problem for us to select the suitable roles for our customized methodology. Therefore, broader view of roles in software development activities are very important for us. However, just as there is no one-size-fits-all methodology for developing applications in software development, there should not be a one-size-fits-all approach to role selection."*

Finally, Company B highlights the importance of face to face communication for agile landscapes, and therefore selection of suitable roles for development activities becomes more important.

> ***Interview quotation:*** *"The process of customization of roles is very important particularly in agile development environments. A summary with roles contained in different agile approaches is very helpful for us to see the suitable roles for our process."*

5 Conclusions

In this paper, we highlight how roles in literature and their actualizations on industrial environments vary for both plan driven and agile methodologies. Software development is a collaborative endeavor that depends on its development methodology. However, selection of a proper methodology is not enough for achieving goals of a software organization. The evidence suggests that we should also tailor the necessary roles depending on development activities.

After analyzing the defined categories in light of the questions above, we confirmed that several roles presented in older methods are emerged with a different name, with similar responsibilities in newer approaches. Some of the roles, however, have their responsibilities changed while revealing in different software development organizations. Most frequently, the role definitions that an organization uses based on a domain and a set of circumstances.

Here, we present role orientations for the selected software development methodologies as shown in Table 7. We identify four types of role orientations: Actor-based, activity-based, artifact-based and methodologies with extended role definitions based on a previously defined role. For example, both Scrum and FDD have actor-based roles, in which the skills of an individual are defined by the role characteristics such as product owner or a class owner. In addition, all methodologies have activity-based roles

Table 7. Comparison of roles for the selected development methodologies

		Role Orientations			
		Actor Based	Activity Based	Artifact Based	Extended
Models	Traditional		✓		
	System Engineering		✓		✓
	ISO/IEC 12207		✓		
	XP	✓	✓	✓	
	Scrum		✓	✓	
	FDD	✓	✓	✓	✓

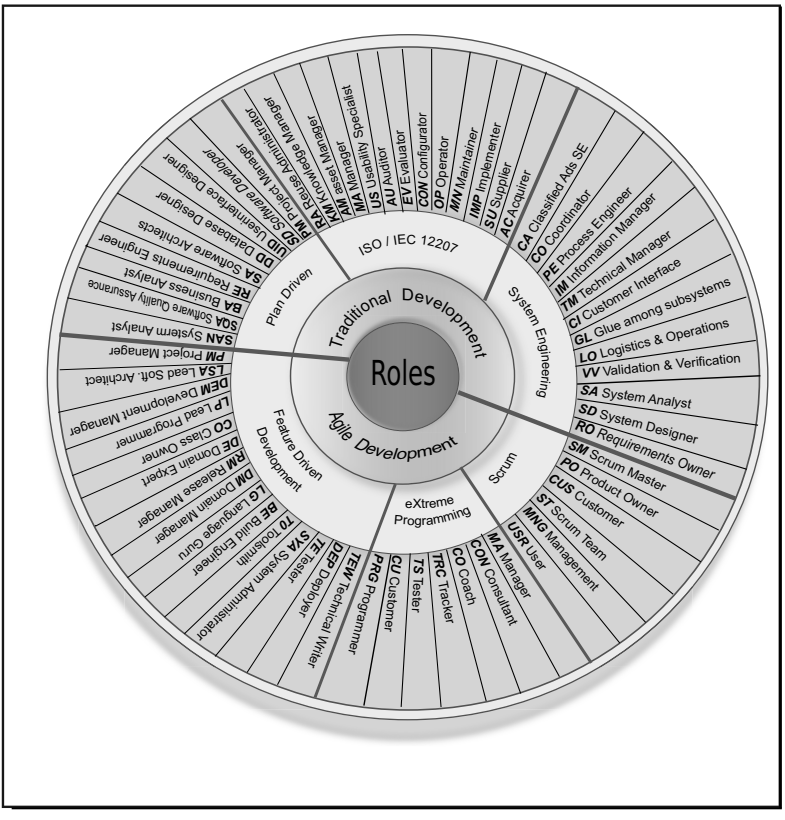

Fig. 2. A Summary of Roles Contained in the Different Approaches

such as software developer, a software tester, etc. We also consider roles that are based on a creation of an artifact, which are highlighted by the agile methodologies. Finally, extended roles are the roles that can be integrated or shared among the individuals such as the roles like the domain expert role, which somehow comprises the technical writer role in FDD.

Our study exhibits that a role-based schema can be useful for a tailoring process of roles regarding to the organizational needs. Furthermore, we argue that a software development organization should customize their own roles suitable for their social structure, where we suggest that our role based construct (see Figure 2) will be beneficial for such activities. In other words, it enables them to select proper roles for their software development methodologies. Consequently, by using such a framework, a software teams may easily choose or customize the necessary roles based on their activities.

Analysis of identified roles from the literature is portrayed in Figure 2. We can confirm that several roles presented in older methods are emerged with a different name with similar responsibilities in newer approaches. The roles, however, mostly have their responsibilities changed and reappeared as another form while revealing in different

software development organizations. Most frequently, the role definitions that an organization uses based on a domain and a set of circumstances. Moreover, it is important to choose roles, based on the social structure of an organization and required interactions. These customized roles are found to be organizational centric, which also clearly supports the notion of *separation of concerns* [18].

Acknowledgments. This work is supported, in part, by Science Foundation Ireland grant number 03/CE2/I303-1 to Lero, the Irish Software Engineering Research Centre (www.lero.ie).

References

1. Humphrey, W.: Introduction to the team software process (sm). Addison-Wesley Professional (2000)
2. Hazzan, O., Dubinsky, Y.: Agile Software Engineering. Undergraduate Topics in Computer Science. Springer (2008)
3. Cooper, D., Sutter, M.: Role selection and team performance. In: Working Papers in Economics and Statistics. University of Innsbruck (2011)
4. Larman, C.: Agile and iterative development: a manager's guide. Addison-Wesley Professional (2004)
5. Pressman, R.S., Ince, D.: Software engineering. McGraw-Hill (2000)
6. Krippendorff, K.: Content analysis: An introduction to its methodology. Sage Publications, Inc. (2004)
7. Glaser, B., Strauss, A.: The discovery of grounded theory: Strategies for qualitative research. Aldine Transaction (2007)
8. Sommerville, I.: Software Engineering, 9th edn. Addison Wesley (2009)
9. Raymond, E.: The cathedral and the bazaar. Knowledge, Technology & Policy 12, 23–49 (1999)
10. Sheard, S.: The value of Twelve systems engineering roles. In: Proceedings of INCOSE. Citeseer (1996)
11. Sheard, S.: Twelve systems engineering roles. In: Proceedings of INCOSE. Citeseer (1996)
12. ISO/IEC: Amendment to ISO/IEC 12207-2008 - Systems and software engineering Software life cycle processes (2008)
13. Acuna, S.T., Juristo, N., Moreno, A.M., Mon, A.: A Software Process Model Handbook for Incorporating People's Capabilities. Springer (2005)
14. Beck, K.: Extreme programming explained. Addison-Wesley (2000)
15. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile software development methods: Review and Analysis. VTT Publications 478. Technical Research Centre of Finland (2002)
16. Schwaber, K., Beedle, M.: Agile Software Development with SCRUM. Prentice Hall (2002)
17. Palmer, S.R., Felsing, J.M.: A practical guide to feature-driven development. Prentice Hall PTR (2002)
18. Dijkstra, E.W.: On the role of scientific thought. In: Selected Writings on Computing: A Personal Perspective, pp. 60–66. Springer (1982)