Preface

This is not a philosophical, theoretical or motivational book, but a practical one. Its purpose is to enable readers—software developers, managers involved in IT, and educators—to benefit from the good ideas and stay away from the bad ones. This book brings together a number of recent contributions to the domain of Software Engineering (CoSE) for achieving enterprise agility from a range of research groups and practitioners. Agility means the power of moving quickly and easily. These range from tools and techniques for managing discrete, low-level activities developers engage in when developing parts of software systems; knowledge, project and process management for large scale software engineering enterprises; and new ways of organizing software teams including outsourcing, open sourcing, highly distributed virtual teams and global software engineering. I believe that all practitioners engaging in or managing software engineering practices, researchers contributing to advancement in understanding and support for innovative software engineering, and students wishing to gain a deeper appreciation of the underpinning theories, issues and practices within this domain will benefit from most if not all of these contributions.

A large number of submissions have been received in response to our call for papers and invitations for this edited book from many leading research groups and well-known practitioners of leading collaborative software engineering techniques. After a rigorous review process 13 submissions were accepted for this publication. I tried to choose best work among the received lot. This book will enhance the learning about formation, management and evolution in domains such as agile teams, projects with substantive outsourcing, open source software, virtual software teams and global software engineering domains. Being an inter-personal and—often—inter-organizational activity, software engineering introduces a number of social and managerial challenges. Teams may be homogeneous or highly diverse in terms of culture, language and location. This introduces many challenges to supporting collaboration at high levels (process, project management) and low-levels (artefact sharing, consistency). Teams may be comprised of many generalist’s e.g., agile methods or highly specialized individuals or sub-teams whose efforts must be coordinated. An organization needs to ensure appropriate management of teams and between teams. In particular, global software engineering domains introduce very new and challenging problems, such as in contracting and quality control in outsourcing, ownership and “group dynamics” in multi-site projects, and overall project direction and co-ordination in open source software projects.

The intended audience of this book will mainly consist of researchers, research students and practitioners in software engineering and allied fields; it can be in academic or industry. The book is also of interest to researchers and industrial practitioners in areas such as agile software methodologies, lean development, knowledge engineering, artificial intelligence, Legacy system, cloud computing, software project management and component based software engineering.
Preface

The book has a simple structure and is intended for sequential reading. The opening chapter deals with the agile teams and challenges involves in distribution. Agile methods emphasize on team’s collaboration, but how do agile teams collaborate with their geographically distributed counter parts to accomplish requirements related activities. In this chapter collaboration patterns of geographically distributed agile teams are identified in terms of reported communication among team members and their awareness of each other. A multiple case study method is used to study the geographically distributed agile teams in four IT organizations. The study identifies that high awareness among agile teams leads to more communication. Implications for research and software industry are discussed and future research directions are also provided.

The second chapter is a short foray into the style of agile description. This chapter proposes the use of agile methodology in designing the innovation management system in young academic institutes. Since process for technology innovations from most universities and research institutes originate out of chaos. There is a need to associate a clearly articulated method for translation of ideas into technology innovations that will certainly help young academic institutes to inculcate research in students and faculties and would help identify the best commercial application of technology innovations. This chapter presents a practical innovation management process for academic institutes to inculcate research in students and faculties using agile methodologies. Agile methodologies are best suited to adopt in the academic scenario due to rapidly changing environment of academic institutes that can be easily handled using SCRUM methodology.

Chapter 3 is a sketch that presents the design and implementation of a cross-platform architecture with intelligent agents for dynamic business rules, process flows and services composition. This architecture helps in meeting business demands. Things get complicated when enterprise software systems, after a decade of evolution, comprise heterogeneous platforms and different technological stacks. The architecture includes an Enterprise Service Bus for service integration. A Central Intelligent Agent that contains a Prolog-style rule-based engine is designed to execute business rules and processes. The services and components for the business processes are dynamically constructed. The proposed architecture and programming model enables fast prototyping and rapid development in an agile development process across different platforms.

Chapter 4 is about Lean methodology and expansion of Lean manufacturing and lean principles to the management of Service and Information Technology industries. It articulates how waste can be minimal at Software Development Process (SDP) which begins from feasibility study and ends till the product is delivered to the customer. To gear up with the market demand Lean improvement concepts is introduced in (information technology) IT industry. Lean IT is the translation of lean manufacturing practices applicable to the Software Development life cycle (SDLC). This chapter presents how lean methodologies and principles works in service industries to deliver the best quality products. Although Lean concept is traditionally been used in manufacturing industries; but nowadays it is adapted by Services companies with the aim of improving their processes and enhance customer satisfaction.

In chapter 5 role of business process reengineering (BPR) in the modern business world is talked about by illustrating the theoretical and practical concept of BPR, the applications of BPR, the drivers of BRR (in terms of internal drivers and external drivers), the critical success factors of BPR (i.e., egalitarian leadership, collaborative working environment, top management commitment, supportive management, information technology, change management, project management, and cross-functional coordination), the implementation of BPR, and BPR software tools. BPR is a continuum of change initiatives in order to deliver better business performance standards through establishing sustainable process capability in
modern organizations. BPR has become a popular tool to dealing with rapid technological and business change in the global competitive environment. Applying BPR will greatly improve business performance and reach business goals in global business.

Chapter 6 take up the case study of a learning organisation. A learning organisation differs from a teaching organisation, such as a higher education institution, in that each person holds responsibility for their own learning, yet are supported and guided by those who wish to help them further in their personal development. This chapter aims to develop a software project management methodology, based on existing approaches, which can accommodate all people, regardless of ability. The model aims to be suited to those who require simplicity as well as those who want complexity, by integrating best practice from other models, such as PRINCE2, which it can work within. The model developed, called the C2-Tech-S2 approach, is specifically designed for projects that use crowd-funding and agile development, particularly in environments based around the Cloud.

In the seventh chapter high points of agile methodologies over traditional methodologies have been illustrated. Authors discussed about the high failure rates of projects in spite of so much of revolutionary methodologies. Factors like changing requirements and late testing and integration are few of the main causes of this high percentage of failure. Agile development is projected as a way out to the issues linked with traditional software development. Agile development focuses on the rapid delivery of enterprise worth in the form of working software. These methodologies reduces the overall risk linked with software as a result of continuous planning as well as feedback. In this way, agile development team can easily adapt to change requirements, which featured even late in the development process.

Chapter 8 examines about globalisation and its effects on business. Traditional methodologies are not appropriate for the projects where user requirements are not fixed. Agile methodologies have been developed to cope up with user changing requirements and emphasize more on working software & customer collaboration. This chapter mainly focuses on the working methodology of agile development and the usage areas of industry where agile development is implemented. Agile software development is difficult in distributed environment as the team members are at distributed locations. Authors discusses agile industry applicability enablers which are useful for agile software development in distributed environment.

Chapter 9 presents a quantitative security risk management cyber security measure namely the Mean Failure Cost (MFC) to quantify the security of an e-Learning application while taking account of its respective stakeholders, security requirements, architectural components and the complete list of security threats. Moreover, security requirements are considered as appropriate mechanisms for preventing, detecting and recovering security attacks, for this reason an extension of the MFC measure is presented in order to detect the most critical security requirements to support the quantitative decision-making. Chapter focuses on non secure system’s problems and a depth insight interpretation about critical requirements, critical threats and critical components. This extension is beneficial and opens a wide range of possibilities for further economics based analysis. Also this chapter highlights the security measures for controlling e-Learning security problems regarding the most critical security requirements.

Tenth Chapter briefs about the evolution of software and factors involved in organisation agility. In order to be used for longer time period, software needs to evolve. The software evolution can be a result of software maintenance. This chapter discussed 10 versions of GLE (Graphics Layout Engine) and FGS (Flight Gear Simulator) evolved over the period of eight years. An effort is made to find the applicability of Lehman Laws on different releases of two software developed in C++ using Object Oriented metrics. The laws of continuous change, growth and complexity are found applicable according to data collected.
Chapter 11 discusses the problems of coupling and interoperability in local and distributed applications. The former emphasize coupling, since interoperability is simply based on local pointers and on a type system with shared names. The latter had to solve the interoperability problem first and coupling became an afterthought. Chapter discuss both SOA- and REST-based solutions that depends on shared schema data description languages and as a result introduce a higher level of coupling than actually needed. This chapter contends that problems, interoperability and coupling, need to be dealt with in a balanced way. The fundamental problem of resource interaction is precisely satisfying the interoperability requirements of the interacting resources while reducing coupling to the minimum level possible. This way, changeability will be maximized and will significantly contribute to the agility of an enterprise when introducing the necessary changes in its EIS. To achieve these goals, this chapter has presented a multidimensional framework to describe the interoperability and coupling aspects of resources, in an orthogonal fashion, including lifecycle, levels of concreteness, levels of interoperability, and non-functional concerns. Coupling metrics have been defined, which show that these effects reinforce each other in the goal of coupling reduction.

Chapter 12 inspect evolution and maintainability of legacy systems. Discovering the crosscutting concerns and separating it from core functionalities of a software system may help in evolution of the legacy systems. Aspect-oriented software development (AOSD) is new programming paradigm which helps to bring in modularity in the program by writing the crosscutting concerns in the form of ‘aspects’. Modularity brings comprehensibility and hence maintainability of the software system. Tools and techniques, which aid in identifying the crosscutting concerns in such systems and refactoring them into aspects, are needed to apply aspect-oriented techniques to legacy systems at use in industry. This chapter aims to identify issues, problems and approaches used in the migration from legacy systems to aspect-oriented software system.

Chapter 13 scrutinize about the malicious intrusion in cloud based systems. Chapter discuss about the existing network security mechanisms face new challenges in the cloud such as DDOS attacks, virtual machine intrusion attacks and malicious user activities. Although various cloud security measures are prevailing to avoid virtual machine attacks and malicious user activities but these are not foolproof. A new security method is proposed to increase users’ level of trust in clouds. By scrubbing traffic at major Internet points and backbone connection, a defense line is created for mitigation of DDOS attacks. In this chapter Dynamic threshold algorithm based approach is proposed as a proactive approach to detect DDOS attacks for achieving secure cloud environment.

Software engineering has been a very heavily researched area and almost all practicing software teams will need to engage in it. However, many challenges still present both in terms of adopting collaboration practices, processes and tools and improving the state-of-the-art. Many of these challenges are long standing, and hence are fundamental to the act of working together to engineer shared artifacts. These include assembling teams, dividing work, social networking within and between teams, choosing best-practice processes, techniques and supporting tools, and effective project management. Others have arisen due to new organizational practices and technical advances, including open-sourced, out-sourced, multi-site and agile software engineering contexts. We still do not know the ideal way to share knowledge, facilitate the most effective communication, co-ordinate massively distributed work, and design and deploy support tools for these activities.

At the end I would like to convey our appreciation to all contributors including the accepted chapters’ authors, and many other participants who submitted their chapters that cannot be included in the book. In addition, I also appreciate all reviewers for their assistance in formatting the book.