Getting Agile Methods to Work for Cordys Global Software Product Development

Jos van Hillegersberg[1], Gerwin Ligtenberg[2], and Mehmet N. Aydin[3]

[1] University of Twente
School of Management and Governance
P.O. Box 217, 7500 AE Enschede, The Netherlands
j.vanhillegersberg@utwente.nl
[2] Cordys Research & Development, The Netherlands
gligtenberg@cordys.com
[3] Department of Information Technologies,
Işık University, Turkey
mnaydin@isikun.edu.tr

Abstract. Getting agile methods to work in global software development is a potentially rewarding but challenging task. Agile methods are relatively young and still maturing. The application to globally distributed projects is in its early stages. Various guidelines on how to apply and sometimes adapt agile methods have been proposed. However, systematic literature reviews reveal that detailed evaluative studies are scarce and limited to small and medium sized projects. This study presents a framework that integrates best practices of adapting and applying agile methods reported in the literature. The framework is applied to analyze the experiences of global software product development company Cordys in a seven year longitudinal case study. Both the framework and the experiences of Cordys documented in this paper will be of value to other larger projects that aim to be successful in applying agile in globally distributed projects.

Keywords: Global Software Development, Agile Methods, XP, Scrum, Global Teams, Offshoring, Globally Distributed Software Engineering.

1 Introduction

1.1 The Rise of Agile Methods for Systems Development

Agile methods for software development were introduced around the beginning of the new century. The increasing business need for fast creation of internet and mobile applications was a key driver for the introduction of these light weight and nimble development processes. Traditional waterfall based methods had often resulted in large, bureaucratic and slow development processes. In many dramatic cases, sizeable project teams had burned significant amounts of money merely creating documents and reports rather than working software. In the rare event that working software was produced, the customer and end user had often been forgotten in the process, resulting in products that did not meet requirements and expectations. Already in the late 1980s and 1990s, innovations in systems development methods focused on more dynamic,

J. Kotlarsky, L.P. Willcocks, and I. Oshri (Eds.): Global Sourcing 2011, LNBIP 91, pp. 133–152, 2011.
© Springer-Verlag Berlin Heidelberg 2011

adaptive and user centric ways of working. The Agilemanifesto.org, published in 2001, contains twelve principles that synthesize the ideas underlying agile methods. These include welcoming changing requirements, close collaboration of business people and developers, self-organizing teams, a focus on delivering working software and face-to-face conversation as the preferred way to transfer knowledge.

Abrahamsson et al. [2], in their review of agile methods, characterize these as incremental, cooperative, straightforward, and adaptive: "Incremental refers to small software releases, with rapid development cycles. Cooperative refers to a close customer and developer interaction. Straightforward implies that the method itself is easy to learn and to modify and that it is sufficiently documented. Finally, adaptive refers to the ability to make and react to last moment changes". Well known examples of agile methods include Scrum development Process [22],[23]. Extreme Programming (XP) [4] and Feature-Driven Development (FDD) [18]. Scrum and XP are among the most well known agile methods. They are often seen as complementary as XP provides specific engineering techniques and Scrum essentially works as a wrapper for such techniques [8],[10]. While agile methods have increased in popularity, they do not yet provide integral support for systems development. Different agile methods cover different phases of the software development life-cycle. Moreover, Project management support by agile methods is limited and limited emphasis is put on how to operate and tailor agile methods to specific organizations and situations [2],[3].

1.2 Agile Methods in Globally Distributed Development

Whether agile methods and distributed development can be combined to harvest the benefits of both is a subject of much debate. It has been widely recognized that geographical distance and the time zone and cultural differences associated with global distribution have caused problems for globally distributed software teams in achieving successful collaboration (e.g. see [15]). Agile methods use short iterations, frequent builds, and continuous integration that all require very frequent communication, coordination and trust. A theoretical analysis by Turk et al. [25] predicts that several agile process principles, such as self-evaluation, frequent customer and team face-to-face meetings, short time intervals between releases, focus on code rather than documentation and team spirit seem to be impractical in a global setting. The temporal, geographical and socio-cultural distance in distributed development makes the application of agile methods a complex task [16]. Paasivaara and Lassenius [16] summarize the key questions as:

> *In the global software development literature, communication is often seen as the most challenging problem of distribution whereas the agile methods basically rely on communication, preferably face-to-face communication, instead of documentation. The application of agile principles to global software development poses several questions regarding communication: How could daily communication be arranged effectively? What kind of communication practices and media are suitable for supporting different agile practices? How could informal communication, that is important to agile methods, be encouraged? How could the risk for misunderstandings, e.g. regarding requirements, be minimized? How could trust be built and retained between teams to ensure open communication?*

Based on a systematic literature review, Hossain et al. [11] conclude there is no clear description or understanding of how the use of agile practices can reduce global software development (GSD) risks and improve project communication, coordination or collaboration processes. They further state that "current research provides limited evidence of the effective use of agile practices in minimizing risks of global software development processes". Holmström [10] states that "the more common view is that agile methods are not applicable for GSD". Some authors report risks and limitations when using agile methods in a globally distributed setting. Sarker and Sarker [21], based on a case study of developing agility in a distributed IS development setting, find that many of the guidelines or prescriptions for agile methods could not be used effectively (and sustained) in distributed teams. The authors suggest that "agile methods need suitable adaptation for effective adoption/use in a distributed ISD setting". Sakthivel [20] puts forward that agile methods are only suitable for small scale offshore development projects. "Due to their high task dependencies and required face-to-face interaction with users during their iterative analysis, design, and trial stages, they are not suitable for mid- and large-size offshore projects". Hossain et al. [11] concludes based on a recent systematic literature review that: "we do not know the risks of using Scrum in a GSD project with a large number of project personnel, or an increased number of distributed sites". Moreover, the risks when using Scrum in GSD may vary according to project contextual factors.

Ramesh et al. [19] state that combining agile and distributed development introduces five challenges. First, agile development relies more on informal interactions than explicit documentation which poses a real challenge in achieving the communication formality that may be needed to assure effective communication between sites. Second, how can fixing quality requirements to guide the remote team be balanced with the agile principle of ongoing customer developer discussions. Third, what would be the appropriate balance between the formal process-oriented control typically used in distributed development and the more people oriented informal agile practices? Fourth, what is the appropriate level of formality in developing contractual agreements in agile development? Fifth, how can team cohesion be improved given the constraints of a distributed environment? Holmström [10] concludes that what is needed is an increased understanding of the characteristics of agile methods and how these can be applied to reduce the negative influence of distance in GSD.

The experiences of using specific agile methods reported on in the literature are scarce and mixed [26]. For example, based on a case study of a development team located in Sydney and Malaysia, Hossain et al. [12] find that daily Stand-up meetings, with the aid of various communication tools, help to build mutual understanding among distributed project stakeholders. Sprint review meetings increased project visibility and transparency and "Test Driven Development (TDD)" helped to maintain a shared standard development view. Based on case studies of global projects at HP and Intel, Holmström et al. [10] conclude that agile practices are valuable in reducing some of the challenges of globally distributed development. In particular, XP and Scrum practices were found useful for improving communication, coordination, and control within GSD teams. However, Sarker and Sarker [21] find that daily Scrum

meetings are difficult to hold or organize in distributed settings where stakeholders are (often) located in different time zones. Their case analysis further reveals that the use of Scrum sprints (i.e., achieving short deadlines), and the maintenance of daily burn-down charts and stringent monitoring rules can become difficult in a distributed setting, because of a variety of unanticipated complexities associated with local cultures, infrastructures of varying quality, and temporal differences. Sarkar and Sarkar [21] find that the use of agile methods in some cases led to: "dysfunctional stress and work-life balance challenges, with negative impact on morale and productivity".

Paasivaara et al. [17] note that while there is increasing interest in applying agile methods to global projects, there are only a few reported experiences on industrial projects and even fewer case studies. This is confirmed by a recent systematic literature review by Jalali and Wohlin [13] of peer reviewed conference papers or journal articles published between 1999 and 2009. Most of the 77 papers identified have appeared recently and a majority of the current literature is in the form of experience reports, in which practitioners have reported their own experiences on a particular issue and the method used to mitigate it. The majority of them have not documented the characteristics of their empirical study and the context under which the project was running. Only three out of the 77 papers were jointly written by academics and practitioners. Jalali and Wohlin note that although experience reports are useful, more evaluation and validation research is needed to establish foundations for a more mature area. Moreover, they find that existing literature mainly consists of reports on small to medium sized projects.

It is the objective of this study to address the open questions in how to apply agile methods in a globally distributed software development project. As the above discussion reveals, more evaluative studies are needed, especially to learn how agile methods and practices should be implemented successfully in a global project. Current literature is usually lacks longitudinal research. As we have studied the continuous adaptation of agile methods in the global project at Cordys over a longer time period, we can better document lessons learned compared to a single snap-shot case study. Moreover, we focus on a large project where most reported experiences so far were limited to small and medium sized projects..

1.3 Research Method

As many open questions exist in applying agile methods to large globally distributed software development projects, this study adopts a longitudinal case study research approach. Theory building is in its early stages and the evolution of agile methods and the lack of knowledge on how to adopt them in GSD do make a qualitative case study a suitable choice [7].

As we learned that Cordys was among the first companies that adopted agile methods in a large global development project in 2004, we contacted them for conducting a case study research. Cordys showed interest in sharing their experiences in their journey and in collaborating with researchers in sharing knowledge. From the early steps at Cordys with implementing agile methods more than six years ago till today we have been exchanging experiences and ideas. We have conducted interviews with Cordys management and developers in 2005, 2008, 2009 and 2011 to assess the

status and adaptations to the agile processes. In the interview we had in November 2005 at Cordys with Hans de Visser, VP services and Steven ten Napel, VP operations, the initial experiences with agile methods were discussed, including the reasons to adopt agile methods and the implementation strategy. In 2008, a workshop was held at the Cordys headquarters with several developers and project managers to share the experiences and future plans. Annual presentations were given by Gerwin Ligtenberg, program manager at Cordys, to us to discuss the ways the methods were tailored and supported with tools to fit the global setting. In 2011 an additional interview was held with Gerwin Ligtenberg in which he reflected on the key steps taken and changes made to the teams, organization, methods and tools. By serving as a second author of this paper, he also directly contributes to documenting the lessons learned at Cordys.

In order to organize the rich data we collected over the years, a comprehensive framework was needed. Especially over the last few years more studies had become available that provide various ways to analyze the use of agile methods in GSD. However, a comprehensive framework is still lacking. Therefore, we constructed such a framework based on a recent overview of the literature. The framework consists of reported issues that can arise in using agile methods in a global setting and practices that have been used to mitigate these. In the next section we present this framework. Next, we use this framework to analyze the Cordys case study. We report how Cordys, based on their experience, adapted practices, teams, organization processes or tool support to make agile methods work. We contrast this to the strategies listed in the framework. Finally, we summarize key findings and discuss potential further extensions to agile methods for use in GSD.

2 Challenges and Best Practices

2.1 Challenges in Global Agile Development

Hansen and Baggesen [9] describe experiences of using Scrum in a distributed setting between Denmark and Bangladesh. To implement Scrum effectively, it was first introduced to the local teams and later, after experiences was gained, implemented in the global teams. Onsite training sessions in Scrum were held by the CTO. Very experienced senior developers and consultants were moved into global Product Owner roles to establish a more structured flow of tasks and collaboration. “Proxy product owners” were introduced as anchor points for each offshore team. To increase understanding of the product requirements and context with the offshore teams, several domain knowledge sessions in Bangladesh were held. As trust between teams became a problem, onshore and offshore teams were merged into a single team with a single backlog of work. Sprints within sprints were introduced to break down user stories into smaller ones. Daily Scrum meetings and global code reviews created trust. The physical task board was quickly converted into a web-based virtual task board. Efficiency and relationships were further improved by introduction of automated testing and code reviews. Code built in Denmark was inspected by a team mate in Bangladesh and vice versa. To bridge cultural barriers, and build social relationships, people were moved around between both locations.

Ramesh et al [19] based on three company case studies of multiple mid size distributed projects elicit successful practices to slightly adjust and customize agile practices to make them work in a global setting. We summarize their findings here. Rather than starting with informal agile processes from the start, devote the first two or three iterations of a project to finalize critical requirements and develop a high-level architecture. Instead of relying exclusively on informal means for project tracking and monitoring utilizing a product/process repository better supports knowledge sharing. Build well understood functionality first to create an atmosphere in which both the developers and the client representatives were acclimatized to the processes, tools, and

Table 1. Challenges and Reported Practices of Global Agile Development

Challenge	***Practices***
Synchronous communication	S1. Synchronize work hours [19],[11] S2. Make meetings short and effective by posting three daily Scrum questions or develop backlog (feature list) before attending the distributed meetings [11] S3. Constant communication [19] S4. A scrum team allows additional distributed meetings along with Scrum master meeting attended by technical lead or design architect of each local Scrum team [11] S5. Distributed daily Scrum meetings are cut down to twice-a-week meetings [11] S6. XP pair programming can help to increase time overlap and reduce temporal distance [10] S7. Reduce the dependency between sites. Each role in a team at one site has a counterpart on the other site [16].
Collaboration and coordination Difficulties, lack of trust	C1. Informal communication through formal channels [19] C2. Frequent visits by distributed partners. Visits should be long enough to allow for informal chat and building social ties [19],[11],[16] C3. Build cohesive team culture [19] C4. Focus on well understood requirements rather than critical new functionality [19] C5. Document requirements at different levels of formality [19] C6. Maintaining valuable documentation [19] C7. Scrum simple planning can help increase “teamness” and reduce geographical distance [10] C8. XP pair programming and Scrum pre-game phase can help increase mutual understanding and collaboration within and between teams and reduce sociocultural distance [10] C9. Scrum team gathers and performs few initial sprints at one site before distributed development starts [11] C10. Scrum team are gathered quarterly or annually for few days [11] C11. Mandatory demo presentation during retrospective sessions to reduce offshore silence [11] C12. Scrum teams may move from a collocated project to a distributed project gradually through several stages (i.e., evaluation, inception, transition and steady state) [11] C13. A business/software analyst interfaces with the customer on the other site. The proxy customer can make decisions on behalf of the real customer [16]

Table 1. (*continued*)

Communication Bandwidth	B1.	Use "multiple communication modes" to ensure that a Scrum team with distributed project stakeholders is supported with various options of communication tools (phone, web camera, teleconference, video conference, web conference, net meeting, email, shared mailing list, Instant Message (IM), Short Message Service (SMS), and Internet Relay chat) [11],[16]
Large Team	L1.	Autonomous sub teams are allocated work based on features, functions and so on that ensure each sub team is allocated independent architectural subsystems with well defined interfaces project teams are geographically [11],[16]
	L2.	Autonomous Scrum teams are formed locally and each site conducts their own scrum. A Scrum of Scrums is attended by a key touch point member for each team to ensure inter-team communication. Independent architectural subsystems with well defined interfaces are allocated to each team to reduce inter site communication [11]
	L3.	Offshore teams can be geographically isolated and are not cross-functional and may not use Scrum processes [11]
	L4.	A Centrally located management team" in which management persons of each Scrum team are located in a central site [11]
Office Space	O1.	Each Scrum team is allocated to a single room so that they can communicate with each other [11]
	O2.	Each site has a separate meeting room with all necessary network connectivity and tools while attending a distributed meeting [11]
Evolving Quality requirements	Q1.	Trust but verify [19]
	Q2.	Distributed Quality Assurance [19]
	Q3.	Supplement informal communication with documentation [19]
Transformation Change and Learning	T1.	Scrum first introduced to the local teams and later, after experiences was gained, implemented in the global teams [9]
	T2.	Rotating gurus provide initial training and mentoring to the other site [16]
	T3.	Conduct "initial Scrum training," "technical Scrum" to clarify new technology issues, reinforce the value of Scrum and improve team collaboration while using Scrum practices [11]

the application. Instead of short time boxing typical in agile methods, allow for a flexible short-cycle approach in which two to three development cycles are used. Synchronize work hours instead of trying to establish 24x7 schemes. Facilitate informal communication through formal channels to prevent miscommunications. More than custom in co-located agile projects, coordination roles of project managers/leads are important. Constant (meaning almost 24h) communication using mail and videoconferencing is required. Trust is even more important in the absence of formal control. Frequents visits of senior management, customers and product managers to developing sites or the reverse are needed to build trust. Agile working in a distributed fashion requires a cohesive team culture. Trust needs to be supplemented with on-site verification. Quality assurance and supplementing informal communication with formal documents can assure this.

Hossain et al. [11] conducted a systematic literature review to identify challenges of using the prime agile management method Scrum in global software development. They initially identified 366 published papers which they reduced to 20 primary papers. Based on the frequency that a challenge is reported in the literature, they identify and rank seven challenges. In Table 1 we combined the challenges of the studies discussed above and listed practices reported on these papers. Tool support for agile distributed development is a complicated topic that has recently received more attention. Therefore we dedicate a separate section to this topic.

2.2 Tools for Globally Distributed Teams

Globally distributed teams that use agile methods need a variety of tool support. Dullemond et al. [6] discuss the advantages and challenges of the combination of agile software development and GSE with a focus on how these be supported with technological aids. They present five types of technology requirements for tools used in global projects. Abbattista et al. [1] provide an overview of various current and emerging tools for supporting global teams. In Table 2, we integrated both studies to provide an integrated overview of technology requirements and available tool support.

Table 2. Requirements for technology support and sample tools

Requirements [6]	***Description***	***Examples of tools [1]***
Facilitates direct contact between colleagues	F1. Technological support which facilitates direct communication between two or more actors.	Email is the most-widely used and successful collaborative application email can support conversations, but also operate as a task/contact manager. Recently, chat and IM have been spreading more and more in the workplace because, unlike email, they are 'socially translucent', providing a lightweight means to ascertain availability of remote team members and contact them in a timely manner. When rich communication is required audio and video conference may be applied Collaborative development environments (CDE). A CDE provides a project workspace with a standardized toolset to be used by the global software team. Earliest CDE were developed within open source software (OSS) projects because OSS projects, from the beginning, have been composed of dispersed individuals (e.g. SourceForge). Some CDE also include facilities for transparency and continuous integration and build.

Table 2. (*continued*)

Facilitates knowledge sharing among colleagues	F2. Technological support which facilitates the sharing of *technical* project knowledge.	Knowledge center. This function is mostly document-driven and web-enabled, and allows team members to share explicit knowledge across a work unit. A knowledge center includes technical references, standards, frequently asked questions (FAQs) and best practices. Some CDE's include wiki's and blogs to facilitate the creation of a project memory. Researchers have also experimented with social tagging of source code in a collaborative environment
Facilitates transparency of the project status	F3. Technological support which facilitates the sharing of *organizational* project knowledge.	Software Configuration management (SCM). A software configuration management tool includes the ability to manage change in a controlled manner, by checking components in and out of a repository, and the evolution of software products, by storing multiple versions of components, and producing specified versions on command. (example: SVN)
Facilitates quality assurance	F4. Technological support which facilitates quality assurance functions to monitor and guarantee the quality of the product.	Bug and issue tracking. This function is centered on a database, accessible by all team members through a web-based interface. Other than an identifier and a description, a recorded bug includes information about who found it, the steps to reproduce it, who has been assigned on it, which releases the bug exists in and it has been fixed in.
Facilitates continuous integration and frequent builds	F5. Technological support which eases the process of continuously integrating the system as well as producing builds frequently.	Build and release management. It allows projects to create and schedule, typically through a web interface, workflows that execute build scripts, compile binaries, invoke test frameworks, deploy to production systems, and send email notifications to developers. are essential tools to perform continuous integration, an agile development practice that allows developers to integrate daily, thus reducing integration problems

3 Cordys Case Study

The benefits of agile processes to enable a lean and flexible development process sound appealing, especially to a software product company that has to deliver competitive products in a rapidly changing market. Cordys is such a company. Founded in 2001, Cordys initiated a venture to develop from the ground up a Business Collaboration Platform, based on the principles of a Service Oriented Architecture covering an

Enterprise Service Bus (ESB), Business Process Management (BPM) and an Integrated Services Environment [5]. The product would allow existing IT assets to be exposed, embedded, deployed and managed from a business-driven, process perspective. Now, after ten years and thousands of man-years of development, the company's recent release is viewed by analists as a significant player in the BPM market [24].

Cordys employs 560 people globally. Roughly 320 are based in India, 110 in the Netherlands, 50 in Germany, 20 in UK, 30 in China, and 30 in the Americas [24]. Cordys has deployed a globally distributed product development process from the start in 2001. From 2001-2005 Cordys focused entirely on new product development financed by venture capital. Over the last five years, gradually more customers were attracted and implementation partners were added to the Cordys ecosystem. Customers include KPN, the World Bank, AXA financial services and Equens. Partners of Cordys include Accenture, Infosys, Cap Gemini and Atos Origin [24].

In search of a development process that would support its highly innovative environment and would focus on delivering high quality software code in short delivery cycles Cordys started to explore agile methods in 2004. Cordys has been a very early adopter of an agile development process in a globally distributed setting. Over the past seven years they have learned and adapted their agile process and diverged from the known agile practices that were typically designed for a co-located project team. In the following sections we use the challenges and practices reported in the literature (Table 1 and Table 2) to analyze the seven years of experience of using agile methods in globally distributed product development at Cordys.

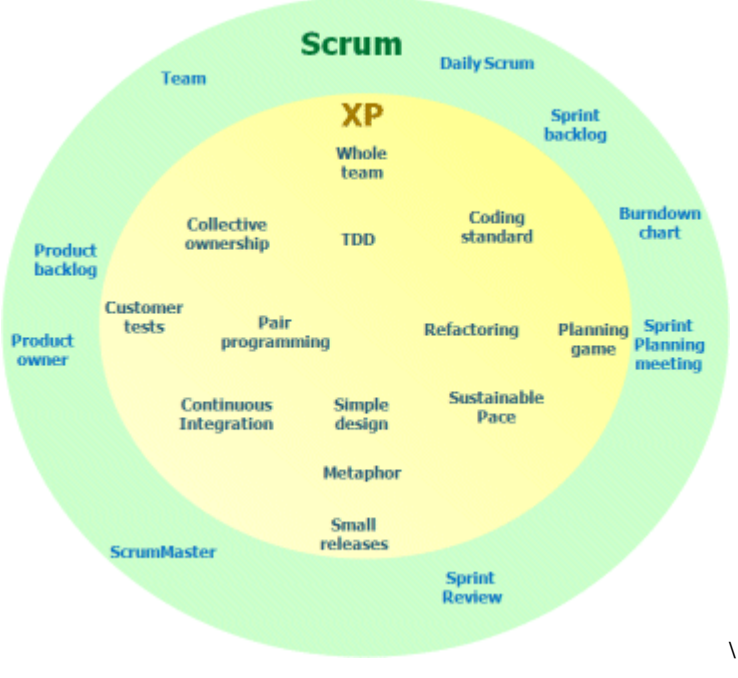

Fig. 1. Combining Scrum and XP: (Source: [14])

3.1 Synchronous Communication

Starting in 2004, Cordys globally implemented Scrum combined with the agile practices of eXtreme Programming (XP) [4]. Cordys quickly rolled out Scrum to its entire development team that at that time had approximately 40 developers in Putten, the Netherlands and 230 in Hyderabad, India. As is demonstrated in Figure 1, Scrum and XP provide complementary agile practices. Where Scrum offers typically a framework for planning and coordinating work, XP provide collaborative practices for design and development work.

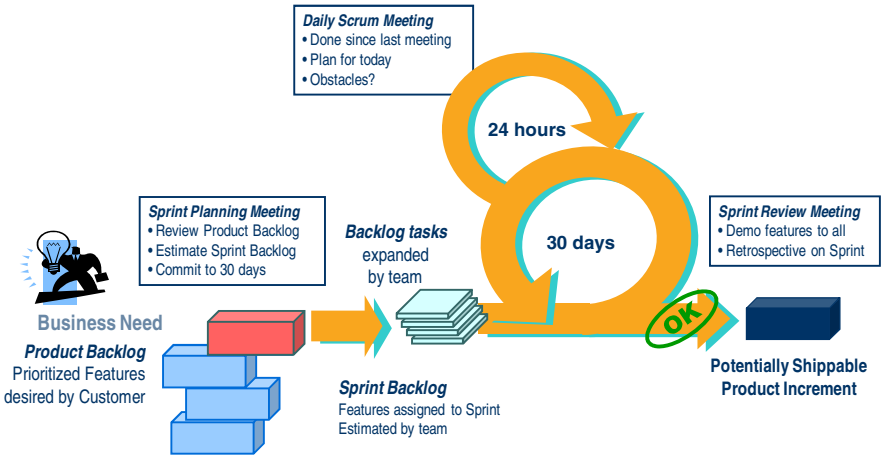

Fig. 2. Scrum process (Source: Cordys)

Cordys adopted a mostly standard Scrum cycle (see Figure 2) that includes agile practices such as daily team meetings and a central product backlog that guides activities. Work towards a new set of functionality is organized in fixed length "Sprints". A sprint starts with planning and ends with a review. A sprint planning meeting is a time-boxed meeting dedicated to developing a detailed plan for a sprint. Project stakeholders attend sprint review meetings to review the state of the business, the market and technology. A daily Scrum meeting is a short daily meeting (usually up to 15 minutes long) in which each team member is expected to address three questions: what did I do yesterday, what will I do today and what impediments are in my way? Three artefacts, namely: product backlogs, sprint backlogs and burn-down charts are produced. Backlogs consist of customer requirements while daily burn down charts show what cumulative work remains. Cordys set the meeting frequency to quarterly business plan reviews and initially 6 week sprints. Later this was reduced to 4 weekly and eventually 2-weekly sprints to get requirements implemented faster and increase productivity. According to the program manager one can not immediately start with short sprints from day one. “You need to build up experience with the process to be able to come up with working demo’s in a short timeframe”. Cordys teams consist of 6-10 persons with various roles like Software Engineers, Product Engineers, Architect(s) and a Program Manager. Management is mainly

concerned with identifying deficiencies or impediments in the development process and practices. Essential to Scrum is that it is goal-driven and gives a high level of personal responsibility and empowerment to developers. For example, developers get the freedom to choose specific software development techniques, methods, and practices.

Synchronizing work hours (S1, **Error! Reference source not found.**) was feasible as Putten and Hyderbad have only a 4½ hour time difference allowing for sufficient overlap to organize joint meetings during regular work hours. XP user stories play a central role in Sprint meetings. A joint repository with user stories helps to create a joint view on the project and its backlog (S2). In an interview in 2008, the program manager noted: “It is important to get the user story ready in time before the sprint starts, as it is delineates the requirements to execute in the sprint and plan the steps appropriately”.

Cordys minimizes required communication by empowering local teams. The team structure and work distribution closely resembles the product architecture. Scrum meetings are mostly held locally on site. However, scrum of scrum meetings still require distributed communications. Also, dependencies between modules do exist. Dependencies that require attention are identified in a sprint meeting and are usually logged in the bug tracker and followed up by frequent email and chat communication between engineers, architects, team leads and release management (similar to S3). When issues gradually escalate conference calls or a video conferencing is organized to resolve the issue. In addition to standard scrum meetings, all new requirements, their user stories and their impact on the component architecture are discussed between architects and team leads in a video conference (similar to S4). Dependencies between requirements and architectural components are discussed in a global sprint meeting. In these meetings work is assigned to teams and to upcoming or future sprints.

3.2 Collaboration and Coordination Difficulties, Lack of Trust

While teams have a certain level of autonomy, the product backlog is centrally monitored (C1). An example of a product backlog chart is shown in Figure 3. Cordys supports the agile principle that Face-to-Face interaction is the most productive method of communicating with customers and among developers. In our interview in 2005 Mr. ten Napel said:"we do believe this is true. We have never attempted to run Scrum meetings using collaborative tools such as videoconferencing. The Scrum master needs to read and even smell emotions of his team members". Fortunately, Cordys' component-oriented architecture enables Cordys to apply Scrum. Ownership of a specific component if is fully given to a single co-located team. The teams pick up tasks from the centrally administrated back-log list that concern their component. As a result, virtually all scrum meetings are co-located. When team members of the remote site do need to be present, they organize a video conference or travel to the site (C2).

As Cordys develops a generic product without a direct customer, the agile principle of direct interaction with the customer is impossible to implement. User requirements are set through bundling and generalizing request from current customers and predicting the needs of new markets. De Visser: "it is essential to have somebody to

play the customer role in scrum team meetings. We usually involve somebody from product marketing to play this role." (similar to C13). Later (from 2008) this role was taken by Product Management. A close collaboration between Product Management and Marketing exists.

Although agile methods direct that developing extensive (relatively complete) and consistent documentation and software models is counter-productive, Cordys does recognize a need for documentation. Ten Napel:"documenting is still important but should not be done only because the method prescribes it. Only if the team sees clear added value to develop diagrams and models, these should be created. We do provide various templates to facilitate this". In addition, Cordys created a configuration management and process support platform using the Cordys product itself. The process can thus be described as a mix of central coordination and local freedom (C5, C6). In an interview in 2008 the program manager stressed: "The requirements need to be articulated till the level that developers can understand them and be sufficiently precise to be a start of the extreme programming process" (C5).

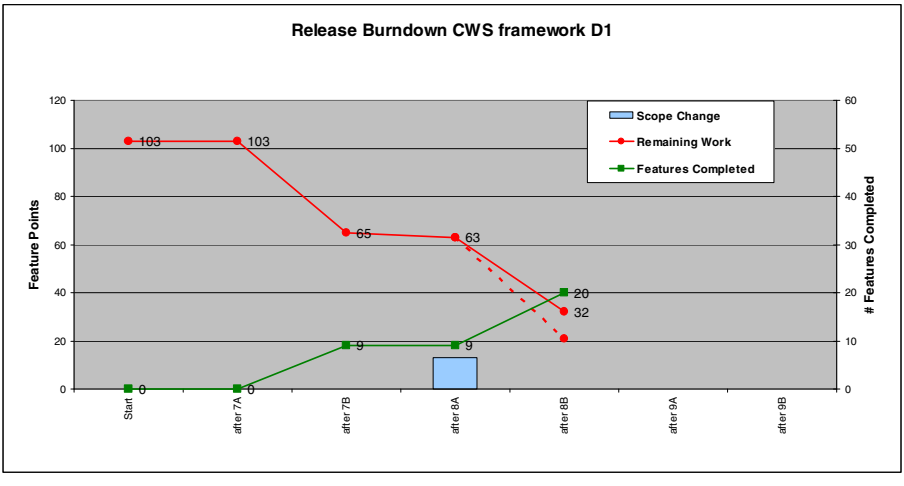

Fig. 3. Example of Cordys' burndown chart (Source: Cordys)

3.3 Communication Bandwidth

Cordys did adopt the practice of "multiple communication modes" to ensure that a Scrum team with distributed project stakeholders is supported with various options of communication tools (phone, web camera, teleconference, video conference, web conference, net meeting, email, shared mailing list, Instant Messages and Internet Chat). The program manager (2011 interview) explains that engineers use chat daily. Complex issues get a functional and business owner and regular video conferences are organized with the teams involved. All designs are maintained on a wiki. As the teams collaborated for multiple years most developers know each other and are

flexible in picking the proper communication tools (B1). To discuss dependencies between teams a Scrum of Scrums meeting was held usually using a telco. “In this evaluation meeting of the last sprint it is important to have rich communication”, the program manager stressed (2011 interview).

3.4 Large Teams

The program manager recalls (2011 interview) that when Scrum and XP were introduced, all the handbooks talked about teams of typically 8, 10 or 12 people. “We had more than a hundred developers, how should we implement agile practices? No guidelines existed for this project scale. So we started to standardize on documenting component interfaces (API’s) as small descriptions that could be demoed. We left the process to arrive in time at such demos to the teams” (this is in line with L1). “One of the most important lessons we learned is that sound component architecture of the product is key and teams need to work dedicated to components in order to make distributed agile work”. “Components that grew too complex were split as team size needed to stay in the range of 8 to 12 people to remain effective”. Over time the Cordys product was spilt into more components: from four large components initially, to more than ninety components today. This process was not easy. An initial attempt to split the product in multiple components failed and the teams had to move back to the original structure. A second attempt took two months but resulted in a successful split up of the product in more independent components. It was a tough decision but management eventually realized that this was the only way the product could be developed further without an enormous overhead in global team to team communications. Today, there are 15 teams that are responsible for several components each. Teams communicate along the lines of API to API communication. They keep each other posted of planned changes in internal API’s. “It is almost like a vendor-customer relationship between teams. The only difference with the external interfaces that our end customers use is that we have to assure backward compatibility for those. Within the product that is not needed”. Components that get more radically new interfaces will continue to offer the deprecated API for some weeks till the consuming teams have migrated to the new component interface.

Offshore teams are allowed freedom as to how they exactly implement scrum meetings and processes (L2, L3). Cordys has learned that enforcing a precise process is not the right way towards effective collaboration. Cordys does not co-locate management in a single site (no L4).

To keep the component architecture simple business requirements are translated to user stories that should map to a small set or preferably a single component.

3.5 Office Space

Project teams share a work room (Figure 4). Separate video conferencing facilities are used for the sprint review meeting. “Only a real collaborative whiteboard remains on our wish list to support joint design and planning meetings. The technology is maturing now” (O1, O2).

Fig. 4. Project room during an architecture meeting (Source: Cordys)

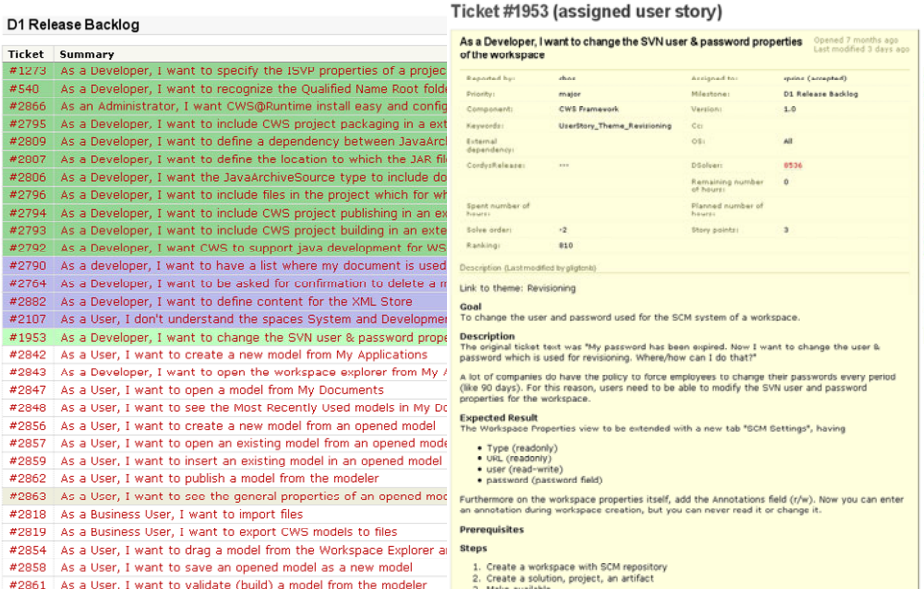

Fig. 5. Example of Cordys user story overview and a user story (Source: Cordys)

3.6 Evolving Quality Requirements

The setting of a product company ensures that trust can be built up over many years of development. Some developers have worked together for decades (also in the former Baan Company). Initial ideas to establish process standards within teams were quickly abandoned. The program manager (2011 interview) recalls why this was not a feasible route: “each time it was different in terms of style and way of working.

Teams work on different parts of the product that consists of different technologies. Each team had their own problems and issues". Management introduced a methodology of empowerment and focused on the quality of the business plan the team produced and the team quality improvement over time". No between teams comparisons were made on quality or productivity.

3.7 Transformation Change and Learning

Cordys experimented in a single team first with the agile approach. In November 2005, Hans de Visser, VP services and Steven ten Napel, VP operations look back on how Scrum and XP were introduced: "Teams are able to and willing to evaluate themselves. When Scrum was introduced in Cordys in 2004, this was not yet common practice". The company was in the process of moving up the CMM ladder. Most developers and architects had a history of working in Baan (the former company of Cordys Founder Jan Baan) that had also heavily invested in achieving high CMM levels through establishing standard methods and processes, measuring quality and predicting progress. De Visser: "adopting Scrum was a shift in mindset. Although we did not throw everything we set the quality focus overboard, the focus changed to innovation and communication…Currently, we manage by setting priorities in the product backlog, the teams themselves decide how many features they will implement in the next Sprint and how". Scrum, became firmly embedded across the organization. Mr. Ten Napel summarizes key steps in achieving this: "You should unconditionally believe in the process. Although it is tempting, never implement an agile process informally. For example, Sprint timelines should really be fixed; Scrum teams should not become too large, etc. Do not start with an insignificant pilot but immediately apply the process in a critical project (not fully in line with C4). Also, we had an enthusiastic Scrum champion; we trained Scrum masters, and established a program office that has supported the full transition. Scrum training was organized in

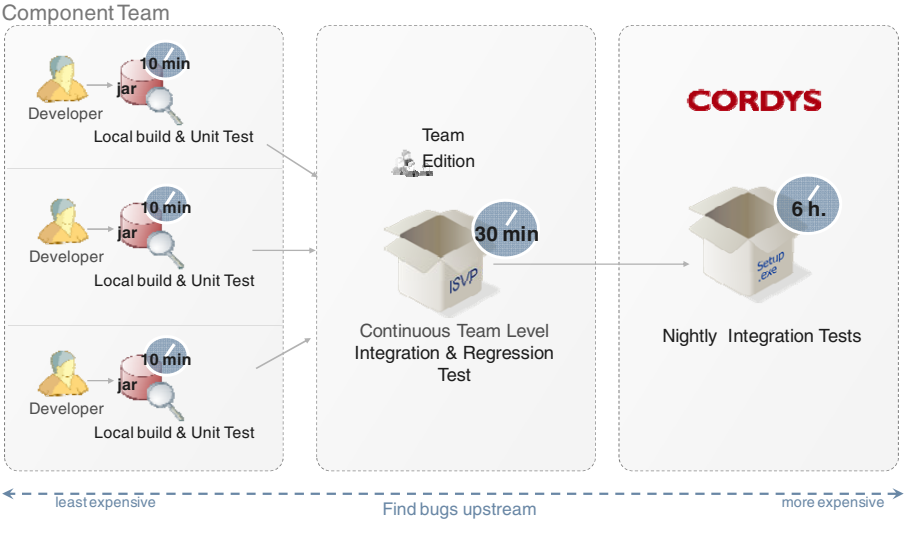

Fig. 6. Continuous local builds and team level integration and testing (Source: Cordys)

Hyderabad and Putten for architects and team leads. Last but certainly not least, from the start Jan Baan, then our CEO, viewed the Scrum process to fit our core goal of being innovative" (T3). In 2011, the Program manager looks back with confidence that initially starting with Scrum in one local team was a good decision (T1). “Still initially, many team managers thought this was yet another management fad that would pass by, they did talk about Sprints and Scrums but hadn’t actually changed much in their processes. It took time to get accustomed to the new way of working”.

Active Builds > Regression Test Results > **Test Report Summary**
Server: **srv-ind-vbeb** (Windows 2003/IIS/MSSQL/OLEDB)
Build: **std1 #234**

Component	TestCase Count						Success %	
Name	Total	Executed	Timeouts	Failures	Errors	Success	Executed	Total
Installer	7	7	0	0	0	7	100.00	100.00
Binary Parser	23	23	0	0	0	23	100.00	100.00
Scheduler	228	228	0	0	0	228	100.00	100.00
XDS	260	260	0	0	0	260	100.00	100.00
CCF	378	347	0	0	0	347	100.00	91.80
Data Mapper	283	248	0	0	0	248	100.00	87.63
ESB	1012	851	0	1	0	850	99.88	83.99
XML	1673	1279	0	1	0	1278	99.92	76.39
Coboc	212	141	0	0	0	141	100.00	66.51
Notification	229	154	0	14	0	140	90.91	61.14
WS-AppServer	343	204	0	4	8	192	94.12	55.98
Rule Engine	163	92	0	0	1	91	98.91	55.83
DBConnectors	5772	2671	0	0	4	2667	99.85	46.21
TestFramework	72	7	0	0	0	7	100.00	9.72
BPM Runtime	277	0	0	0	0	0	N/A	0.00
BPM Modeling	226	0	0	0	0	0	N/A	0.00
XForms	172	0	0	0	0	0	N/A	0.00
State Machine	5	0	0	0	0	0	N/A	0.00
SecurityServices	24	0	0	0	0	0	N/A	0.00
Classic Studio	0	0	0	0	0	0	N/A	N/A
SSO	0	0	0	0	0	0	N/A	N/A
CWS	0	0	0	0	0	0	N/A	N/A
Undefined	0	0	0	0	0	0	N/A	N/A
ALL	**11391**	**6512**	**0**	**20**	**13**	**6479**	**99.49**	**56.88**

Fig. 7. Report of automated regression tests (Source: Cordys)

3.8 Tools for Global Software Teams

Over the years Cordys invested much in tools to support the agile way of working. Initially very limited and simple tools were used but as the product increased in size various tools were introduced. In order to gain more productivity, each step, feature or functionality is tested. The program manager (2011 interview) notes that “in order to make the approach work we established a central code repository. We found that each team needed to have a code branch that they could individually test and in addition we needed to make sure that every one or two weeks a commit was made to our configuration management tool SVN so that integration tests could be run (F3, F5 in Table 2). We further established continuous integration tests for team branches and the entire product. It took us half a year to make sure that new code could be added to the trunk (the main body of development, originating from the start of the project until the present) and from this automatically several builds could be generated and tested (Figure 6)(F4). Each team has to comply to the rules so that when they add to the trunk the build is not allowed to break and install and the functionality should pass the tests. “The only way to get to shorter sprints is to have automated testing”. Bugzilla was used to report bugs and assign them to teams (F4). A single installation allowed management to keep an overview of bug statuses. The Auto Pilot tool is used to create

daily builds, to execute all the static and dynamic tests automatically, to monitor the build quality, and to compare the quality with previous builds. The project manager also knows the test coverage; in which line that developer or programmer found difficulty and how much time they tested it. When evaluating the tools used by Cordys against those we listed in Table 2, than we can conclude that most categories have been implemented. Only a true collaborative development environment is not yet in place (CDE). This can be explained as teams work on various components using different technologies and a fully integrated CDE would be of limited added value.

4 Conclusion and Discussion

Getting agile methods to work in global software development is a potentially rewarding but challenging task. Agile methods are relatively young and still maturing. The application to globally distributed projects is in its infancy. The debate whether agile methods are suitable for such projects is ongoing. The literature provides mixed results and recommendations as to how and under what conditions agile methods can bring added value. Various guidelines on how to apply and sometimes adapt agile methods have been proposed. However, systematic literature reviews reveal that detailed evaluative studies are scarce and limited to small and medium sized projects.
This study contributes to both theory and practice. The theoretical contribution includes a framework that integrates best practices of adapting and applying agile methods to GSD reported in the literature. It integrates various experience reports and systematic reviews on practices and tools. The framework is applied to analyze the experiences of Cordys in a longitudinal case study. To our knowledge, no other studies exist on large projects using longitudinal data. The experiences from Cordys documented in this paper will be of value to other larger projects that aim to be successful in applying agile in GSD.

Several key lessons have been learned at Cordys in seven years of applying agile methods in a global setting. Introducing agile is hard in a big bang fashion. A gradual approach though firmly supported by management works better. It is essential not to try to standardize the details of agile methods across global teams. Agile is all about trust, empowerment and some freedom should be allowed to local teams to tailor their process and documentation style. A mix of central coordination and local freedom and empowerment works best. Although the deadline of a sprint should be firm, one should not be too dogmatic about the length of sprints. It takes time to move to short sprints especially as many teams are jointly working on a complex product. Co-located scrum meetings are far more effective than distributed scrums. Teams should ideally share an office space. To achieve as much local scrums as possible, the product architecture should be component based and local teams should remain small and work only on a limited set of components. Communication between teams should follow the lines of the product architecture. Multiple communication modes should be adopted by the project and distributed scrum meetings absolutely require high quality tool support. The role of tools is vital in constructing a complex product globally. Getting configuration management, bug reporting, automatic integration and testing in place is critical to success. It is a significant investment but worth every effort in the

longer run. Cordys did not find all Scrum/XP practices equally useful. Collective code ownership could cause confusion and mixed responsibilities and was not adopted. Pair programming was found too expensive and inefficient and is not practiced at Cordys.

We can conclude that the framework compiled in this study (Table 1, Table 2) is effective for analyzing the use of agile methods in large global software development projects. The findings from the Cordys case reveal that the practices included in the framework are sometimes directly applied, adopted over time, adapted to the context or not applied at all. Remarkably, some of the practices were not found effective at Cordys and alternative practices were invented. To understand how agile practices should be introduced and adapted to fit the project context and stage of maturity we plan to study more global projects that are in the process of adopting, tuning and adapting agile methods. We also aim to continue to follow the further experiences and adaptations of agile methods at Cordys.

References

1. Abbattista, F., Calefato, F., Gendarmi, D., Lanubile, F.: Incorporating social software into distributed agile development environments. In: 1st International Workshop on Automated Engineering of Autonomous and Runtime Evolving Systems, and ASE 2008 the 23rd IEEE/ACM Int. Conf. Automated Software Engineering, pp. 46–51 (2008)
2. Abrahamsson, P., Warstab, J., Siponenb, M.T., Ronkainena, J.: New Directions on Agile Methods: A Comparative Analysis. In: Proceedings of the 25th ÍEEE International Conference on Software Engineering (2003)
3. Aydin, M.N., Harmsen, F., Slooten van, K., Stegwee, R.: On the Adaptation of An Agile Information Systems Development Method. Journal of Database Management, Special issue on Agile Analysis, Design, and Implementation 16(4) (November-December 2005)
4. Beck, K.: Embracing Change With Extreme Programming. IEEE Computer 32, 70–77 (1999)
5. Cordys Website, `http://www.cordys.com/`
6. Dullemond, K., Van Gameren, B., Van Solingen, R.: How technological support can enable advantages of agile software development in a GSE setting. In: Proceedings 4th IEEE International Conference on Global Software Engineering, ICGSE, pp. 143–152 (2009)
7. Eisenhardt, K.M.: Building theories from case study research. The Academy of Management review 14, 532–550 (1989)
8. Fitzgerald, B., Hartnett, G., Conboy, K.: Customising agile methods to software practices at intel shannon. European Journal of Information Systems 15(2), 200–213 (2006)
9. Hansen, M.T., Baggesen, H.: From CMMI and isolation to scrum, agile, lean and collaboration. In: Proceedings of the Agile Conference, AGILE, pp. 283–288 (2009)
10. Holmström, H., Fitzgerald, B., Ågerfalk, P.J., Conchúir, E.Ó.: Agile practices reduce distance in gloral software development. Information Systems Management 23(3), 7–18 (2006)
11. Hossain, E., Ali Babar, M., Paik, H.: Using scrum in global software development: A systematic literature review. In: Proceedings - 4th IEEE International Conference on Global Software Engineering, ICGSE, pp. 175–184 (2009)

12. Hossain, E., Babar, M.A., Paik, H., Verner, J.: Risk identification and mitigation processes for using scrum in global software development. In: Proceedings - Asia-Pacific Software Engineering Conference, APSEC, pp. 457–464 (2009)
13. Jalali, S., Wohlin, C.: Agile practices in global software engineering - A systematic map. In: Proceedings - 5th International Conference on Global Software Engineering, ICGSE, pp. 45–54 (2010)
14. Kniberg, H.: Blog on Combining Scrum and XP, http://blog.crisp.se/henrikkniberg/2007/10/13/1192249140000.html
15. Kotlarsky, J., Oshri, I., Van Hillegersberg, J., Kumar, K.: Globally distributed component-based software development: An exploratory study of knowledge management and work division. Journal of Information Technology 22(2), 161–173 (2007)
16. Paasivaara, M., Lassenius, C.: Could global software development benefit from agile methods? In: Proceedings - IEEE International Conference on Global Software Engineering, ICGSE 2006, pp. 109–113 (2006)
17. Paasivaara, M., Durasiewicz, S., Lassenius, C.: Using scrum in distributed agile development: A multiple case study. In: Proceedings - 2009 4th IEEE International Conference on Global Software Engineering, ICGSE, pp. 195–204 (2009)
18. Palmer, S.R., Felsing, J.M.: A Practical Guide to Feature-Driven Development. Prentice-Hall, Englewood Cliffs (2002)
19. Ramesh, B., et al.: Can distributed software development be agile? Comm.of the ACM 49(10), 41–46 (2006)
20. Sakthivel, S.: Managing risk in offshore systems development. Communications of the ACM 50(4), 69–75 (2007)
21. Sarker, S., Sarker, S.: Exploring agility in distributed information systems development teams: An interpretive study in an offshoring context. Information Systems Research 20(3), 440–461 (2009)
22. Schwaber, K.: Scrum Development Process. presented at OOPSLA 1995 Workshop on Business Object Design and Implementation (1995)
23. Schwaber, K., Beedle, M.: Agile Software Development With Scrum. Prentice-Hall, Englewood Cliffs (2001)
24. Thompson, M.: Technology audit Business Operations Platform 4, Cordys, Butler Group (June 2009)
25. Turk, D., France, R., Rumpe, B.: Assumptions Under lying Agile Software Development Processes. Journal of Database Management (JDM) 16(4) (October-December 2005)
26. Woodward, E., Surdek, S., Ganis, M.: A Practical Guide to Distributed Scrum, Deployment and Advanced Configuration. IBM Press (2010)