

Agility in a Large-Scale System Engineering Project: A Case-Study of an Advanced Communication System Project

Amir Shatil

Dept. of Communications Systems Software
RAFAEL – Advanced Defense Systems
Haifa, Israel
ashatil@rafael.co.il

Orit Hazzan

Dept. of Education in Technology & Science
Technion – Israel Institute of Technology
Haifa, Israel
oritha@technion.ac.il

Yael Dubinsky

Computer Science Department
Technion – IIT
Haifa, Israel
yael@cs.technion.ac.il

Abstract—In this paper we describe the application of the agile software development approach in system engineering projects. We describe the main characteristics of system projects, highlighting the challenges that the application of agility in such cases raises. Such challenges emerge due to the unique characteristics of system projects, which include the multidisciplinary nature of such projects, which encompasses, in addition to the software teams, also hardware, firmware, algorithms, and mechanics teams; the significant importance of including the project management in the process; the multiple kinds of integration tasks; the need for high-level and skilled human resources; and finally, the actual software development process, which in many system projects is embedded real-time software. Though agile development is an accepted development methodology in software projects, many questions are still unanswered when agility is applied to system projects. This paper aims to partially close this gap by addressing the application of agility in an advanced communication system project.

Keywords—Process implementation and change; Software Engineering Process; System integration and implementation

I. INTRODUCTION

Agile software development is now accepted as a mainstream paradigm in software development. At the same time, however, its fitness for other kinds of projects is still questioned.

In this paper we examine the application of the agile approach in system engineering projects (hereinafter referred to as "system projects"), which, in addition to a software component, encompasses other components and disciplines, such as firmware, hardware, algorithms, mechanics, and more. In addition, in many cases, system projects include complex integrations, dependencies among the different disciplines, and heavy overheads, such as complex setups and tight inter-discipline coordination. Further, since the software component in such projects is embedded real-time software, which relies on specific hardware/firmware, software development in intermediate development stages depends on other project components (e.g., the hardware). Furthermore, the project development process may take, in some cases, several years, during which the customer's requirements change frequently and are clarified only together with the engineers as the development process

proceeds. These characteristics lead to additional difficulties of system projects that are related to time estimation and meeting deadlines. To overcome these difficulties, all factors involved in the process should be synchronized both on the planning level and on the actual implementation level. Needless to say that such complex project environments require highly skilled and experienced human resources.

One relevant question to be asked now is: How should a system project be managed? Is the agile approach, which has already been proved successful for software projects, appropriate for system projects as well? This paper explores the answer to the last question. It describes and analyzes a case study in which the agile approach is applied by the software team of a system project with a significant software component. The project is developed by RAFAEL, an Israeli firm that specializes in the development and production of multi-disciplinary systems. The case study focuses on the initial stages of the application process of the agile approach, which started in 2008 with one software team of one advanced communication system project, namely the development of an airborne voice and data radio network. The paper outlines challenges inherent in the application of agile software development in the project, challenges that are not usually considered in pure software projects, and suggests possible solutions for overcoming these challenges.

The first author of this paper is the senior software development leader of this project and a member of the system team. The second and third authors are external consultants with 8 years of experience with the agile methods, who guide the introduction and application of the agile approach in the said project.

The rest of the paper is organized as follows. Section II describes the paper background, the nature of the agile approach and the questions that arise when an attempt is made to apply it to system projects. Section III describes the case study mentioned above, including the project environment prior to the introduction of agile development and its current status with respect to the adoption of agility, the challenges of such an assimilation process, and possible ways to overcome these challenges based on preliminary lessons learned with respect to the application of the agile approach in system projects. In Section IV we conclude. All topics addressed in the paper are examined within the multi-faceted HOT – human, organizational and technological – analysis framework introduced by Hazzan and Dubinsky [8].

II. BACKGROUND: AGILITY IN SYSTEM PROJECTS

Over the past decade, as it became clear that the typical problems of software projects are global and are not limited to specific kinds of software projects, software practitioners around the world started looking for alternative development approaches and began adopting development methods to help them overcome the difficulties they face in a systematic manner. Agile software development is one such approach and it is applied by one of several methods, such as Extreme Programming [1] and Scrum [13]. Agile software development has become increasingly accepted in the software industry and is currently considered one of the mainstream approaches in the software industry.

Agile software development offers a professional approach to software development that encompasses human, organizational, and technological aspects of software development processes [8]. The agile manifesto (<http://agilemanifesto.org>) outlines the following principles:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile software development methods aim to develop high- quality software both from a technical perspective (meaning less bugs) and from the customer perspective (better meeting customer needs). In practice, agile software development methods are based on short iteration development, a rigid and tight testing process, and on meeting customer requirements. The word agility refers to the application of the agile approach including its practices, spirit, and inspiration.

Data indicate that agile software projects cope successfully with common problems of software projects. For example, according to the State of Agile Development survey¹ conducted by VersionOne and the Agile Alliance in 2007, 60% of respondents estimated a 25% or greater improvement in time-to-market; 55% of respondents reported a 25% or greater improvement in productivity; and 55% of respondents reported a 25% or greater reduction in software defects.

Though agile software development is well accepted nowadays for software projects, the application of agile software development has been challenged in system projects in which software is only one component. This is the focus of this paper.

A system is a set of interrelated components of persons, organizations, procedures, software, and equipment that has a specific goal [14]. System engineering combines different disciplines for the development of such systems; software development is one of these disciplines. Software development, as well as other disciplines, has its own techniques and tools to support the work, and the joint work of all disciplines is managed by the mechanism of system engineering. This complex setting introduces coordination

difficulties that encompass technological, managerial, and human-related aspects. In this paper, we examine whether the agile approach to software development can be adapted to system projects, and if yes, how.

We propose, and illustrate in this paper, that such an application is possible by making specific adjustments to the application of agility for system projects. This argument is based, among other things, on the realization that agility is also suitable for non-software projects. Indeed, with respect to system engineering, this assertion makes further sense: after all, in both software and system projects, the same central factors, such as customer requirements and testing, play a central role. Since the agile approach has already proven itself with respect to software engineering, it is only reasonable to try to apply it in system projects, which include a software component. Indeed, agility has already been examined in system projects [5,6,7,10,11,15]. In this paper we describe our experience.

We end this section by noting that the agile approach has already been investigated in different areas such as management and education. The following descriptions of agility reflect its nature as a general management approach that can be applied in different areas. The case described in this paper belongs to this approach, which extends the application of the agile approach beyond software projects.

An agile process is both light and sufficient. The lightness is a means of staying maneuverable. The sufficiency is a matter of staying in the game [3 (p. 178)].

Agility is dynamic, context-specific, aggressively change-embracing, and growth oriented. [...] It is about succeeding and about winning: about succeeding in emerging competitive arenas, and about winning profits, market share, and customers in the very center of the competitive storms many companies now fear[3, (p. XXII)].

III. A CASE STUDY: AGILITY IN AN ADVANCED COMMUNICATION SYSTEM PROJECT

A. The Project Environment

This paper focuses on the application of agility in complex, one-of-a-kind projects, tailored for specific customer needs. Such projects are in fact systems, and sometimes even systems of systems, which means that besides software, other different and varied disciplines, such as hardware, firmware, algorithm, and mechanics, participate in the project development. Such projects have relatively long development periods, taking several years and sometimes even several generations.

Most projects developed by RAFAEL, the company that examines the suitability of agility for its system projects, the focus of this paper, are developed using the waterfall model or a version of it with some iterative characteristics, including the fixed-price and (in most cases) fixed-schedule budget. In such projects, an integrated project team performs the initial stages of requirement analysis and high-level design, prior to the full-scale development definition. Following the concept definition stage, these projects proceed through the traditional, long, linear development

¹ Agile Development: Results Delivered:
http://www.versionone.net/pdf/AgileDevelopment_ResultsDelivered.pdf

stages, simulations, experiments, and end with the system's deployment at the customer's site.

This was also the development process in the specific project reported on in this paper prior to the application of agility by the project's software team. Due to several major problems, such as frequent requirement changes, expensive overheads (mainly due to inter-disciplinary coordination and integrations), and lack of communication within the team, this project was chosen in order to examine the application of agile methods in system projects in RAFAEL, in an attempt to improve product quality and process efficiency. The project leadership decided to start this examination with the software team, which consists of 15 people, organized in four sub-teams. Most team members are skilled, embedded real-time software engineers, specializing in networking and communication; some were group leaders/architects.

Agility has been adopted in two stages.²

In the first stage, the following agile practices were applied within a Scrum framework: Sprints³ of 8 weeks each, daily standing meetings in sub-teams, maintaining a sprint backlog, a sprint review/summery (presentation), and a sprint retrospective.

This stage was a big step forward, towards improvement in many aspects: planning, communication and interactions, and customer (system engineer) collaboration. However, it seemed that a more substantial and courageous act is needed to be taken in order to make the best out of this methodology. Furthermore, the sprint reviews did not include any measurement so there was no objective data regarding the process achievements (and so there was no way to measure the improvement, nor any ability to improve the process systematically).

In the second stage, which started five months later, the process was tightened up. Specifically, the software team adopted the following agile practices:

Short iterations: Two-week iterations were set. At the end of each iteration, a business day [8] is dedicated to the following activities: live demos on the target systems of several selected features developed in the previous iteration (takes place in the development lab); summary of team accomplishments in the past iteration; measure presentation and analysis; feedback from the customer, project manager and software team leader; one-hour reflection; and planning of the next iteration. One team member, the process engineer, owns the responsibility for preparing the business day, including meeting with the customer to discuss his priorities for the next iteration, obtaining the required data from the tracker, and analyzing the tasks allocated for the next iteration and their time estimation, together with the relevant practitioners.

Customer involvement: Instead of the real customer, the main system engineer acts as a proxy customer. His involvement includes setting iteration goals and task priorities, participating in the business day, and participating

in the daily integration/technical meetings, whose aim is to coordinate on a daily basis the activities of the different project disciplines, focusing on multi-disciplinary activities, mainly integrations. This daily coordination is required since, as mentioned before, the other project disciplines are not working according to the agile approach at this stage.

Measurement: At this stage, the following measures, defined by specific metrics, are collected on a daily and iteration basis: non-technical meeting hours, unplanned activities hours, burndown chart, burnup chart, planning vs. accomplishment, and number of open defects.

Roles: Role assignment is not a simple task in system projects, partially because the organizational structure of firms specializing in system projects is usually a matrix organizational structure (see below). Accordingly, at this stage, the main roles that are active in the software team are:

Tracker: in charge of maintaining the daily sprint backlog (according to team members' reports) and the automatic generation of all required measures.

Process engineer: in charge of the iteration planning, which includes collecting available team capacity (working hours), meeting with the customer to discuss his vision and priorities for the iteration, decomposing these tasks into small chunks in the Iteration Backlog (together with the relevant team members, including estimating work efforts and assigning team members), balancing work effort loads, allocating hardware resources, examining dependencies among disciplines, and finally, meeting the customer to approve the backlog and, if necessary, adding/removing tasks ("fine tuning").

Architect (not fully adopted yet): in charge of the system and software architecture (mainly algorithmic modules). This includes high- and low-level design, coding, unit testing, refactoring, and accountability for software delivery and quality.

In addition, the assimilation process of the agile approach includes an on-going activity that aims to define those roles that are tailored to the specific project setting. The artifact-centric approach [2] is used to identify significant development artifacts, for each of which artifact lifecycle is then defined, including the decision making authorities of the relevant roles at meaningful milestones of the lifecycle. For example, with respect to the iteration-backlog artifact, two role holders are defined: one role holder is responsible for preparing the backlog and one role holder is in charge of approving it.

Stand-up meetings: Stand-up meetings take place every morning with all the software team members, the customer and sometimes the project manager. We highlight that the software team includes now the 15 practitioners who were originally organized in four sub-teams, as well as two integrators and a tester.

Reflections: On each business day, which takes place at the end of each two-week iteration, the external consultants facilitated a one-hour reflection session. The aim of the reflection is not only to let the team members improve their understanding of the development process and to find ways to improve the process, but also to help the team members understand the complexity of the development process and

² Prior the decision to apply the agile approach was made, a regular, long-iteration approach was considered according to which the project would be divided into half-year iterations, each consists of a 'mini waterfall' process.

³ In this paper, the terms Sprint and Iteration are interchangeable.

their critical role in the adoption of the agile approach on the project level.

In the team described here, the agile process has been applied for about eight months. It is clear, however, that the current application of agility it is not sufficient for system projects, and that it should be broadened and deepened, in order to fully benefit from the potential advantages offered by its application. As we illustrate in Section III.B, the process involved many challenges. In Section III.C we present preliminary solutions to overcome these challenges.

B. Challenges in Adopting Agility in the Project

In this section we outline the main challenges that emerged during the adopting process of the agile approach in the system project reported on in this paper.

Tailor the agile process to a beyond-software project: This challenge calls for the spirit of agility to be expressed by a methodology that addresses also the non-software components of the projects as well as the connections and dependencies between the different components. How should such a methodology be expressed and defined? What agile ideas – values, practices, roles, etc. – should be defined by such a methodology?

Tailor the agile process to a project with embedded real-time software: This challenge addresses the kind of software developed in many system projects, namely embedded real-time software. The issue has been addressed in the past by the agile community. For example, Cordeiro et al. [4] suggested strong unit testing as the foundation of the proposed methodology for ensuring timeliness and correctness. Due to the previous challenge, however, the actual application of agility to this kind of software should be formulated carefully, especially in the hardware ramp-up stages, when the software development process is not pure development, but rather is about supporting hardware, board initialization, and stabilization.

Recruiting the project management's support and making it part of the agile process: This challenge obviously does not emerge in cases in which the project management initiates the adoption of agility into the system project. In other cases, for example, when the initiative to work according to the agile approach is suggested by the software team, or even by the software team together with the non-software disciplines involved in the project, it is crucial to recruit the project management to the process, even though the project could be managed without the management's direct involvement. Without the management's involvement and support, however, the outcome of the agile process will be less effective by far, since the lack of customer collaboration will force the teams to maintain two methodologies: one for the team itself and one for the management. This scenario also has a strong negative mental influence on the team members.

Nevertheless, it is important to emphasize that even when management support is obtained, its actual integration into the agile process must be examined so as to fully benefit from its contribution to the process. Such management involvement is especially important in matrix organizations (see below), which is the organizational structure of most firms that develop systems.

Working with other disciplines: As mentioned before, in addition to the software component, a system project encompasses other disciplines as well, such as hardware, algorithms, and mechanics. In most cases, each discipline works according to its own development method and culture. Thus, in addition to merely investigating whether agility is appropriate for these disciplines, the definition of the interrelation between the different disciplines should be examined in the agile spirit as well as the implementation of the agile approach when teamwork of team members from different disciplines is required. This challenge is further echoed when the practitioners from the different disciplines have varied expertise domains.

One can argue that it is sufficient that the software team adopts the agile approach, and that if the other disciplines do not adopt it, the software team should simply minimize the influence of such a situation. This, however, raises several problems.

First, when the software team works according to the agile approach, it keeps measures on a regular basis; thus, it is the only team whose progress can be analyzed, scrutinized and improved. Due to the mere existence of data, and without any bad intentions, it naturally becomes the only team that can be criticized. Furthermore, since the software team is the only team that measures its development process, the information that can be extracted from the measure analysis cannot be fully utilized. This may even lead in some cases to erroneous and negative conclusions regarding the performances of the software team, while in reality, it is working very well, and the data in fact reveal that problems exist in the other disciplines or in the interface between the different disciplines.

Second, the fact that the different disciplines work according to different methodologies makes it harder to synchronize the work between them. Due to strong dependencies between the different disciplines in system projects, such a situation leads in many cases to expensive overheads and long idle periods.

Third, a significant (and complex) part of a system project consists of long periods of hardware setup, which increases the dependency of the software team on the hardware team (and vice versa). If the different disciplines do not work according to the same process, or even if they are not synchronized, this further amplifies the overhead problems and the long idle periods mentioned above.

Fourth, in many system projects, multidisciplinary teamwork is required for integration tasks and other kinds of problems. In such teams (or task force), each discipline has a representative, and a task leader is assigned from one of the disciplines. Needless to say that such teamwork requires coordination among the team members. If agility is applied only by the software team, the actual coordination may be difficult due to the different characteristics of the development methods. A radical organizational change would include teams with team members from more than one discipline, who all work according to the agile approach.

Skilled and experienced human resources: Human resources are important in any project; the same implies for system projects, in which highly skilled practitioners with a

variety of expertise domains are required. Not only must they understand and be expert in their own discipline, they should also have the ability to see the connection and interfaces between the different parts of the project. Furthermore, they should have the ability to think on different levels of abstraction, both on the detailed level and on a higher level of abstraction, and to move between these abstraction levels wisely, when needed. This ability is especially important in integration tasks, when a clear and systematic analysis of complex problems is required in order to locate the problematic component of the system and then activate the right people to coordinate a fine-tuned analysis and find a solution.

In addition, when the process is driven by the agile approach, the question of knowledge management should be addressed since in many system projects, almost each team member has a specific expertise domain with a long learning curve. For example, in the project described here, expertise domains include board support packages, e.g. drivers, for specific kinds of processors, networking, and algorithms. Since agility fosters knowledge sharing among team members, the challenge is to manage knowledge wisely and effectively, without increasing the overhead imposed in such cases of knowledge gaps.

Matrix organizational structure: A matrix-structured organization contains teams of people from several sections of the business. These teams are created for the purposes of a specific project, often existing only for the duration of the project, and are led by a project manager.⁴ The matrix organization is an attempt to combine the advantages of the pure functional structure and the product organizational structure. This form is perfectly suited for “project-driven” companies⁵ like the one in which the project described in this paper is developed.

Though there are many advantages to the matrix organizational structure, if communication is a vital value, as it is in agile software development, the matrix structure may clash with this value simply because people are often located in different physical parts of the organization.

Other challenges that the matrix organizational structure faces when implementing the agile approach in a system project, address the team structure and definition, as is explained in what follows.

First, if tasks are accomplished by people from different disciplines, how is the team defined? Is the team defined by the disciplines, that is, do all members of a specific discipline form a team? Or is the team composed of all people working on the project, from all disciplines? Is the team defined only for the software practitioners? Should designated task forces be formed for specific integration tasks?

Second, since each agile method defines a set of roles to be accomplished by the team members, how are roles defined when the agile approach is implemented in the matrix organization? Should new roles be defined? The answers to these questions are even more difficult when an

engineer is working simultaneously at more than one project, as happens frequently in many projects developed in matrix organizations.

Multiple integrations: As mentioned earlier, system projects include many integration tasks of different kinds and levels (such as, board ramp-up, diagnostics, interfaces integration, functional integration, system integration, etc.). Since agile development delves into the details of time allocation and planning, it seems to be perfectly suited to the following aspects, which are at the heart of system projects in general, and of the multiple integrations carried out in such projects in particular:

Risk management: By handling the biggest risks first and planning them carefully like any other sprint/iteration task, the project can reduce its risks in its early stages.

Integrations: By detailed planning of each integration, this vague activity is decomposed into small, fine-grained and specific tasks, including dependencies between the different disciplines and needed hardware setups.

Testability: This is a crucial factor in any software project, and is even more crucial in large-scale, multi-disciplinary system projects, in which the system must be tested as a whole, including its hardware, firmware, and even its environmental conditions (such as temperature). The idea of automatic builds and tests (which is part of continuous integration) is one of the agile basics, but is not trivial when it comes to embedded real-time software systems. Different solutions can, however, be found to accomplish full, distributed automatic target system tests, e.g., commercial off-the-shelf tools such as Rhapsody and vectorCAST, and homemade tools.

Hardware resources/setup: This resource usually causes a bottle neck, since it has many configurations, its setup time is not immediate, and in many cases it can be carried out only by a hardware team member. By planning and defining fine-tuned tasks, it is possible to maximize the use of these resources, to identify the need for each specific setup, and to coordinate its use with the appropriate disciplines (hardware, firmware).

Non-agile culture: Connections between software development methods and cultural issues have been discussed previously [12,16]. Highsmith [9], for example, says that “a particular culture is not necessarily change tolerant or change resistant—but it may resist certain types of changes and embrace others. ... Many methodology failures are caused by a problem definition followed by a solution design, with little analysis of whether or not the solution design fits the company or the project team's culture.”

As mentioned above, in its current development process, the company reported here uses the waterfall model or a version of it with some iterative characteristics; naturally, this implies a specific culture. However, it is well known that the introduction of agility into organizations requires a change in the organizational culture, which, as indicated above, is not a trivial process. Such a cultural change changes the attitudes towards many issues, such as time management, work habits, and knowledge and information maintenance, management and sharing. When such a change

⁴ Source: <http://www.learnmanagement2.com/matrix%-20structure.htm>

⁵ Source: <http://www.visitask.com/matrix-organization.asp>

involves several disciplines, as in the case of system projects, it may be even more complicated.

One significant change is expressed by the transparency that the agile process introduces, which requires significant courage from all project stake holders. This, in itself, is not a trivial matter to achieve in any organizational culture change, but is especially important due to the fact that the agile approach exposes many problems and defects in the project, and reflects a clear, either positive or negative, picture of the project status. It is important to emphasize, though, that the agile approach does not create these problems; it just exposes them. All parties involved, and especially the project management, should well be aware of this fact.

The different approaches towards the new culture introduced by the agile approach are expressed even with respect to meeting schedules, which are part of the time management approach introduced by agility: How many meetings should take place during the day? How long should they be? How should it be decided if a meeting is even necessary?

Other cultural issues potentially intensified by the agile approach in system projects, and which have been mentioned before, are mutual planning, integration teams, and role definition (especially of leadership roles).

Measurement: The monitoring of any project requires measures. Agile development further highlights the importance of such measures. This does not change in system projects in general, nor in system projects in which the agile approach is applied in particular. Specifically, the following questions should be addressed: What should be measured? How? How should the gathered data be analyzed? How may the interrelations and dependencies between the different disciplines be expressed and how do they affect the measurement process?

For additional, specific measure-related challenges, see the paragraph on "Working with other disciplines" presented above.

Customer: The customer plays a central role in agile teams, which is manifested mainly on the Business Day [8], when he or she gives the team feedback, explains his or her vision for the next iteration, and answers questions raised by the practitioners. The customer plays a central role also during the iteration, when he or she participates in the stand-up meetings, answers the software engineers' questions, and works with the specific role holder on the planning of the next iteration.

In the system project examined here, the real customer is external to the process and the role of on-site customer is played by the project system engineer; that is, the team communicates with a proxy customer. There are two main reasons for this: First, the real customer is physically located in a different city in Israel (about 100 km from the development site). Second, the customer does not wish to visit the site as frequently as suggested by the agile approach, nor does he have the resources to do so.

Such indirect contact with the customer raises the typical problems of communication gaps. How they can be solved? How can the agile approach be implemented if the customer does not speak to the team directly?

Technological issues: In what follows, we mention several technical issues that should be addressed in any system project. As it turns out, the mere introduction of agility into the project intensifies the need to discuss and address these issues.

Definition of a completed task: The nature of each task in a system project may range between a "simple", pure software module to a system-level integration process. The definition of a completed task (i.e. one that is "done") also varies accordingly, and is sometimes hard to define. Furthermore, in many cases, the completion of a task may depend on other disciplines. It is emphasized that incomplete tasks may have a great impact on a project's cost (hours and resources), on team moral and on the system's quality.

Bugs: Many bugs in system projects are system-level problems and require the coordination of two or more disciplines, both in the analysis stage and in the fix-and-test stage. Complex bugs may require a task force, with a leader and a specific work plan.

Artifact mapping: A process is required for mapping key artifacts such as design, integrations, testing, sprint planning, requirement changes, etc., in order to specify the steps and actions they are composed of and the roles and responsibilities of each team member in their accomplishment.

C. Possible Solutions to Overcome the Challenges of Applying Agility in the Project

In what follows, we describe several agile-inspired solutions to the aforementioned challenges. We categorize the solutions into two groups: organizational solutions and methodological solutions. The organizational group of solutions addresses the human infrastructure, relating to culture, attitudes, beliefs, etc; the methodological group of solutions addresses the technical infrastructure, relating to the actual implementation of the agile approach.

Organizational solutions

This group of solutions addresses mainly the culture change that the agile approach introduces. In order to foster the adoption of the agile approach, all parties involved must become familiar with it. As is evident, most of the suggestions in this group address this approach.

Recruiting the project management to the process and teaching it the agile approach: As mentioned in the challenges section, management involvement in the project in general, and in the adoption process of the agile approach in particular, is crucial. Accordingly, one of the first things to be done, after recognizing the desire to incorporate the agile approach into the project, is to let the management understand the power, as well as the risks, of this adoption process. Data, of course, is the best way to do that; in a system project, however, it may be not sufficient due to the unique characteristics of such projects, as well as the matrix organizational structure in many project-based companies. In addition, accumulated experience with the application of agility in system projects is very rare.

Accordingly, management should be recruited to the project, for example, by explaining the theoretical background, both on the practical level and on the conceptual

and emotional level, and by exposing them to a de-facto agile team.

In the case described in this paper, the project management realized the potential benefits of the agile approach at a relatively early stage. For example, although at the stage of the agility adoption process described here, the agile approach has been adopted only by the software team, the project management, which realized the benefits of applying the agile approach by the software team, uses the software team's bi-weekly planning to manage the entire project. In general, since in software engineering projects, everything is related, dependent, and connected to everything else, the release backlog and the iteration planning process force the management to think, plan, and manage the project in an agile manner, a fact that directly influences all project teams and activities.

Recruiting the software team members to the process and teaching them the agile approach: As with the project management, the software team members must also be led to realize that they are on track and that they benefit in some way. This benefit can be expressed by improving their professional skills, meeting deadlines, and receiving positive feedback from the management and customer, which reflects their appreciation.

In addition, we must not forget that introducing agile development may raise some concerns that should not be neglected. Therefore, team members should be given the opportunity to express these concerns. In the said project this is done in several ways:

First, the external consultants held personal conversations with each team member. The software department management promised team members that the opinions they express in these personal conversations would remain anonymous, as indeed they were.

Second, the bi-weekly reflections offer another opportunity for the software team members to express their opinions and concerns with respect to the adoption process of the agile approach. For example, in one of the reflection sessions, team members were asked to share one positive aspect and one negative aspect of the agile approach as they have experienced it so far. They also were asked to suggest one aspect that can be improved. As a result of this reflection, a mapping process was initiated of each team member's expertise domains, with the aim of improving knowledge management and sharing. With the intention of having more than one person become familiar with each component of the software, and based on the mapping process mentioned above, team members began taking responsibility for development tasks in areas they were not familiar with. Thus, the concern about distributing knowledge and concentrating specific knowledge in the hands of only a few people was addressed. Additional concerns that were raised and addressed in the reflection sessions included the context switch (between different work tasks) phenomenon and several problems related to task definitions and time estimations.

Recruiting the other disciplinary teams to the process and teaching them the agile approach: In order to derive maximum benefit from the agile approach for the sake of the

project's success, it was realized that the other disciplinary teams should also adopt the agile approach or at least be synchronized to some degree with the software team. One way to achieve this is, as mentioned, by having the management view the project through an agile lens.

In addition, to help the other group leaders grasp the essence of agility, the software team leader gave the other group leaders a lecture on agile development, followed by many informal and frequent conversations on the subject.

Methodological solutions

Since it is clear that the implementation of the agile approach in a system project differs from its implementation in a software project, adjustments should be made to the agile methods for software projects, which address the unique characteristics of the system project. For example, while the adoption of agile practices, such as the daily stand-up meeting and the sprint backlog, is quiet straight forward, some fine tuning is nevertheless required, such as in the composition of the participants in the daily meeting (which currently includes, as is described above, the software team members and the system engineers in the role of the customer), backlog task granularity, resource allocations, etc.

This tailoring is also performed using an agile process. This means that at each step, the actual implementation of the agile process is examined with respect to its fitness to the project, and specific adjustments are made accordingly. In other words, based on data obtained from the project, the unique characteristics of the firm, the team, and the project are identified, investigated, and analyzed in order to improve the adjustment of the agile approach to the project. In what follows, we describe the implementation of this approach to two topics – the Business Day and measures.

The Business Day: The business day consists of the last sprint summary, reflection, and planning of the next sprint. After a gradual process, the final agenda and schedule for the business day was set, including live lab demos (on the real target system), a presentation of the accomplishments of the previous iteration (tasks' status and measures), and customer feedback (on the last iteration) and vision (for the next iteration). However, the agile principle stating that the current product at the end of each iteration should work as a whole cannot be fulfilled in a two-week iteration process of a system project. Thus, instead of presenting the customer with all features of the product work completed so far, only several specific features that were completed in the last iteration are presented in the lab, on the real target system.

Measurement: In order to monitor the project successfully, measures should be relevant to the project as a whole and not only to its software component. Accordingly, measures should be defined and taken in a way that takes into consideration the process complexity and dependencies between the different disciplines.

IV. CONCLUSION

In this paper we presented a case study of the introduction of agility into a system engineering project. Based on an examination of the assimilation process after its initial stages, in which the agile approach was adopted only by the software team of the said project, we outlined the

challenges that arose in the adoption process, and made some preliminary suggestions for overcoming these challenges.

In general, the adoption process was addressed from the HOT – human, organizational and technical – perspectives [8]. From the human perspective, we addressed topics such as teamwork in multi-disciplinary teams and human resources; from the organizational perspective we discussed topics such as culture, organizational structure and management; from the technical perspective we explored topics such as measures, testing and the multiple integrations.

REFERENCES

- [1] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison Wesley, 2005.
- [2] M. Cantor, and J. Sanders, “Operational IT governance”, IBM developerWorks, May 15, 2007.
- [3] A. Cockburn, *Agile software development*. Addison-Wesley, Reading, MA, 2002.
- [4] L. Cordeiro, C. Mar, E. Valentin, F. Cruz, D. Patrick, R. Barreto, and V. Lucena, “An agile development methodology applied to embedded control software under stringent hardware constraints”, *ACM SIGSOFT Software Engineering Notes* 33(1), 2008, pp. 1-10.
- [5] B. Greene, “Agile Methods Applied to Embedded Firmware Development”, In *Proceedings of the Agile Development Conference (ADC’04)*, 2004.
- [6] A. Hari, and D.H. Cropley, “Agile System Engineering for Creative Anti-Terror Solutions”, *Proceedings of the Systems Engineering , Test & Evaluation Conference, SETE 2005 - A Decade of Growth and Beyond*, Brisbane, Queensland.
- [7] A. Hari, and D.H. Cropley, “Agile Systems Engineering for Creative Anti - Terror Solutions”, TAE Report No. 974, http://ae-www.technion.ac.il/library/tae_view.php?id=21, 2007.
- [8] O. Hazzan, and Y. Dubinsky, *Agile Software Engineering*, Springer, 2008.
- [9] J. Highsmith, *Agile Software developments Ecosystems*. Addison-Wesley, 2002.
- [10] P. Manhart, and K. Schneider, “Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report”, *Proceedings of the 26 th International Conference on Software Engineering (ICSE’04)*, 2004, pp. 378-386, pub: IEEE Computer Society
- [11] J. Ronkainen, and P. Abrahamsson, “Software Development under Stringent Hardware Constraints: Do Agile Methods Have a Chance?” *XP*, 2003, pp. 73-79.
- [12] S. Sawyer, and P.J. Guinan, “Software development: Processes and performance”, *IBM Systems Journal* 37(4), 1998, <http://www.research.ibm.com/journal/sj/374/sawyer.html>.
- [13] K. Schwaber, and M. Beedle *Agile Software Development with SCRUM*, Pearson Technology Group, 2002.
- [14] R. Shishko, R. G. Chamberlain et al. *NASA Systems Engineering Handbook*. NASA Center for AeroSpace Information, 1995.
- [15] N. Van Schoonderwoert, and R. Morsicato, “Taming the Embedded Tiger – Agile Test Techniques for Embedded Software”, *Agile Development Conference (ADC’04)*, 2004.
- [16] E. Yourdon, *Death March: The complete software developer's guide to surviving "mission impossible" projects*. N.J.: Prentice Hall PTR, 1997.