Tool Support for Team Building in Global Software Development

Muhammad Ali Babar and Tan Phong Phan
Abstract

The project aimed at developing a tool support for Accelerated Team Building for Global Software Development (ATGSD). The tool purports to support team building activities in Global Software Development (GSD) as well as enhancing awareness among team members. GSD has emerged as one of the most popular paradigms for developing software intensive systems. It is characterized by geographical distribution of software development teams across multiple sites, cultures and time zones. GSD offers a number of benefits, including extending working hours, taking advantage of low cost destinations, closer to the market, shortened time-to-market cycles and rapid response to customer needs. Additionally, it also helps to trigger new ideas [1] since developers originate from diverse backgrounds. However, GSD team members also suffer from several kinds of coordination and communication challenges. Moreover, there are other kinds of potential challenges such as different time zones, lack of awareness. Lack of awareness information makes it difficult to initiate contact and often leads to misunderstanding of communicating content and motivation. Hence, a deep understanding of distributed team members’ awareness about which project members are currently available, information about group members, their status and their roles as well as their interests can play an important role in addressing some of the communication challenges. To address some aspects of awareness in distributed teams, TEO and its collaborators have proposed an ATGSD methodology for supporting distributed team building and managing as well as enhancing the awareness among team members by providing information of project’s mission statement, best practices, connectivity map, sharing team members information via a support of team poster artifact. In order to identify and to understand awareness support and studying how the ATGSD methodology is applied in practice as well as team members’ behaviors during the ATGSD phase meeting we conducted a qualitative research methodology in this project. The research data was collected through three interviews with the developers of the ATGSD methodology and two observations of the different phases of implementing ATGSD with the project team. The qualitative data were then analyzed carefully for identifying the requirements for the ATGSD tool. Moreover, we have also conducted a systematic literature review on cloud-based system architectures and a review of the multi-tenant characteristics of Software as a Service (SaaS) for gaining insights into current problems and corresponding proposed solutions. The understanding of cloud-based architectural knowledge was then applied to design a cloud-based architecture for the ATGSD tool. With cloud-enabled architecture applied, the ATGSD tool shall be able to provide a consolidated access point, which will help to reduce operation as well as maintenance cost. Hence, organizations may have access to the ATGSD tool whenever required during the development process without requirements for a separate installation in each individual organization. This research project has also performed a preliminary evaluation of the developed tool using a comparative case study evaluation study for assessing usability and efficiency of how the ATGSD tool assists practitioners in team forming session.

Keywords: Global Software Development, GSD, Cloud Computing, Service Oriented Architecture, AT4GSD, Team Building
# Table of Contents

1 Introduction ................................................................................................................................. 1

2 Background and motivation ........................................................................................................ 4
  2.1 Global software development (GSD) ..................................................................................... 4
  2.2 Accelerated Team Building for Global Software Development Teams (ATGSD) ............... 5
  2.3 Cloud computing and multi-tenancy ...................................................................................... 6
    2.3.1 Multi-tenancy at application level .................................................................................... 7
    2.3.2 Multi-tenancy at database level ....................................................................................... 7

3 Research methodology ................................................................................................................. 9
  3.1 Research objectives ................................................................................................................. 9
  3.2 Ethnographically inspired approach ...................................................................................... 10
  3.3 Organizational context .......................................................................................................... 10
  3.4 Data collection and analysis ................................................................................................. 11
  3.5 Proof of concept development .............................................................................................. 11
  3.6 Technology evaluation .......................................................................................................... 12

4 Requirement specification ............................................................................................................ 13
  4.1 Findings related to the development of ATGSD methodology ............................................ 13
  4.2 Requirement specifications for ATGSD tool support ............................................................ 16
    4.2.1 Functional requirements .................................................................................................. 16
    4.2.2 Non-functional requirements ......................................................................................... 19
  4.3 Use case model ..................................................................................................................... 19
    4.3.1 Use cases for a normal user ............................................................................................ 20
    4.3.2 Use cases for an administrative user ............................................................................. 20

5 Architectural overview ................................................................................................................. 22
  5.1 Domain model ....................................................................................................................... 22
  5.2 Architectural model .............................................................................................................. 23
  5.3 Architectural factors .............................................................................................................. 24
    5.3.1 Maintenance .................................................................................................................... 24
    5.3.2 Scalability ....................................................................................................................... 25
    5.3.3 Data and functionality visibility ....................................................................................... 25
    5.3.4 Persistence Management ................................................................................................. 26
    5.3.5 Isolation and sharing among tenants .............................................................................. 27
    5.3.6 Integration with external web services and platform ...................................................... 27
  5.4 Design specification .............................................................................................................. 28
    5.4.1 Overall package diagram ............................................................................................... 28
    5.4.2 GUI layer ....................................................................................................................... 29
    5.4.3 Web service layer ......................................................................................................... 30
    5.4.4 Integration with external services and platform ............................................................ 35
    5.4.5 Data access layer ........................................................................................................... 36
    5.4.6 Interactions among layers .............................................................................................. 36
    5.4.7 Deployment overview .................................................................................................. 38

6 Implementation overview ............................................................................................................. 40
  6.1 Technologies and supportive frameworks ............................................................................. 40
    6.1.1 ASP.NET ....................................................................................................................... 40
    6.1.2 Windows communication foundation (WCF) ............................................................... 40
    6.1.3 ADO.NET entity framework ......................................................................................... 40
  6.2 Data model ............................................................................................................................ 40
  6.3 Graphical user interfaces ........................................................................................................ 42
    6.3.1 Graphical user interfaces for normal users ................................................................. 42
    6.3.2 User interfaces for administrative users ...................................................................... 48
7 Tool evaluation .................................................................................................................. 50
  7.1 Objectives and methodology ....................................................................................... 50
  7.2 Feedbacks .................................................................................................................... 50

8 Conclusion and future work ............................................................................................ 53

9 References ....................................................................................................................... 56

List of figures

FIGURE 1: USE CASES FOR A NORMAL USER ................................................................. 20
FIGURE 2: USE CASES FOR AN ADMINISTRATIVE USER .............................................. 21
FIGURE 3: DOMAIN MODEL ............................................................................................... 22
FIGURE 5: CALLING THE AUTHENTICATION SERVICE FROM PTAAS ......................... 28
FIGURE 6: PACKAGE DIAGRAM ......................................................................................... 29
FIGURE 7: PRESENTATION LAYER ................................................................................... 29
FIGURE 8: WEB SERVICE LAYER ....................................................................................... 30
FIGURE 9: PROJECT WEB SERVICE DATA CONTRACT ................................................. 31
FIGURE 10: WEB SERVICE CLIENT .................................................................................... 32
FIGURE 11: WEB SERVICES ............................................................................................... 33
FIGURE 12: PROJECT WEB SERVICE CLASS DIAGRAM .............................................. 33
FIGURE 13: XML REPRESENTATION OF TENANT INFORMATION ................................. 35
FIGURE 14: LOG-IN ACTIVITY DIAGRAM & ACCESSING EXTERNAL WEB SERVICE ......... 35
FIGURE 15: DATA ACCESS LAYER .................................................................................... 36
FIGURE 16: SEQUENCE DIAGRAM - RETRIEVE AND DISPLAY A LIST OF PROJECTS ...... 37
FIGURE 17: SERVICE CONTRACT OF PROJECT PROVIDER AND CONSUMER ............... 37
FIGURE 18: SEQUENCE DIAGRAM- RETRIEVE A LIST OF PROJECTS FROM DATA ACCESS LAYER ............................................................................................................. 38
FIGURE 19: DEPLOYMENT VIEW ...................................................................................... 39
FIGURE 20: DATA MODEL ................................................................................................. 40
FIGURE 29 - LOG IN USER INTERFACE .......................................................................... 42
FIGURE 30 - OVERVIEW PAGE ....................................................................................... 43
FIGURE 31 - TEAM POSTER ............................................................................................. 44
FIGURE 32: PROJECT'S MISSION STATEMENT AND BEST PRACTICES ......................... 45
FIGURE 33: DIFFERENT TIME ZONES BETWEEN COLLABORATIVE LOCATIONS ....... 46
FIGURE 34: AVAILABLE TEAM MEMBERS IN ISLAMABAD ............................................ 46
FIGURE 35: AVAILABLE TEAM MEMBERS IN DENMARK ............................................. 47
FIGURE 36: CONNECTIVITY MAP SHOWING THE COMMUNICATION STATUS BETWEEN DISTRIBUTED TEAM MEMBERS ......................................................................................... 47
FIGURE 37: CONNECTIVITY STATUS VIEW FOR AN ADMINISTRATIVE USER .......... 48
FIGURE 38: ADMINISTRATION PANEL ........................................................................... 48
FIGURE 39 - CREATING A NEW PARTICIPANT AND ASSIGNING A ROLE ..................... 49
1 Introduction

Global Software Development (GSD) has emerged as one of the most popular paradigms for developing software intensive systems. It is characterized as geographical distribution of software development teams across multiple sites, cultures and time zones [2]. Nowadays many organizations have been adopting and practicing GSD. It offers a number of benefits, including extending working hours, taking advantage of low cost destinations, closer to market, shortened time-to-market cycles and rapid response to customer needs. Additionally, it also helps to trigger new ideas [1] since developers originate from diverse backgrounds. However, due to the characteristics of GSD including temporal, geographical, cultural and linguistic distances, there are several kinds of communication and coordination challenges that may not be encountered by teams developing software in a more traditional way [3]. Furthermore, GSD team members also suffer from several kinds of potential challenges in particular different time zones and lack of awareness. Different time zones can have adverse effects on communication and collaboration since the higher the number of time zones crossed, the less the time people are at work at the same time (e.g., overlapping time) [4]. Whereas, lack of awareness information makes it difficult to initiate contact and often leads to misunderstanding of communication content and motivation [1] because in GSD environments team members may not know each other personally [5].

Many research, studies have been attempting not only identifying GSD problems but also providing the solutions to overcome these challenges. As reported in [6] an increasing number and variety of technologies are being developed to support GSD for addressing various challenges related to communication, collaboration and coordination in global settings. However, there has been little research on awareness in GSD teams. In order to address some aspects of awareness in a globally dispersed environment, TEO has proposed a methodology named Accelerated Team Building for GSD teams 1(ATGSD). ATGSD is defined as a specialized team building methodology improving the social skills of a GSD team, addressing the known distance in GSD namely spatial distance, temporal distance, cultural distance and the software engineering environment. It attempts to compensate for the lack of social relations and team cohesion in a distributed software development team environment [7]. One of the most important artifacts in ATGSD methodology is the team poster, which contains all of the relevant personal as well as professional information about team members for example participants of a team, their profile pictures, their names, their contact information, interests, professional, their status, their roles as well as expectation for the collaborative work, etc. This awareness information is critical for members who work in a globally dispersed environment, which helps them to initiate the communication easier. As a mutual support, communication is considered as one of the best ways of creating and maintaining awareness about different technical and social aspects in software development team [3] and inadequate communication can lead to lack of awareness [4]. That can be a problematic for GSD teams.

1 http://at4gsd.wordpress.com
As the current situation indicates there is a need for a tool to support ATGSD practices. With the aims to support team building activities and enhancing awareness among team members in the globally dispersed environment, the tool shall be widely accessible across many organizations around the world. Moreover, it is also required to serve many different users concurrently, which drives us to consider applying cloud-enabled architecture for the tool.

As reported in [8] and [9] cloud computing has the potential to address challenges associated with GSD. Cloud computing has emerged as the new distributed software paradigm. It provides an alternative solution by letting the users allocate resources on demand. The cloud service model comes in three forms: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Software as a service is a software application service. It is defined as online software delivery service that allows customers to subscribe to a paid application service instead of paying for a software license and permits for sharing the service with multiple customers. Multi-tenancy is the primary characteristic of SaaS. It allows SaaS vendors to run a single application, which supports multiple tenants on the same hardware and software infrastructure. It is the fundamental design approach, which improves the manageability of SaaS applications [10]. With the capacity/possibility provided by multi-tenancy, the global distributed team can access to a consolidated application, which helps reducing development efforts as well as maintenance cost for the application. In addition, it also provides for alignment across different teams since they use the same application, and achieve more or less the same set of services. When incorporating cloud computing with GSD, it provides some potential benefits including enhancing communication and collaboration and rapid solutions to technical issues [8][9]. Furthermore, when the application is provisioned as a service, it helps to reduce the licensing cost, provide easy access to expensive technology and sensitive hardware and enhance awareness and collaboration [9].

The abovementioned discussion motivated us to set up the goal for the project, which is defined as follow: the primary aim of the project is to design and develop a tool for supporting practices of the ATGSD methodology (the ATGSD tool). This includes facilitating project managers to construct and manage distributed teams more efficiently as well as enhancing awareness among team members. Moreover, the ATGSD tool is designed as a cloud-based solution in order to achieve all of the abovementioned benefits provided by cloud computing and the multi-tenant characteristics of the SaaS service model.

In order to support this goal, several observations and interviews have been made to understand participants’ behavior, team culture as well as how ATGSD is applied in practice. This understanding helps to figure out which activities need support as well as which corresponding features need to be implemented for the tool. Hence, the information gathered during the observations and the interviews are used as a resource for gathering the potential requirements for developing the ATGSD tool. Moreover, researching on GSD and cloud computing are equally important since the ATGSD tool is intended for supporting team building activities in GSD as well as enhancing awareness among distributed team members. Furthermore, it is also potentially deployed on a cloud infrastructure. Therefore, the outcome was used as an inspiration for devising a suitable and optimal solution for the ATGSD tool. When all
requirements for the ATGSD tool were ready, the next step was to design, develop and evaluate the tool.

The organization of the thesis is structured as follows: chapter 2 discusses related work and a review of recent contributions in cloud computing and multi-tenant SaaS architecture. In chapter 3, we outline the research method. The research method includes a combination of ethnographic observations and interviews. Moreover, we also conducted a systematic literature review on cloud-based architecture systems and multi-tenant SaaS architectures. Chapter 4 outlines the requirement specifications of the ATGSD tool. In chapter 5, we present the architectural overview of the ATGSD tool including system architecture based on the architectural design decisions and trade-offs. Chapter 6 describes the implementation details. It also discusses the technologies and frameworks applied in the ATGSD tool in order to carry out the requirements as defined in chapter 4. In chapter 7, we present an evaluation of the ATGSD tool in collaboration with the distributed teams practicing ATGSD. Chapter 8 provides the conclusion of the thesis and discusses the limitations as well as possible improvements in future works.
2 Background and motivation

In this chapter, we provide an overview of the background literature. This will form a basis for designing and developing a cloud-enabled tool for supporting the ATGSD methodology. In particular we focus on the multi-tenant aspect of cloud-enabled systems, which is one of the potential characteristics of cloud-enabled architectures that we consider to incorporate in the ATGSD tool.

2.1 Global software development (GSD)

GSD offers a number of benefits, including extending working hours, taking advantage of low cost destinations, closer to market, shortened time-to-market cycles and rapid response to customer needs. Additionally, it also helps to trigger new ideas [1] since developers originate from diverse backgrounds. However, for adopting GSD successfully organizations must address typical challenges due to the characteristics of GSD including communication, coordination and collaboration. Additionally, there are other kinds of potential challenges such as lack of project awareness, lack of trust, delayed feedback and different time zones [3].

Different time zones causes several effects for example the more time zones you cross, the less the time when people are at work simultaneously [4]. Whereas, lack of awareness information makes it difficult to initiate contact and often leads to misunderstanding of communication content and motivation [1]. Awareness is defined by Dourish and Bellotti [11] as “... an understanding of the activities of others, which provides a context for your own activities”. Distributed teams suffer from awareness issues during the distributed software development process. Since teams are distributed globally, it is more difficult to develop and maintain such awareness in distributed software teams than in co-located ones [12]. This is because the awareness information required during software collaboration is tacit, inherent, dynamic and contextual and therefore extremely challenging to distribute automatically[12]. Lack of informal awareness causes the difficulty in facilitating casual interactions and initiation of appropriate modes of communication [13], which affect the team performance and effectiveness. As a consequence it decreases the level of successfulness of GSD projects. Moreover, the absence of ongoing conversation can also lead to surprises from distant sites, potentially resulting in misalignment and rework [3] which may cause organizations to be reluctant to adopt GSD and may cause failure in adopting it.

Many studies have not only identified the challenges in GSD but also proposed corresponding solutions to address these challenges in order to decrease the rate of unsuccessful projects as well as improve the productivity of distributed teams in GSD settings. For example, the authors of [14] argued that the productivity of collaboration is threatened by a lack of common ground among team members and lack of awareness in working processes. The common ground refers to a state when all the participants have similar or same understanding of the work and processes [4]. Teams with common ground, who do loosely coupled work, and that have high levels of collaboration readiness seem to be more likely to succeed[4]. When team members are sharing common ground such as some general knowledge about the team members’ background, project’ goals, team members’ appearance and behavior, this will help them to initiate a conversation easier as well as improving the effectiveness
of the communication as stated by (Clark, 1996) effective communication between people requires that the communicative exchange take place with respect to some level of common ground.

The abovementioned discussion justifies that addressing awareness issues in GSD teams is critical. Therefore, the developers of the ATGSD methodology proposed solutions for supporting team building activities as well as addressing some aspects of awareness in GSD.

### 2.2 Accelerated Team Building for Global Software Development Teams (ATGSD)

TEO has proposed a methodology named ATGSD, which is defined as “a specialized team building methodology for improving the social skills of a GSD team, addressing the known distances in GSD namely spatial distance, temporal distance, cultural distance and the software engineering environment. It attempts to compensate for the lack of social relations and team cohesion distributed software development team environment”. The ATGSD methodology proposes a solution that not only focuses on the technical and process skills but also focuses on the social skills. As defined by the developers of the ATGSD methodology “social skills are the ability to participate, cooperate and collaborate”. Social skills are important factors that glue the distributed team together for achieving good performance. The ATGSD methodology defines four main phases forming, norming, storming and performing[15]. In the *forming* phase, a team is formed. The team formulates a mission statement as well as a strategy for achieving the mission statement of the project. There is an exercise in this phase, which is for identifying project mission statement and team best practices. In the *storming* phase the first tasks are planned for delivery as well as solving conflicts if any in the team. In the *norming* phase, the roles of the team members are redefined, the best practices are adjusted if inapplicable and team culture is formed. There are two exercises in the *norming* phase, which are connectivity map and feedbacks. The *connectivity map* (or *communication matrix*) is used to keep track of the communication between participants. Three colors *red* (not good), *yellow* (ok), *green* (good) are used to indicate communication status of participants. Whereas, the *feedback* exercise is used to assess how the team complies with the set of defined practices that was created in the previous phase. In the *performing* phase, the team is confident and delivers smoothly. Each phase has a predefined set of activities that brings the team members close together by sharing a common understanding of the project context, solving main issues, conflicts, establishing trust among team members, learning from previous mistakes and most importantly team members are being aware of each other’s activities to minimize the global distance gap. It helps to increase the awareness among team members as well as compensating for the lack of social relation and team cohesion in a globally dispersed environment.

The ATGSD methodology is an event-based methodology, which means that team building activity is based on some events that actually happen in a project. A team may know that some certain events are going to happen for example a team change event. When a new member is added to the team (i.e., a team-changed event), the team needs to be reconstructed. The new team member must be given a fair chance to join, share information and knowledge. As a result, a team building session is built on the top of this event.
The current situation indicates that it is necessary to develop a system for supporting distributed teams that are practicing ATGSD in a globally dispersed environment.

2.3 Cloud computing and multi-tenancy
Cloud computing has the potential to address challenges associated with GSD [8] and [9]. It is defined by the US National institute of standards and technology (NIST) as “a model for enabling convenient on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”. Cloud computing comes into three forms of service models: Infrastructure-as-a-Service (IaaS), Platform-as-a Service (PaaS) and Software-as-a-Service (SaaS). IaaS is a cloud computing service model in which hardware is virtualized in the cloud. In this particular model, the service vendor owns the equipment: servers, storage, network infrastructure and so forth. A developer creates virtual hardware on which to develop application and services (Barrie Sosinsky, 2011, p.66). Amazon elastic cloud [16], Amazon storage services [16], Eucalyptus² and OpenNebula³ are example of IaaS. PaaS model describes a software environment in which a developer can create customized solutions within the context of the development tools that the platform provides (Barrie Sosinsky, 2011, p.70). In this type of service model, developers have access to the development platform through its APIs. Google App Engine[17] and Windows Azure[18] are examples of this type of service. SaaS represents applications that are built on the top of either IaaS or PaaS. It is the most complete cloud computing service model, which provides a complete infrastructure, software and solution stack as the service offering. The examples of this type of service are Google GMail[19], Google Calendar⁴, Zoho Office Suite⁵, etc.

One of the key characteristics of SaaS applications is multi-tenancy. Multi-tenancy allows a SaaS vendor to run a single instance application, which supports multiple tenants on the same hardware and software architecture [10]. Multi-tenant application could effectively reduce the cost of usage and thus has great attraction for many small and medium enterprises [20]. Multi-tenancy in cloud-enabled systems can be applied at different levels of abstractions for example application, middleware, database, virtual machine and operating system [21].

For the nature of the system, in this project we consider to apply only the multi-tenancy at the application and database level. For the data management at the database level the data architecture is identified as three distinct approaches called separate database, shared database, separate schema and shared database, shared schema [22]. A separate database is used to store the data of an individual tenant. All related data of a tenant is kept separately from other tenants. This approach is costly and has high cost of maintaining the database [22]. Shared database, shared schema is the approach where multiple tenants are accommodated into a single database, though the set of tables for each tenant is different. This approach is relatively simple to implement [22]. Shared database, shared schema is an approach where tenants are sharing everything which means that the same database and schema are shared by all

---

² http://www.eucalyptus.com/
³ http://opennebula.org/
⁴ https://www.google.com/calendar/
⁵ http://www.zoho.com
tenants. This approach requires the lowest cost since it supports large number of tenants per database server.

In order to identify the multi-tenant characteristic of cloud-enabled systems, we have conducted a structured literature review of selected papers from the available literature. These selected papers are classified into two different categories including multi-tenancy at the application level and multi-tenancy at the database level. The categorization criteria are based on the level of abstraction of proposed solutions covering the multi-tenant characteristic in cloud applications.

2.3.1 Multi-tenancy at application level
In order to achieve multi-tenant characteristic at the application level, the authors of [23] proposed a metadata driven architecture for building the multi-tenant application, which is independent of the underlying cloud infrastructure. The proposed solution uses a web service interface to access the database, which means that it is selected as database interface to hide direct access of a database server. The web services implement all possible data transactional functions for example insert, update, and delete. It is implemented in such a way that web service operations take a tenant id as a common parameter to access data. If one tenant knows the interface URL of other tenant databases then it will be able to access that tenant’s data by configuring known URL in its own metadata. Whereas, the authors of [24] presented an online application server for a multi-tenant environments based on code generation and metadata model technology. The proposed approach is a dynamic approach, which uses code generation to generate customized code for each tenant user based on the metadata model. The customized codes and application default codes are then put together and delivered to the Multi-tenant Runtime Engine for being executed. The result is finally displayed to tenant users or managers.

In contrast, the authors of [25] proposed a solution to solve issues of providing individual tenant users configurable user interfaces of multi-tenant SaaS based on their own needs. The proposed solution allows tenant users to configure graphical user interfaces by incorporating Portlet standards with configuration information described in XML. The XML documents are used to define display modules of the applications. Developers can decide the appearance and layout of the pages by reading the definition document. With Portlet technology, the portlet fragment is developed according to the individual requirements for multi-tenant of the SaaS system and provided to tenants through the user interface. Tenant users are allowed to define a customized layout and style for their own applications.

2.3.2 Multi-tenancy at database level
There is already much work on database isolation models for SaaS multi-tenancy. The authors of [26] proposed a solution to address the multi-tenant issues specifically in the educational information system. The proposed architecture utilized a schema of distributed database architecture, and a distributed database access process module. The distributed database access process module can be integrated into systems as an independent module. It parses a query statement, processes query results, and redirects a database access target of each tenant. Whereas, the author of [10] proposed a general solution providing a customizable database design for the multi-tenants

---

6 http://www.oracle.com/technetwork/java/jsr286-141866.html
application which enables tenants to create their own elastic database schema at run-time. The proposed solutions contains two type of schema tables including common tenant tables which contains all fixed tenant shared schema, and extension tables which are elastic tables that tenants can create during multi-tenant application run-time to satisfy their business needs.

In this project, we propose an approach that utilizes a shared table, shared schema approach, which means that tenants are sharing the same database and schema. The data of tenants will be stored and uniquely identified by TenantID. Moreover, we also applied the approach suggested in [23] by adding a web service layer for providing abstractions to database access. The web services are designed to take TenantID as a parameter to uniquely identify tenants.

The review of the literature on cloud computing has provided us with detailed knowledge of multi-tenant characteristics in cloud-enabled systems. This knowledge was used to devise architectural solutions incorporated in the ATGSD tool. We have utilized this knowledge in designing architecture of the ATGSD tool described in Chapter 5.
3 Research methodology

This chapter outlines the research objectives and methodologies used in the reported research project. The primary objective of this project was to provide tool support for team building in the context of global software development. Especially, we focus on supporting distributed team building activities as well as studying how to support sharing awareness information among distributed team members. In order to conduct the research, we decided to apply an ethnographically informed research approach with the aim of understanding the ATGSD practitioners’ point of view as well as identifying awareness needs for supporting distributed teams. We used a mixed method research approach. For identifying the requirements for tooling support we used ethnographically inspired approach. For developing and evaluating tooling support, we used an experimental computer science approach. We also used structured literature review for part of this research.

3.1 Research objectives

The ATGSD methodology aims at supporting the team building activities in the globally dispersed environment and enhancing awareness among team members. The ATGSD methodology developers claimed that it also helps to address communication and coordination challenges as well as improves the quality of collaboration in the distributed arrangements. As one of the methodology developers stated that the team poster artifact of the ATGSD methodology helps to ease communication:

“Basically if you and I have a better at contacting each other, and collaborating better with each other at the individual level, the team is collaborating better. So if I have your contact data on my hand, if I had talked to you already once, twice, three times, right? If I know who you are, what is your preferences are, I know what you like to achieve in this collaboration and what you would like to put in this collaboration, and you have said yes I am a part of this saying charter as you are, we have built sort of stronger communication foundation to work on than without it, that is the idea...”

Also, the ATGSD methodology directly helps to improve the quality of collaboration as stated by one of the ATGSD methodology designers:

“Why are we sitting together having conducting session with each other of one point five hours talking about collaboration and social skills, our mission together and how we achieve our mission and why are we doing that? This is all about quality of collaboration; it is all about we want to be better in collaboration.”

From the abovementioned discussion, in this research project we sought to answer the following research questions:

- **RQ1**: What are the needs of tooling support for team building methodology?
- **RQ2**: How team building efforts are supported in GSD?
- **RQ3**: How can cloud computing be incorporated in the ATGSD tool to address challenges associated with GSD?

In order to find out the answers for the abovementioned questions, we conducted an ethnographically inspired approach in which we performed interviews with the
ATGSD methodology designers and observations during the ATGSD’s phase meetings to understand how ATGSD methodology is applied in practice. As a result, the findings from the study are used to identify requirements of the ATGSD tool. A detailed explanation of requirement specification and findings from the study are discussed in chapter 4. The subsequent sections discuss the methodologies carried out with attempting to seek the answers for these research questions.

3.2 Ethnographically inspired approach

Ethnography refers to a type of research that seeks to describe and interpret a cultural or social group or system, and involves the participation of the ethnographer(s)

When designing the study, we focused on understanding how participant’s behavior, team’s culture, factors that influence the teamwork as well as how the awareness is shared among team members. Given that our research uses a qualitative approach, which aims to observe activities, to seek for understanding the informants’ point of views as well as discovering how awareness are shared among team members, we applied the ethnographically inspired approach by performing participant observation during ATGSD’s phase meetings and performing interviews with methodology designers. Since in this research project, we develop the ATGSD tool for supporting the ATGSD methodology in practice, hence the interviews are mainly taken place with methodology designers in order to elicit requirements. Additionally, qualitative data was gathered by observing a team meeting (e.g., the behavior of distributed team members).

There are totally two observations occurred at different phases of the ATGSD methodology. The first observation was conducted at the forming phase in the ATGSD methodology, whereas, the second observation occurred at the last session of the forming stage. The agenda for the second meeting is to setup a project mission statement and to define the best practices that the team shall follow during the development process. The result of the findings are summarized and studied carefully in order to elicit requirements for designing and implementing the ATGSD tool which will assist the manager to manage distributed teams and the meetings smoothly as well as incorporating the features for the ATGSD tool to enhance the awareness among team members.

For the nature of the project, the interviewees are forced as the methodology designed and the observations should occur at the organizations, which are practicing ATGSD methodology.

3.3 Organizational context

We have carried out this research with two software development organizations consisting of software development professionals from Talented Earth Organization (or for short TEO), a Danish company, headquartered in Copenhagen. TEO has specialized in software outsourcing or “Global Software Engineering” which is part of the collaboration in this study. The development department of the company is located in Pakistan. The partner company is another Danish company, which is also located in Copenhagen. It offers services in the areas of antenna, housing and homeowner’s associations as well as providing broadband TV and telephony to their associations. This partner company communicates intensively with the TEO's

---

development department in Pakistan for supporting its IT department located in Denmark. This company was selected because it is using the ATGSD methodology in which the team is distributed and communicates on a daily basis. The requirements of the ATGSD tool were obtained through a set of interviews conducted with the developers of the ATGSD methodology as well as with the methodology practitioners (distributed team members).

### 3.4 Data collection and analysis

Data was collected through the interviews and observations of the meetings. In this research project, we have conducted four interviews and two observations in total. After each meeting the author documented notes from the meeting in order to ensure that important points were captured. In addition the meeting was also recorded. Our findings are reported in section 4.1, which is also used as a resource for identifying requirements of the ATGSD tool. In order to validate our findings, we used different approaches (or by having a review on the findings) and set up a meeting with the participants in order to communicate requirements and discuss the findings and their evolution.

### 3.5 Proof of concept development

In order to realize the findings, we implemented a prototype of the ATGSD tool for supporting team building and management in a global context. At the first phase of the project, we developed a sample multi-tier web application and deployed the application on a local host for capturing the requirements. Then, we incorporated the web service layer in order to make the ATGSD tool loosely coupled and scalable. And then, in order to support multi-tenant SaaS characteristic discussed in Chapter 1 and Chapter 2, we add the TenantID column to the corresponding database tables as well as incorporate the TenantID as a parameter in the web service operations for uniquely identifying the tenants. Also, a logging web service was developed for keeping track of how often a particular tenant invokes a certain web service function. This feature potentially can be used for demonstrating pay on demand characteristic of cloud computing. Furthermore, a set of Graphical User Interfaces (GUIs) was also implemented to present to projects artifacts as well as the functionalities of the ATGSD tool to the end-users.

The ATGSD tool implemented some of the key features of the ATGSD methodology including a team poster artifact, which contains all relevant information of the team members including the personal and professional information for example profile picture, full name, interests, roles in the project, etc. Moreover, the ATGSD also provides an overview of the current projects, available teams, as well as all available participants. Different time zones between collaborative locations are also shown together with a list of available members in remote locations and collocated sites. In addition, the ATGSD tool also contains the administrative features to support an administrator to manage project’s artifact for example project’s mission statement, best practices, team poster, etc. Furthermore, the communication status between team members is also visible to the users. This piece of information keeps track of the state of communication between members, which helps the distributed team to response immediately for improving the communication in the team.
3.6 Technology evaluation
There are two approaches for evaluating the ATGSD tool. The first approach is to deploy the ATGSD tool on a dedicated server (e.g., Microsoft Internet Information Services\textsuperscript{8} and Microsoft SQL Server\textsuperscript{9}) for evaluating the initial ATGSD tool features since the ATGSD tool manages its own users as well. The second approach is to deploy the ATGSD tool on a real cloud computing infrastructure. Both approaches can be used for assessing the ATGSD tool features for supporting the different activities and roles in the ATGSD methodology as well as its impacts on collaboration in the distributed team environment for example enhancing the awareness among distributed team members.

\textsuperscript{8} http://www.iis.net
\textsuperscript{9} https://www.microsoft.com/en-us/sqlserver/default.aspx
4 Requirement specification

This chapter outlines the requirement specifications identified for the ATGSD tool. The requirements have been elicited from different resources including relevant literature review, meeting observations as well as interviews carried out with the ATGSD methodology designers. In the next sub section, we are discussing the findings related to the development of ADGSD methodology.

4.1 Findings related to the development of ATGSD methodology

This section describes findings after analyzing the extracted data from observations and interviews with the developers of the ATGSD methodology. A detailed description of the data collection and analysis method can be found in section 3.4. The result from data collection and analysis can be used as resources for eliciting the necessary requirements for the ATGSD support tool.

**Lack of face-to-face meetings:** this is one of the outstanding issues in the GSD settings, which can lead to the problem of lack of trust, miscommunication as well as many issues associated with coordination and collaboration.

**Time zone barriers:** the benefits of GSD come in many forms for example extended the working hours, low cost destinations, closer to a significant market, accessing to wider resources. However, the time zone difference also can be a burden. The time difference between Denmark and Pakistan is five hours for instance. The time difference also impacts the freshness of the employee. The presenter needs to be aware of different time zones in order to run a meeting smoothly.

**Presenter’s overhead workload:** during a meeting, the presenter needs to talk all the time in order to connect team members together. On one hand this is good that he can drive the meeting and give him the chance to communicate with team members. On the other hand, the work seems a bit like overhead since the presenter need to talk most the time and the team member passively listen. This can lead to a situation where a team member loses interest and is inattentive during the meeting. There is a need to balance the work of the presenter so other team members can participate on relevant activities.

**Communication barriers:** one of the issues identified in communication barriers is afraid of communicating, which may be due to lack of trust. The ATGSD proposes an exercise section for keeping track of communication status between team members and between the development and the customer in order to have the right and correct response to solve the communication. For example connectivity map and feedback exercises define in the norming phases are designed to serve this purpose.

**Supportive tools and needs for automation:** in order to manage the team and support collaborative work, several tools have been used during the meeting i.e., as observed by the research team there are five different tools was used to assist the meeting namely Skype, Email, Calendar, GotoMeeting and Excel. This means that in order to run the meeting, five different applications were required and the presenter had to switch back and forth between applications to complete his tasks.
Moreover, the ATGSD methodology defines four main phases including *forming, norming, storming* and *performing*. The main problem with this is that a team member has no chance to go back to see the history of a previous phase. This can lead to lack of awareness of the project. In addition in the distributed team settings, team members join and leave the team frequently. Therefore when a new team member joins a particular team he/she has no chance to review on what has been decided or discussed in the previous phase i.e., the mission statement of the project, other important decisions, etc. As a result, this leads to a lack of browsing the history data of the previous phase.

**Shared goals and common understanding factors:** one of the important factors for a project running in a distributed environment to succeed is that the team members must share awareness within the team and the project for instance activity awareness, process awareness and project awareness. The members should understand the common goals when joining to the project. The team shares the project mission statement, which defines the common shared goals of the project. The more the team shares understanding the more success the project can achieve, for example sharing knowledge between team members, and sharing design decisions that were made. As observed in the meeting, the decisions were shared to team members via email after each meeting.

**Lack of consolidated collaborative tool:** as described earlier, many different tools are used to assist the presenter during a meeting, which means that the presenter has to switch between different applications to assist his work. This may lead to discontinuous or lacking of the throughput of the process as well as lacking of structuring important data. Moreover, the relevant data of the project is stored on different applications, which may lead to lack of data management as well as backup important data.

In addition, in the distributed development setting the team boundary may vary as team members join and leave the project. The member in the team are changed frequently as the project proceeds, the list of the team members together with their profiles need to be maintained.

**Needs for intuitive Graphical User Interfaces (GUIs):** as described earlier, there is a need for supporting the ATGSD methodology with intuitive GUIs, which provides a flexible way to add and remove members as the project proceeds. The GUIs must be intuitive which means they are easy to use and provide enough information as well as the level of automation that members can update their profile data without disturbing the presenter. Moreover, the GUIs also need to display the entire ATGSD process so that the presenter may refer to the previous phase as required.

**Multiple user environments - tools deployment environment:** at the current situation, there is no available tool for supporting the ATGSD methodology. However, a combination of variant tools was used to run the process. The developing ATGSD tool will be used in the distributed development settings. It is required that the ATGSD tool is deployed on the cloud environment so that distributed teams do not need to install and maintain the application locally.
One of the characteristics of the cloud environment is resource elasticity, which means that the resource will be provisioned on demand as request peak load increase. The benefits will be clear in the long run as TEO gain more and more customers, all the customers can access the ATGSD tool without any requirement for installing and managing the ATGSD tool themselves.

**Important artifacts in the ATGSD methodology:** the development of the ATGSD methodology is not completed yet. At this current stage the *team poster* is the most important artifact of the ATGSD methodology. It provides detailed information of team members. It also contains the mission statement of the team, the culture of the team as well as communication channels, which are used in case of the Internet is down during a meeting. The purpose of the team poster is to facilitate communication, collaboration and awareness between team members because according to the developers of ATGSD methodology if the contact data, team members profile and preferences are shared between participants, this helps them to communicate easier. As one of the developers of the ATGSD methodology stated that:

“If I know who you are, what is your preferences are, I know what you like to achieve in this collaboration and what you would like to put in this collaboration, and you have said yes I am a part of this saying charter as you are, we have built sort of stronger communication foundation to work on than without it, that is the idea...”

And another developer of ATGSD methodology also added more description of the team poster to claim that it helps to enhance the awareness among distributed team members:

“Team poster has some characteristic features, one is that it is something that very visually,... it has a very simple model that you shall be able to print it out and put it all your wall for your own pleasure, the team for remembering ... the team poster has information about the time and differences and the time differences, the localization and the telephone number and fall back model if one line is off”

Beside the team poster artifacts, the ATGSD methodology also defines two other artifacts including connectivity map (or communication matrix) and feedback (or compliance matrix). As described by the developers of the methodology the two matrices are used to measure social skills in a distributed team. It verifies how well the performance of a team is. The developer of the ATGSD methodology stated that:

“...And we measure social skills, so far we have only a very.... very basic matrix for how well the performance of a team is for instance we have a matrix which is called a communication matrix, who is talking to who, how often, that is a very basic matrix we have, so are you red, green or yellow.... the communication matrix gives us a good picture of how communication is flowing in the team.”

“.... The second matrix we have is the compliance matrix to the team’s practices. So we ask to do an assessment saying how compliance are you to the practices you defined as your team practices... eh ha... are you green, yellow or red. That gives us another very basic measure about this team is actually serious of being a team if they are compliance to their practices....)”
Best team size for applying ATGSD methodology: the best team size for applying the ATGSD methodology is between seven to ten members. From experiences of ATGSD methodology practitioners, if the size of a team is large, the team building session becomes too long and boring in the end since many people are involved. As described by the developer of the ATGSD methodology, who works as a facilitator for conducting a ATGSD meeting; “the problem I see we encounter is that sometime in the meetings, the team building session become too long because everybody has to be involved and we have to listen to what ... how they see, their attitudes and how they articulate their preferences and so on... I think it is not very good in the current way done, it is not very good for more than seven or so, seven eight people in the team because the meeting then is becoming too long and boring in the end”

4.2 Requirement specifications for ATGSD tool support
This section describes the identified requirements for the ATGSD tool, which are based on the abovementioned findings. At this stage of the project, the requirements of the project are categorized into functional and non-functional requirements. The functional requirements describe functional features being implemented for the ATGSD tool whereas non-functional requirements identify the quality attributes supported by the ATGSD tool for instance performance, maintainability, usability, etc. Moreover, we rank the importance of the identified requirements by using “Nice to have” and “Need to have”. A requirement is prioritized as “need to have” indicates that it shall be scheduled for implementation because they will affect the completeness functionality of the ATGSD tool. Whereas a requirement is prioritized as “nice to have” indicates that it is less important or it is an extra feature, which might be implemented to add extra values to the ATGSD tool when there is available time.

4.2.1 Functional requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Importance</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>The ATGSD tool shall provide a feature that enables a user to create a new project.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R2</td>
<td>An authorized user shall be able to browse the list of existing projects.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R3</td>
<td>The ATGSD tool shall display a project name and a current stage of the project while displaying a list of projects.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R4</td>
<td>The ATGSD tool shall display all phases defined in the ATGSD methodology for example forming, norming, storming, performing.</td>
<td>Need to have</td>
<td>Not implemented</td>
</tr>
<tr>
<td>R5</td>
<td>The ATGSD tool shall contain a mechanism for switching between different phases in the ATGSD methodology.</td>
<td>Need to have</td>
<td>Not implemented</td>
</tr>
<tr>
<td>R6</td>
<td>The ATGSD tool shall provide a mechanism for preventing a user to navigate to a next phase when the current phase is not completed.</td>
<td>Need to have</td>
<td>Not implemented</td>
</tr>
<tr>
<td>R7</td>
<td>For each phase in the ATGSD methodology, the ATGSD tool shall</td>
<td>Need to have</td>
<td>Not implemented</td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
<td>Importance</td>
<td>Status</td>
</tr>
<tr>
<td>----</td>
<td>-------------</td>
<td>------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>display relevant activities and artifacts related that particular phase.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R8</td>
<td>The ATGSD tool shall highlight the current phase of the current opening project.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R9</td>
<td>The ATGSD tool shall provide a log-in mechanism.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R10</td>
<td>The ATGSD tool shall support the <em>team forming phase</em> as a default when a new project is created in the ATGSD tool</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R11</td>
<td>The ATGSD tool shall display a <em>team poster</em> of a project. The <em>team poster</em> shall contain the team introduction information and shall display a list of team members as well as the communication channels guideline used in case of the internet is down.</td>
<td>Need to have</td>
<td>Partially Implemented</td>
</tr>
<tr>
<td>R12</td>
<td>The ATGSD tool shall provide a mechanism for adding a new participant in a project.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R13</td>
<td>The ATGSD tool shall provide a mechanism for removing a participant from an existing project.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R14</td>
<td>The ATGSD tool shall provide a mechanism for adding a collaboration site for a project.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R15</td>
<td>The ATGSD tool shall support users to update their profile data.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R16</td>
<td>The ATGSD tool shall allow participants to upload their profile pictures.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R17</td>
<td>The user profile data shall be able to be reused from one project to another project.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R18</td>
<td>The ATGSD tool shall display a list of participants in the master-details view. In the master view, the ATGSD tool shall display: picture, name and location of a participant. Whereas, in a detail view all of the relevant participant profile shall be displayed.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R19</td>
<td>The ATGSD tool shall provide a paging function in the team poster to facilitate users navigating through participant’s profiles.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R20</td>
<td>The ATGSD tool shall provide a mechanism for defining mission statements of a project.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R21</td>
<td>The ATGSD tool shall provide a mechanism for updating, deleting the</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
<td>Importance</td>
<td>Status</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>R22</td>
<td>The ATGSD tool shall provide a mechanism for defining best practices of a team.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R23</td>
<td>An authorized user with sufficient permission shall be able to modify or delete impractical best practices</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R24</td>
<td>The defined practices shall be able to be modified, updated, added or removed</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R25</td>
<td>The ATGSD tool shall provide a mechanism for assigning a certain participant to enforce a particular best practice is defined in the project.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R26</td>
<td>The ATGSD tool shall be able to display different time zones of collaborative sites participating in a specific project.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R27</td>
<td>The ATGSD tool shall provide a mechanism for displaying time zone in a master-details view for example: the master view displays different time zones between collaborative sites whereas the detailed view depicts the availability information of each individual, which helps to enhance the awareness in a team.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td>R28</td>
<td>The ATGSD tool shall provide the exercise sessions in the <em>norming phase</em> including the <em>connectivity map</em> exercise. The <em>connectivity map</em> (or <em>communication matrix</em>) is used to keep track of the communication between participants. Three colors red (not good), yellow (Ok), green (good) are used to indicate the communication status.</td>
<td>Need to have</td>
<td>Implemented</td>
</tr>
<tr>
<td></td>
<td>The ATGSD tool shall also provide a mechanism to support administrative users to state a communication status between participants with red, yellow or green.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R29</td>
<td>The ATGSD tool shall provide the exercise sessions in the <em>norming phase</em> including the <em>feedback</em> exercise. <em>Feedback</em> exercise is used to assess how the team complies with the set of defined practices that was creating in the previous phase.</td>
<td>Need to have</td>
<td>Not implemented</td>
</tr>
<tr>
<td></td>
<td>The ATGSD tool shall provide a mechanism for marking the importance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
<td>Importance</td>
<td>Status</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>R30</td>
<td>The ATGSD tool shall be developed as a cloud-based application, which means that it shall be deployed on a cloud infrastructure.</td>
<td>Need to have</td>
<td>Partially implemented</td>
</tr>
<tr>
<td>R31</td>
<td>The ATGSD tool shall provide a calendar so that the users can share awareness as well as making an appointment for a team session meeting.</td>
<td>Nice to have</td>
<td>Not implemented</td>
</tr>
<tr>
<td>R32</td>
<td>The ATGSD tool shall provide a chat based communication channel.</td>
<td>Nice to have</td>
<td>Not implemented</td>
</tr>
<tr>
<td>R33</td>
<td>The ATGSD tool shall provide a phone call based communication channel.</td>
<td>Nice to have</td>
<td>Not implemented</td>
</tr>
<tr>
<td>R34</td>
<td>The team members shall be able to share their screens.</td>
<td>Nice to have</td>
<td>Not implemented</td>
</tr>
<tr>
<td>R35</td>
<td>The ATGSD tool shall contain a discussion board or similar kind of knowledge exchange.</td>
<td>Nice to have</td>
<td>Not implemented</td>
</tr>
<tr>
<td>R36</td>
<td>The ATGSD tool shall be integrated to an email service so that an email can be sent to team members after each team session meeting.</td>
<td>Nice to have</td>
<td>Not implemented</td>
</tr>
</tbody>
</table>

### 4.2.2 Non-functional requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Quality Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF1</td>
<td>The ATGSD tool shall provide users a mechanism for switching between different phases in the ATGSD methodology.</td>
<td>Flexibility</td>
</tr>
<tr>
<td>NF2</td>
<td>The ATGSD tool shall be ease for use and provide the users enough guideline information to facilitate team-building events/stages.</td>
<td>Usability</td>
</tr>
<tr>
<td>NF3</td>
<td>The ATGSD tool shall be simplicity and lightweight, which mean it shall provide a clear and simplicity mechanism to assist the users and display relevant and helpful information. The users don’t necessary to go through complicated steps to figure out the simple task.</td>
<td>Simplicity and light-weight</td>
</tr>
</tbody>
</table>

### 4.3 Use case model

A use case as defined in [27] is “a collection of related success and failure scenarios that describe an actor using a system to support a goal”. A scenario is a specific sequence of actions and interactions between actors and the system. The main motivation of using use case is to make it easier to capture customer goals and
motivate them to contribute their definition and review, since in software projects the lack of user involvement is near to the top of the list of reasons for project failure (Larman03). In this project, we identify two types of user for the ATGSD tool, the normal user and the administrative user, which cover the interactions of the users with the ATGSD tool. Figure 1 and Figure 2 respectively depict the use cases of the normal users and the administrative users.

4.3.1 Use cases for a normal user
The main interactions of normal users with the ATGSD tool are to simply view the awareness information including project, team, participants, and different time zones of the team members who are globally distributed in many different countries. Figure 1 shows typical use cases of a normal user.

![Figure 1: Use cases for a normal user](image)

4.3.2 Use cases for an administrative user
The administrative users of the ATGSD tool have wider privileges in comparison with the normal users. As administrative users, they can manage the project, team, project mission statements, team best practices so on as illustrated in Figure 2. We name use cases using high-level description names for example “manage project”, “manage team”, etc. However, “manage project” may include sub use cases for example “create project”, “edit project”, “delete project”, “browse project list” and so on, similar use case naming convention is applied for other groups of use case as well e.g., team, participant, mission statements. In this section, we decided to describe most important use cases in the fully dressed text fashion, for other use cases in the same use case group (e.g., edit, update, delete, etc.) we consider that they are less important to describe here since steps are more or less repeated similarity. Error! Reference source not found. to Error! Reference source not found. describes the use case text for administrative users.
Figure 2: Use cases for an administrative user
5 Architectural overview

This chapter describes the architectural design decisions in terms of architectural and design patterns as well as the problems and proposed solutions to address these problems.

5.1 Domain model

This section describes the various elements of the problem domain as well as the structure of each element and the relationship between the elements and their constraints as well.

In the middle of Figure 3 are the participant and project elements. The participant element represents a participant in a team (or a team member). The participant can be located globally in different locations. A location has a name and is belonging to a country. Moreover, each location also has a time zone corresponding to that particular location for example a large country may have different locations, which are belonging to different time zones. A participant is a member of a team, from the interviews with the methodology designers; the ideal number of the participant in ATGSD team is usually from seven to ten. Each team is associated with a particular project; however, a new established project can reuse existing teams if required.
Each particular project has defined its own mission statement and best practices. As defined by the developers of the ATGSD methodology, mission statements of the project are the common shared goals between the team members. In a distributed setting, the more the team shares understanding the more success the project can achieve. Each project also has a set of best practices, which are the agreements between team members to ensure that a team achieves its goal. An example of a best practice can be: code review or using the task management, unit testing tool and the degree of automatic testing or a best practice to bridge the gap in communication and achieve better understanding of other tasks and so on.

Each participant has a specific role to the project for example software developer, project manager, CTO, test engineer and so on. This kind of role is different from a role that is defined for using the ATGSD tool such as a normal user role or an admin user role. Each participant is belonging to a particular tenant. The term tenant refers to a single organization (e.g., an institution) or a person [22]. The reason for us to model the tenant in this domain model is that the ATGSD tool is planned to serve as a multi-tenant SaaS application. Multi-tenancy refers to a technology where a single instance of the application serves requests from multiple clients [10]. In the next section, we are going to discuss the architectural model of the ATGSD tool in details.

5.2 Architectural model

We decided to use the multi-tier (layers) architecture for developing the ATGSD tool. Layered architecture is an architectural style that structures applications so they can be decomposed into groups of subtasks such that each group of subtasks is at a particular level of abstraction [27]. The benefits of layered architecture include:

- Separation of concerns, a separation of high from low level services, and of application-specific from general services. This reduce coupling and dependencies, improves cohesion, increase reuse potential and increase clarity.
- Related complexity is encapsulated and decomposable
- Some layers can be replaced with new implementations and can be distributed
- Development by team is aided because of the logical segmentation.
- By designing the ATGSD tool using layered architectural design pattern, we are able to insert the web service layer in to the layer stack when migrating from the web-based system to the service-based system with only minor changes and modifications. As shown in Figure 4, we divide the ATGSD tool into three main architectural layers including:
  - **Web Presentation Layer**: this layer serves the web site (dynamic HTTP content) and acts as an interface to the business logic of the application.
  - **Web Service Layer**: this layer provides services to the web presentation layer and implements business logic. The web service layer accesses the data layer via the data access layer.
  - **Data Access Layer**: this layer provides access to the data layer by using ADO.NET Entity Framework. The ADO.NET Entity Framework [28] provides a mechanism for managing the transformations between the logical database schema that’s present in the relational store and the conceptual Entity Data Model (EDM) schema used by the application10. The EDM is an Entity-Relationship data model; by using the EDM the ADO.NET Entity Framework

allows data to be treated as entities independently of their underlying data store representations.

- **Data Layer**: this layer stores the application’s data in the database schema.

![System layered architecture](image)

**5.3 Architectural factors**

This section discusses the important factors that drive the architectural design decisions of the ATGSD tool. For each architectural factor, we outline main concerns and propose corresponding solutions after carefully analysis and performing trade-offs for each design decision-making.

**5.3.1 Maintenance**

This section discusses the problem and proposed solution associated with maintenance of the ATGSD tool.

**5.3.1.1 Problem**

The ATGSD tool is constituted by a series of functions that should provide important information about team members in a global context. This information involves personal data such as name, roles, email address, interests, etc. and other awareness information i.e., availability of team members, communication status, etc. Moreover, the extension of the ATGSD tool may also include consuming data from other APIs i.e., LinkedIn. The information provided to users is heterogeneous, and highly different in terms of functionalities. It is a risk that the ATGSD tool is designed and implemented in a monolithic architecture, which affects code reusability as well as maintainability of the ATGSD tool. Since the responsibilities of the system components are not clearly distinguished (i.e., high coupling and low cohesion – the graphical user interfaces contain components or elements for handle the business logic of the application for instance).

**5.3.1.2 Solution: Layers**

In order to solve this problem, the layered architectural design pattern is applied for the tool to distinguish and distribute the responsibilities among components. In this
way, the application is divided into layers with corresponding responsibilities for each layer. For example presentation layer contains only components and elements for presenting data to the end-user, whereas the business layer resides on another layer handling the business logic of the application, and the data access layer handles the data persistence, etc. Since the responsibilities and functionalities are clearly defined for one layer, replacement, modification or removing services of one layer have minor impacts on other layers. Therefore, maintenance of and enhancements of the solution are easier due to the low coupling between layers, high cohesion between the layers, and the ability to switch out varying implementations of the layer interfaces.

5.3.2 Scalability
This section discusses the problem and proposed solution regarding to the scalability of the ATGSD tool.

5.3.2.1 Problem
The primary aim of the tool is to support team building and enhance awareness among team members. At some stage in the software development life cycle, the request for using the ATGSD tool is higher in comparison with other stages. For example the request of using the ATGSD tool is high at the early stage of the project when a new project is established and a new team is formed. This requires that the ATGSD tool shall be able to elastically response to the increasing number of user requests. In other words, the design of the ATGSD tool shall consider supporting the scalability quality attribute.

5.3.2.2 Solution: Service Oriented Architecture (SOA)
In order to solve this problem, we apply the SOA architectural style for the ATGSD tool. SOA is defined as an architectural style where systems consist of service users and service providers. An architectural style defines a vocabulary of component and connector types and constraints on how they can be combined (Shaw, 1996). Whereas, web service technology to implement SOA architecture. A service is self-contained, highly modular and can be independently deployed [29]. The web service layer is incorporated into the ATGSD tool. It resides on the top of the data access layer and utilizes services of the data access layer. The web services in the web service layer expose the data and functionalities to clients through well-defined service interfaces. By applying the SOA architectural style for the ATGSD tool, we are able to decouple the component’s dependencies from using library references to using a well-defined HTTP11 connection. As a result, it enhances the independence among components. The web service layer is deployed independently with other layers; hence it enhances scalability of the tool.

5.3.3 Data and functionality visibility
This section describes the problem related to the data and functionalities that are accessible for a particular user.

5.3.3.1 Problem
For supporting awareness among team members, the ATGSD tool provides the team poster, which contains all the relevant personal and professional of team members. It is required that only a user who owns that particular profile data shall be able to

---

11 http://www.w3.org/Protocols/rfc2616/rfc2616.html
modify it. In addition, only a user with the administrative privileges shall be able to create a new participant and assign roles to the participants (or the team members). Moreover, only users with administrative privileges shall be able to perform the administrative operations for example creating a new project, add new participant, etc.

5.3.3.2 Solution
In order to solve this problem, we incorporate a role management mechanism provided by ASP.NET, which allows developers to define roles for system users in order to restrict access to the application. The ATGSD tool defines two different roles including normal user and admin user in order to limit access of normal users to the administrative operations.

For preventing modification of other profile’s data, we apply the authentication mechanism that validate whether a logged-in user owns the profile. The validation checks if an email property is matched since email is a globally unique property.

5.3.4 Persistence Management
This section describes the problem and proposed solution associated with the data access layer of the ATGSD tool.

5.3.4.1 Problem
The ATGSD tool requires handling information of participants, projects, etc. In addition, it needs to persist the information to a database management system. In order to achieve that, it uses Microsoft SQL Server. As a result, the challenge is how to manage the connection to the database and how to provide a flexible mechanism for switching between different data sources.

5.3.4.2 Solution
There are many available approaches for solving this problem. The first approach is manually constructing the database connection object (using SqlConnection and SqlCommand classes in .NET framework [30]) to access the database. The benefit of this approach is that developers have full control over the implementation. The drawback of this approach is that returned data from a database call is in a tabular format, which requires more programming effort to encapsulate returned data to an object-oriented model. The second approach is to use the SqlDataSource control, which is implemented as a control allowing access and manipulating in ASP.NET pages without using ADO.NET classes directly. However, this approach violates the layered architecture discussed in 5.3.1 since the implementation of the presentation layer contains the logic of the data access layer. We decided to apply the third approach, in which we use the Entity Framework for handling the connection to database. Precisely, the ObjectServices in the Entity Framework [28] manages the connection to the database. It opens connections only when required. Moreover, the ADO.NET Entity Framework provides the right level of abstraction and a mapping mechanism to map objects and their state to a relational data store allowing a conceptual model to be defined independently of the concrete data store that may be used and it becomes possible to create a model of entities that hold data in an object-

---

oriented way. This helps to shorten the development time and simplify the way the
ATGSD tool queries data from the database for example no connection object is
required, and there is no need for constructing any SQL queries, which is error-prone.

5.3.5 Isolation and sharing among tenants
This section describes the problem and proposed solution regarding to the multi-
tenant characteristic of the ATGSD tool.

5.3.5.1 Problem
One of the main goals of the ATGSD tool is to support team building activities in the
global context. It can be used by many organizations at different phases in a software
development life cycle. In order to reduce the operational cost as well as provide a
high availability, the ATGSD tool shall be provisioned as a service supporting the
multi-tenancy characteristic. It means that software vendors who own the ATGSD
tool shall be able to run a single instance of the ATGSD tool for supporting multiple
tenants on the same hardware and software architecture. In order to fulfill this
requirement, the ATGSD tool shall provide a secure mechanism for isolating data
among tenants.

5.3.5.2 Solution
Multi-tenancy is the primary characteristic of SaaS. It allows a SaaS vendor to run a
single instance of an application, which support multiple tenants on the same
hardware and software architecture [20]. In order to solve this problem, the researcher
has conducted a structured literature review of selected papers associated with multi-
tenant architecture of SaaS applications. The selected studies proposed solutions on
different areas that could be applied multi-tenancy for example application,
middleware, database, virtual machine and operating system. For the scope of the
project (or time constraints) we decided to implement the multi-tenant at the database
level. Where tenants are isolated by the tenant id incorporated in tables in the database
so that each tenant has access to their own data. Moreover, the ATGSD tool is
developed based on SOA principles with a set of implemented services covering the
entire functionalities of the ATGSD tool. With this design, multi-tenancy can also be
achieved at the application level by configuring web services to point to different
database servers owned by tenants. Precisely, a mapping table needs to be developed
in the ATGSD tool for keeping track of tenants and their corresponding database
servers. At run-time the connection string to the database is dynamically constructed
based on the information of a particular tenant for routing to the correct database
server. Currently, the ATGSD tool is designed to be compatible with the Platform as
a Service (PTaaS) platform. It utilizes authentication services provided by the
platform for authenticating tenant’s users. In section 5.3.6, we describe how the
Platform is incorporated to provide the authentication service for authenticating tenant
users.

5.3.6 Integration with external web services and platform
This section describes the system design considerations support for integration of the
ATGSD tool with external services and systems. In the software development life
cycle, the software development tools need to be properly integrated with each other
in order to support end-to-end software development life cycle to maximize the

productivity of the stakeholders involved in software development activities [32]. In this case, the ATGSD tool is integrated in PTaaS platform. PTaaS is a platform for providing software development applications and tools as a service. The stakeholders may request PTaaS for a software development tool at a particular phase in the software development process; for example, the tenant user may request for the ATGSD tool, which requires that tenant users need to be authenticated by the PTaaS platform, since the tenant is registered with the platform. The main problem and proposed solution are discussed in the subsequent sections.

5.3.6.1 Problem
A tenant user who is already registered to the PTaaS platform needs to access the ATGSD tool at the team-forming phase. In order to handle this request, the ATGSD tool shall be able to consume the authentication service of the PTaaS platform to authenticate the tenant user for that particular request. Figure 5 shows the communication between two systems for validating user access.

![Figure 5: Calling the authentication service from PTaaS](image)

5.3.6.2 Solution
In order to solve this problem, the ATGSD tool shall contain a tenant web service. The functionality of this service is to invoke the authentication web service with user provided credentials. If the provided user credential already exists in the PTaaS platform, the tenant web service will also create a new user in the ATGSD tool with the given credentials and adding this user to the tenant. A detailed description and implementation of this solution are discussed in section 5.4.4.

5.4 Design specification
This section describes the design specification of the ATGSD tool, which is designed to fulfill the requirements and design for implementing the solutions discussed in the previous section. In the next section, we will describe the overall package diagram of the ATGSD tool.

5.4.1 Overall package diagram
Figure 6 depicts the package view of the ATGSD tool, package contains three main packages: presentation layer, service layer and data access layer. The external systems package represents the integration with the PTaaS platform. Logging package presents the simulation of pricing model services in the cloud-based application, the data sources package represents the implementation of the database, whereas the
external services represents the application of using the ADO.NET Entity Framework service for mapping the database schema to the data source business entities. Detailed explanation of the presentation layer, service layer and data access layer packages are provided in the next sections.

Figure 6: Package diagram

5.4.2 GUI layer
The presentation layer presents the data to end-users and provides a mechanism for end-users to interact with the ATGSD tool. The GUI layer is a set of ASP.NET classes, which utilizes web service clients to invoke web services for manipulating data and displaying final results to end-users.

Figure 7: Presentation Layer

As shown in Figure 7, the presentation layer contains two important components: the web interface and the presentation logic. The web interface is a set of ASP.NET pages, which contain the mark-up for defining how data and web user interface controls are rendered and displayed to end-users. The presentation logic contains code behind classes corresponding to ASP.NET view pages, which have references to web service clients and utilize a service client to invoke a service function call for desired
data and functionalities. Moreover, the presentation logic also prepares the data set provided to the web user interface controls and using the .NET data binding mechanism to bind the data returned from a service call to web user interface controls which are objects on ASP.NET Web pages that run when the page is requested and that render markup to a browser.

5.4.3 Web service layer
This section describes the design of the web service layer of the ATGSD tool, which attempted to address the scalability problem described in section 5.3.2. The web service layer resides on the top of the data access layer and right below the presentation layer. It contains three mains components: service interfaces, web services, and web service clients. Web services as defined in [33] are autonomous, platform-independent entities that can be described, published, discovered, and loosely coupled in novel ways. In the ATGSD tool, web services act as the provider of data and functionalities, whereas the client acts as a consumer. The web service client invokes web service’s operations through the service interface. In the design of the ATGSD tool, web pages (ASP.NET pages) utilize web service clients to communicate with the web services for desired data and functionalities. The processed data is returned back and bound to the relevant web interface control before being rendered as markup to end-users through the browser. Figure 8 illustrates the components of the web service layer.

![Figure 8: Web Service Layer](image)

5.4.3.1 Web service interfaces and service data contract
A service interface is a service contract, which declares a set of service functions that a web service provides to its clients. In this project, we decided to implement the service interfaces, and data contract in separate assemblies. With this approach, we can later simply distribute the service contracts to third parties who wish to consume the service without disclosing their actual implementation. The data contracts of the project web service are shown in Figure 9.
Moreover, we can create the ChannelFactory, which is a factory for creating the communication channel at runtime. The ChannelFactory\(<T>\) takes a generic parameter of the service type (e.g., service contract or service interface) that we want to create a channel for, which means that no matter how often a service contract changes, we don’t need to manually re-generate proxy classes or alter client-side service model configurations each time. The relationship between a service contract and ChannelFactory is shown in Figure 10.

### 5.4.3.2 Web service client

The main responsibility of a service client is to invoke a call to a web service. The client should not know about the implementation details of the web service. All the client should be aware of is that there is an object, which provides some useful functionality. Everything else about the service should be hidden. There are many different ways to create a WCF service client using .NET [30] technology for example generate a service proxy, or using the ChannelFactory as described in the previous section, or using HttpClient or WebClient in case of WCF RestFul web service. Microsoft Visual Studio 2012 IDE 15 also supports generating a proxy by using the “Add New Reference” feature. The disadvantages of this approach is that we need to regenerate the proxy each time the service address changes (e.g., WSDL) and we rely on the Microsoft Visual Studio 2012 IDE to handle the hard work and hence loose control. The second approach is to use a ChannelFactory to create a communication channel of a known service type (a service contract or a service interface). We decided to apply the second approach since we have control over the clients and the web services. Moreover, we do not need to regenerate the proxy every time a web service address changes, since we only need to add a reference to the new service interfaces 16.

Figure 10 shows the class diagram of the web service clients. The web service clients are derived from a generic ServiceClientBase, where \(T\) is a service contract (or a service type). The ServiceClientBase is a composer of the ChannelFactory class; this class is a .NET library class. The responsibility of the ChannelFactory is to create channels of different types that are used by clients to send messages to variously

---

15 [http://www.visualstudio.com](http://www.visualstudio.com)
configured service endpoints. After a communication channel is being created, the service clients are able to communicate with the web service through this channel. We will describe the web services of the ATGSD tool in the next section.

5.4.3.3 Web services
This section describes the design of the web services in the ATGSD tool. By applying the service-oriented architecture in the ATGSD tool, the services can be deployed independently of each other and ensure loose coupling with the web presentation layer, which helps to address the scalability challenges of the ATGSD tool discussed in section 5.3.2. Figure 11 shows a list of web services were designed for the ATGSD tool including Best Practices, Collaborative Location, Mission Statement, Participants, Projects, Teams, Connectivity Map, Feedback and Tenants services. The reason for us to fragment into a group of small services is that it helps to maintain and replace a particular service without affecting other services. Additionally, the ATGSD tool will still be able to function although with reduced functionality. Each of the web service mainly contains functions to cover the Create-Read-Update-Delete (CRUD) operations of the database. However, some services may contain additional functions e.g., get list of objects or for performing the business logic of the application e.g., set communication status, etc.

Figure 10: Web service client

Each web service is a client of the log web service. The log web service is a simulation web service. It is developed for keeping track of how many times a particular web service is being accessed. Based on this information together with a pre-defined pricing model, a bill may be calculated. This service is used to demonstrate the pay on-demand characteristic of cloud computing.

Figure 12 shows the class diagram of the Project web service. The interface declares eight functions that a Project web service must implement as a contract between web service providers and web service consumers. Table 1 briefly summarizes all of the function declarations in the IProject service interface.
Table 1: IProject service interface

<table>
<thead>
<tr>
<th>Operation Contract</th>
<th>Parameters</th>
<th>Briefly Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetProjects</td>
<td>tenantId</td>
<td>Get a list of project, which is belonging to a particular tenant</td>
</tr>
<tr>
<td>GetProjects</td>
<td>None</td>
<td>Get a list of existing projects</td>
</tr>
<tr>
<td>GetProjectById</td>
<td>• Id: Identification of the target project</td>
<td>Get the project with a provided project id</td>
</tr>
<tr>
<td>Create</td>
<td>• Project</td>
<td>Create a new project</td>
</tr>
<tr>
<td>AssignTeam2Project</td>
<td>• projId: Identification of the target project • teamId: identification of the target team</td>
<td>Assign a team with a given team id to a particular project</td>
</tr>
<tr>
<td>UnAssignTeamFromProject</td>
<td>• projId: Identification of the target project • teamId: identification of the target team</td>
<td>Remove the team which has been assigned to a particular project</td>
</tr>
<tr>
<td>Update</td>
<td>• Project</td>
<td>Update the project information</td>
</tr>
<tr>
<td>Delete</td>
<td>• projId: identification of the target project</td>
<td>Delete the project with a given project id</td>
</tr>
<tr>
<td>SetProjectStage</td>
<td>• projId: the identification of the current project. • CurrentStage cs: this is a enumeration type contains four values representing four stage in the ATGSD methodology: forming, norming, performing and re-norning</td>
<td>Set the new stage for the project</td>
</tr>
</tbody>
</table>

The web service layer is on the top of the data access layer and utilizes data and business logic provided by the data access layer. Data access layer is the topic for discussion in the next section.
5.4.4 Integration with external services and platform

This section describes the design specification for integration with external services, which attempted to address the challenge described in section 5.3.6. Figure 14 illustrates the activity diagram describing the activities that a particular user has to perform for accessing the ATGSD tool. When a user first accesses the ATGSD tool, it checks whether the user is valid or not, if so he/she starts to use the ATGSD tool. Assume that an administrative user attempts to access the ATGSD tool for the first time; the ATGSD tool validates the user input credentials. If the ATGSD tool does not recognize the provided credentials, it will invoke a service call to the PTaaS and passes the user credentials (e.g., email and password) to the authentication service of the PTaaS platform. An email is chosen as a user credential since it is globally unique so that the ATGSD tool shall be able to distinguish the users from different tenants. After being invoked by the ATGSD tool, the PTaaS platform processes the request, if the user is already registered, the PTaaS platform returns the tenant information in XML format as illustrated in Figure 13.

![Figure 13: XML representation of tenant information](image)

After receiving the response from the PTaaS platform, the ATGSD tool invokes its tenant web service for creating a new tenant. A new user with an administrative privilege is created as well. This user is a representation of the project manager as well as meeting host, who can later access the ATGSD tool to create users (or participants) in a team. Since a tenant and its users were created, the users can access the ATGSD tool without the intervention of the PTaaS platform.

![Figure 14: Log-in Activity Diagram & Accessing External web service](image)
5.4.5 Data access layer
This section explains the design of the data access layer, as shown in Figure 15 this layer contains two main components: the ATGSD Model component and Data Access component. The ATGSD Model component is the Entity Framework entity data model (EDM), which is a conceptual data model transformed from a logical database schema presented in the relational database and being used as the data access component. The data access component provides a mechanism for accessing and manipulating entities defined in the ATGSD Model.

![Data Access Layer](image)

Figure 15: Data Access Layer

As described in section 5.3.4, the ADO.NET Entity Framework [28] was chosen for handling the persistence of data in the ATGSD tool. Additionally, we decided to use the database first approach because we can develop a database separately and we can manually change the database tables and use the Entity Framework [28] to update the model from the database. However, by using this approach, the business entities are generated by the Entity Framework [28] will contain the additional properties representing the relationship among tables in a database schema or business entities after mapping. This association information did not need to be exposed to the service client. Therefore, we were required to perform a two-way mapping for reading and writing data between data access layer and the web service layer.

5.4.6 Interactions among layers
In this section, we utilize sequence diagrams for describing the logical view of the ATGSD tool. The sequence diagrams show interactions among components and elements for displaying a list of current projects in the Overview page’s panel. In order to perform this task, there are many communication processes between the ProjectClient and ProjectWs, which are handled by the underlying Windows Communication Foundation [34] framework. However, to keep the sequence diagram simple, we do not show the interactions handled by the framework.

The interactions between components and elements of the presentation layer and web service client are illustrated in Figure 16. When the Overview page is first loaded, the page load event is triggered, which causes the Page_Load function to be called. The logic defined in the Page_Load function is to create a new ProjectClient object. The ProjectClient class is a sub-class of the ServiceProxyBase class. Hence, the process of creating new ProjectClient object causes a creation of based object ServiceProxyBase, which in turn instantiates a new ChannelFactory instance. The ChannelFactory is a .NET [30] library class, which is a generic class. In this case, the IProject is passed as a generic type to the ChannelFactory. The logic is defined for
The constructor of the ServiceProxyBase class also calls the CreateChannel function of the ChannelFactory instance to create the communication channel of IProject type. After a communication channel is established, the Project client can invoke the Project web service functions through the pre-defined service contract IProject.

![Sequence Diagram - Retrieve and Display a list of projects](image)

Figure 16: Sequence Diagram - Retrieve and Display a list of projects

Figure 17 shows the IProject service contract and the interaction between the Project service client and the Project web service.

![Service Contract of Project provider and consumer](image)

Figure 17: Service Contract of Project provider and consumer

Figure 18 explains interactions that occur at the web service layer and the data access layer. When the GetProjects service operation is invoked, the ProjectWs instance calls the GetAllProjects function of the ProjectDbOperations instance. The ProjectDbOperations class is a class defined in the data access layer for handing data access and manipulation. For each service call, the WriteToLog service function of the Logging web service is initiated for keeping track of how many times a service operation is being accessed. This is an extra implementation for demonstrating how to calculate price of pay-per-use characteristic in cloud computing.
The returned from GetProjects service operation call is a list of projects (if any). On the Overview page, the result is saved into a session variable for performance considerations, since we only access the service as required i.e., data list has been altered. The list of project is used as a data source, which is bound to the GridView control in order to present the list of projects to the end-user.

![Sequence Diagram](image)

**Figure 18: Sequence Diagram- Retrieve a list of projects from data access layer**

### 5.4.7 Deployment overview

This section outlines the deployment view of the ATGSD tool. The ATGSD tool is designed by using layered architectural model in which it is constituted by different layers. Each layer focuses on its own tasks and provides relevant services to other layers. This design principle decreases coupling between layers so that each layer can be deployed independently on different server instances. Detailed architectural description of the ATGSD tool can be found in sections 5.2, 5.3 and 5.4.

As illustrated in Figure 19, we are applying Amazon EC2 instances, which run Microsoft Windows Server and SQL Server\(^\text{18}\) as an underlying infrastructure for hosting the application. The web presentation and web services of the ATGSD tool are hosted on Amazon EC2 Windows Server 2008 [35] instances. The underlying persistence unit is hosted on Amazon EC2 running Microsoft SQL Server [35]. The Amazon Cloud Watch\(^\text{19}\) and Amazon Load Balancer\(^\text{20}\) are incorporated to the instance that hosts the web services of the ATGSD tool for enhancing the reliability and scalability of the tool.

---

18 http://aws.amazon.com/windows/
19 http://aws.amazon.com/cloudwatch/
20 http://aws.amazon.com/elasticloadbalancing/
After describing the design specification of the ATGSD tool, in the next chapter, we are going to discuss the implementation of the ATGSD tool.
6 Implementation overview

This chapter describes the implementation of the ATGSD tool in details as well as explaining the technologies used to facilitate the implementation. Technologies and supportive frameworks are discussed in the next sub section.

6.1 Technologies and supportive frameworks

We decided to implement the ATGSD tool using .NET framework [30] since the development team has more than three years of experience and are confident in development using this framework. In the sub-sections we are going to explain the relevant frameworks that are used to accomplish the requirements for the ATGSD tool including ASP.NET [31], Windows Communication Foundation [34], and ADO.NET Entity Framework [28].

6.1.1 ASP.NET

ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applications with a minimum of coding. ASP.NET is part of the .NET framework. It was developed by Microsoft to allow programmers to build dynamic web applications. It was first released in January 2002 with version 1.0 of the .NET framework, and is the successor to Microsoft's Active Server Page (ASP) technology. In this project, we use ASP.NET framework for developing the web presentation layer of the ATGSD tool.

6.1.2 Windows communication foundation (WCF)

Windows Communication Foundation (WCF) is a framework for building service-oriented applications. It is a runtime and a set of APIs for creating systems that send messages between services and clients. In this project, we are using WCF framework for developing web services and corresponding web service clients of the ATGSD tool.

6.1.3 ADO.NET entity framework

As explained in section 5.4.5, ADO.NET Entity Framework database-first approach is used to implement the entity relationship mapping between the database schema and the business entity object so that the interaction to the database tables is abstracted to the object-oriented manner.

6.2 Data model

This section describes the data model of the ATGSD tool. Based on the analysis of domain problems, we designed the database model for the ATGSD tool as illustrated in Figure 20. We used Microsoft SQL Management Tool for generating the diagram, which shows the most important tables and the relationships among them. There are eleven tables shown in the diagram.

- **Participant**: that contains all information regarding to a participant such as id, first name, last name, interest, picture and the information supports

---

21 http://msdn.microsoft.com/library/4w3ex9c2.aspx
collaborative activity for example email, Skype name, phone, location, availability, office hour and so on.

- **CommunicationMatrix**: table for gathering the communication status of participants in a project, therefore it holds two references to the participant table, one is the *user id* which represents the user in the based location, and another reference is the *CommUserId* which represents the participant in a remote location. The Comm_Status is used to store the communication status of these two participants in the collaborative activities in the project.

![Data model diagram](Image)

- **Location**: table for gathering the information of the collaborative locations where the participants are located. The table contains the information of the current location for example *id*, *location name*, *country id* which stores the country where the location is belonging to for example location can be Copenhagen, Islamabad and so on and a *time zone id* to represent a particular time zone for the current location. Moreover, the location table also contains the *tenant id*, which is used to distinguish between tenants that use the ATGSD tool.

- **Country info**: this table holds the information of the countries of the participants involving in the project, like Country ID, Country Name and Calling ID.

- **Timezones**: table for gathering data regarding to the time zone of locations where the participants in a project come from. This table contains the Id and Display Name.

- **Project**: that contains all the information regarding to the project for example *id*, name, description, Created, Finished and so on. The Created and Finished store the start and finish date. By default, the start date of a project is set to the date on which the project is created. Moreover, the project also is also associated with a team. The Tenant ID is used to differentiate tenants that sharing the application instance.

- **MissionStatement**: table for gathering data of the project mission statements as understood as shared goals between participants in the project. The mission statement table contains id, description that is a text describing the goals that
the team would like to achieve and a project id represent a particular project where the mission statement is defined.

- **BestPractice**, that contains all the information for a best practices defined in a particular project such as id, project id and description.
- **Team**, table for gathering data of a team like id, name, description, created by, active. The Tenant ID is used to differentiate tenants that sharing the application instance.
- **Tenant**, that contains all information related to a tenant such as id, name, email and address. This table is designed and implemented in respect to the tenant data returned from the PTaaS platform.

6.3 **Graphical user interfaces**

This chapter describes the design and the implementation of the graphical user interfaces (GUIs) of the ATGSD tool.

6.3.1 **Graphical user interfaces for normal users**

This section discusses the design and implementation of the GUIs for normal users, which mainly provide a mechanism for browsing/viewing the project, team awareness information (e.g., projects and teams, participants in a project, team poster, available team members, etc.). Next section, we are going to describe the login user interface, which is shared for both admin users and normal users.

6.3.1.1 **Login user interface**

The ATGSD tool only allows authenticated users to access. Additionally, an authorization mechanism is used to restrict the accesses to the functionalities of the ATGSD tool by preventing accesses to administrative operations for normal users.

As illustrated in Figure 21, the ATGSD tool requires users to provide the sufficient credentials using email and password. The email is used instead of username since email is globally unique. Therefore, the ATGSD tool is able to distinguish users who are belonging to different tenants. After a user is successfully logged in, the ATGSD tool navigates the user to the Overview page, which presents all relevant project awareness data for example a list of current projects, available teams and collaborative locations.

6.3.1.2 **Overview user interface**

As shown in Figure 22, the Overview page contains important panels including: current projects, available teams and collaborative sites. The **current projects panel** displays a list of current projects, from each project the user is able to view the team
poster of the team, which is assigned for the project. The **available teams panel** shows a list of available teams. A new created project is created, this team list is available for the user to choose to assign to the new project, or the user can create a new team for working with the project. From the available teams panel, the user is able to clicks on the team name to view the team information in details including team name, description, created date, created by and if the team is active or not, together with the list of participant who are members of the team. The **collaborative sites panel** shows the list of countries and locations where the team members are located.

<table>
<thead>
<tr>
<th>Current Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Name</strong></td>
</tr>
<tr>
<td>TEO Project</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Available Teams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Team Name</strong></td>
</tr>
<tr>
<td>TEO Team</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Collaborative Sites</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Country</strong></td>
</tr>
<tr>
<td>Denmark</td>
</tr>
<tr>
<td>Pakistan</td>
</tr>
</tbody>
</table>

**Figure 22 - Overview page**

### 6.3.1.3 Team poster user interface

Team Poster is one of the most important artifacts in the ATGSD methodology. It contains detailed information of all the team members participating in the project which helps to enhance the awareness among team members for example profile picture, full name, contact information, personal interests, professional experiences and so on as stated by methodology designer:

> "Team poster has some characteristic features, one is that it is something that very visually,... it has a very simple model that you shall be able to print it out and put it all your wall for your own pleasure, the team for remembering ... the team poster has information about the time and differences and the time differences, the localization and the telephone number and fall back model if one line is off"

As shown in Figure 23, the team poster displays detailed information of four team members on a single page and the paging function is incorporated to support the user navigate through the list of existing team members in case of a single page cannot present the entire information of the team members.
<table>
<thead>
<tr>
<th>Picture</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Picture 1" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fullname</th>
<th>Muhammad Rizwan</th>
<th>Jibran Masud</th>
<th>Gufran Usman</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>TEO</td>
<td>TEO</td>
<td>TEO</td>
</tr>
<tr>
<td>Location</td>
<td>Pakistan</td>
<td>Pakistan</td>
<td>Denmark</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Email</th>
<th><a href="mailto:aslam.khan@teo-intl.com">aslam.khan@teo-intl.com</a></th>
<th><a href="mailto:Muhammad.rizwan@teo-intl.com">Muhammad.rizwan@teo-intl.com</a></th>
<th><a href="mailto:Jibran.masud@teo-intl.com">Jibran.masud@teo-intl.com</a></th>
<th><a href="mailto:Gufran.usman@teo-intl.com">Gufran.usman@teo-intl.com</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Skype</td>
<td>aslam.khan.iui</td>
<td>rizwan.mughal83</td>
<td>jmasud</td>
<td>gufran</td>
</tr>
<tr>
<td>Mobile</td>
<td>+92 323 934 4284</td>
<td>+92 333 714 5937</td>
<td>+92 322 400 5421</td>
<td>+92 3123 3340</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Role</th>
<th>QA Test Engineer</th>
<th>Software Developer, iOS</th>
<th>Overall delivery manager, Mobile</th>
<th>Sales, Account Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>My office hours</td>
<td>From: 9 To: 18</td>
<td>From: 9 To: 18</td>
<td>From: 10 To: 19</td>
<td>From: 9 To: 19</td>
</tr>
<tr>
<td>Availability of hours</td>
<td>Flexible, contact if online</td>
<td>Contact if online</td>
<td>If online... call me anytime</td>
<td>If online... call me anytime</td>
</tr>
<tr>
<td>Business domains</td>
<td>Medical, CRM, office management system, SharePoint</td>
<td>GNAT, dealchicken.com</td>
<td>Ad. event management, mobile app, mobile backup</td>
<td>Sales, telecom, insurance</td>
</tr>
<tr>
<td>Technology experiences</td>
<td>Java, PHP, .NET</td>
<td>iOS native</td>
<td>Mobile business, management</td>
<td>None, product in terms of business perspective</td>
</tr>
<tr>
<td>Interests</td>
<td>Cricket</td>
<td>Reading, cricket, table tennis</td>
<td>Squash, tennis, photography, books</td>
<td>Gadgets, food</td>
</tr>
<tr>
<td>What do I bring to the collaboration?</td>
<td>Grittily to the project</td>
<td>Follow the standards, not only code, think about usability</td>
<td>Experience in delivering apps, usability understanding for the vertical</td>
<td>Coffee, commercial perspectives</td>
</tr>
<tr>
<td>What do I want from my team mates?</td>
<td>Flexibility, if I need help they are ready to provide it</td>
<td>Follow the practices</td>
<td>Openness, passion for what we do</td>
<td>Dedication, meeting of deadlines, feedback about how things are going, validation.</td>
</tr>
</tbody>
</table>

Figure 23 - Team poster
6.3.1.4 Mission statement and best practice user interfaces

This section describes the design and implementation of the project mission statement and best practices page. The mission statements of the project are the common shared goals that the team would achieve. Best practices are the set of practices defined and enforced by team members for achieving the shared goal. As illustrated in Figure 24 the “Our Mission and Practices” page contains three main panels. The panel on the top shows the current mission statement defined by the team. The second panel presents a table of best practices that the team members enforced and participants who are assigned to the best practices for example as shown in Figure 24 the best practice id 5015 is “We update ScrumDo before Scrum meetings” which is enforced by the all team member whereas the practice id 5016 “We have Daily Scrum meetings and first meeting of each sprint uses planning poker to scope the sprint” is assigned to only Ashhar who ensures that this practice shall be enforced. The sidebar of the page displays the list of participants in the project with profile pictures and full names. This helps to enhance the awareness among distributed members. Moreover, the user is also able to click on a full name link to view the profile information of that particular member in details.

Mission of our team
Revolutionizing mobile payments by creating a seamless user experience in perfect harmony with bank grade security

The practices that our Team shall follow to achieve our mission

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
<th>Enforced By/Compliant</th>
</tr>
</thead>
<tbody>
<tr>
<td>5015</td>
<td>We update ScrumDo before Scrum meetings</td>
<td>Aslam, Mohammad, Jibran, Grufran, Jonatan, Stefan, Ashhar, Altaf, Sufyan</td>
</tr>
<tr>
<td>5016</td>
<td>We have Daily Scrum meetings and first meeting of each sprint uses planning poker to scope the sprint</td>
<td>Ashhar</td>
</tr>
<tr>
<td>5017</td>
<td>We do code reviews before shipment - to ensure that quality of code and coding standards have been followed, process defined as “Friday code reviews” twice a month</td>
<td>Jonatan, Stefan, Altaf, Sufyan</td>
</tr>
<tr>
<td>5018</td>
<td>We have weekly big meetings where all relevant stakeholders meet the full team and talks</td>
<td>Ashhar</td>
</tr>
<tr>
<td>5019</td>
<td>We send out DoM - Decisions of Meetings after each meeting. The DoM is written on shared screen and send it out immediately after the meeting to the full team</td>
<td>Ashhar</td>
</tr>
<tr>
<td>5020</td>
<td>We have solution Walk-throughs, where we show and demo a build every Friday</td>
<td>Ashhar</td>
</tr>
<tr>
<td>5021</td>
<td>We send Release notes to the QA</td>
<td>Aslam, Mohammad, Jibran, Grufran</td>
</tr>
</tbody>
</table>

Figure 24: Project's mission statement and best practices
6.3.1.5 Communication channel and availability user interfaces
This section explains the design and implementation of the “Our Communication” and “Availability” pages. The “Our Communication” page shows the list of locations where the participants of a project are locating. Additionally, different time zones between locations are also visible since the awareness of the different time zones is considered as one of the important factors in a globally dispersed environment. Figure 25 illustrates different time zones between Islamabad and Copenhagen. When a user clicks on the location name e.g., ISB (abbreviation of Islamabad), the ATGSD tool navigates the user to the “Availability” page, which presents available team members if any at that particular time as shown in Figure 26. Similarly, the user can click on CPH (abbreviation of Copenhagen) for viewing available team members reside in Copenhagen as shown in Figure 27.

### Our Communication

<table>
<thead>
<tr>
<th>Time</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPH</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

Figure 25: Different time zones between collaborative locations

### Availability

**Available Team members in ISB**

<table>
<thead>
<tr>
<th>Picture</th>
<th>Fullname</th>
<th>Company</th>
<th>Location</th>
<th>Select</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Avatar" /></td>
<td>Aslam Khan</td>
<td>TEO</td>
<td>ISB</td>
<td>Select</td>
</tr>
<tr>
<td><img src="image2.png" alt="Avatar" /></td>
<td>Muhammad Rizwan</td>
<td>TEO</td>
<td>ISB</td>
<td>Select</td>
</tr>
<tr>
<td><img src="image3.png" alt="Avatar" /></td>
<td>Jibran Masud</td>
<td>TEO</td>
<td>ISB</td>
<td>Select</td>
</tr>
<tr>
<td><img src="image4.png" alt="Avatar" /></td>
<td>Asghar Saeed</td>
<td>TEO</td>
<td>ISB</td>
<td>Select</td>
</tr>
<tr>
<td><img src="image5.png" alt="Avatar" /></td>
<td>Altaf ur Rahman</td>
<td>TEO</td>
<td>ISB</td>
<td>Select</td>
</tr>
<tr>
<td><img src="image6.png" alt="Avatar" /></td>
<td>Sufyan Anees</td>
<td>TEO</td>
<td>ISB</td>
<td>Select</td>
</tr>
</tbody>
</table>

Figure 26 - Available team members in Islamabad
6.3.1.6 Connectivity map user interface
This chapter describes the design and implementation of the Connectivity Map, which belongs to an exercise session in the norming phase of the ATGSD methodology. The purpose of the connectivity map is to keep track of how well distributed team members communicate with each other during a software development process.

As shown in Figure 28, the team members are displayed in a two-dimension table with the first row displaying team members’ name in Copenhagen and the first column displaying distributed team members’ name. The communication status is graded in three levels: good, medium and bad represented in the colors of green, organ and red respectively. Figure 29 illustrates the connectivity map graphical user interface for an administrator who has the privileges to modify the communication status of team members.
In the next section, we are going to describe the graphical user interface design and implementation for administrator to manage the projects artifacts.

6.3.2 User interfaces for administrative users

This section describes the graphical user interface restricted for the administrative users only. When a user has the administrative rights, he or she is able to see the “Administration” panel, which provides mechanism for managing: project, team, participant, role, project’s mission statements and best practices, and collaborative location. The administrative panel is illustrated in Figure 30.

For the nature of the system, the administrators have privileges to create, read, delete and update the project’s artifacts. Hence, the design and implementation of the ATGSD tool and the GUIs shall be able to facilitate the administrators to perform administrative operations on project, team, participant, role, mission statement and best practices, and collaboration and communication channel.
Figure 31 - Creating a new participant and assigning a role

As illustrated in Figure 31, in order to create a new participant in the ATGSD tool, the administrator navigates to the “Administration” menu item in the menu and selects the “Participant Management”. The ATGSD tool presents the “Add a new participant” panel where the administrator enters all the minimum required data for creating the new participant (or user). After successfully creating a participant, the administrator shall assign roles for the participant. The role assignment process is separated from the participant creation process since as defined in the business rules of the system; one participant may have many different roles in one particular project. After successfully creating and assigning roles to the participant, the new participant is displayed in the list of already existing participants in the system. As shown in Figure 31, a participant named Sufyan with the role “Senior iOS developer” has been created. Then the participant shall be able to log in to the ATGSD tool using his email and password for updating his own personal profile data. His profile data shall be available for any participants in the same team.
7 Tool evaluation

This chapter describes the methodology, logistics, and findings from a preliminary evaluation of the tool developed in this research project for supporting team building activities and enhancing awareness among team members in GSD.

7.1 Objectives and methodology
The primary goal of this evaluation was to assess the efficiency and effectiveness of the tool for supporting team building activities and providing awareness about team members in globally distributed settings. For the evaluation, the tool was made available on the Internet by deploying the ATGSD tool on a Microsoft Windows Server 2012 and Microsoft SQL Server 2012. We managed to seek three volunteer participants for this evaluation. Two of them evaluated the tool as administrative users, whereas the third participant assessed the tool as a normal user of the tool. The participants with the administrative privilege belonged to the organization that had developed the ATGSD methodology, whereas the participant with a normal user privilege was from an organization that has been using the ATGSD. All of the participants had at least a master degree. All three participants were located in Denmark. Before the evaluation, the evaluation scripts and the evaluation questions were sent to all the participants. The evaluation scripts were aimed at providing the participants with an overview of the key features of the tool and a set of guidelines about how to use the tool. They were also informed about the tasks to be performed for the evaluation. The evaluation questions were designed for gathering participants’ perception of different features of the tool based on their experience of using the tool compared with the traditional papers based templates. The evaluation questions contain two main sections: the rating scale questions for collecting quantitative data of the participants’ opinions on the usability of the features of the ATGSD tool and the advantages of using the ATGSD tool. The purpose of the open-ended questions was to seek participants’ ideas for improving the developed features and devising potential features for the tool. For the nature of the system, the administrative level users had a complete access compared with the normal users. Therefore, the evaluated tasks of the administrative users involved the management features such as creating a new project, a new team, assigning the team to the project, creating a new participant, adding a participant to a team, defining a mission statement of a project and so on. Whereas, the evaluated tasks for the normal user simply involved modifying own profile and browsing project awareness information such as viewing a team poster, checking available team members, etc. After completing these tasks, the participants were requested to answer a set of questions. The participants’ answers were analyzed for the effectiveness and the usability of the ATGSD tool as well as identifying the potential areas for improving the tool. The participants’ feedback is discussed in the next section.

7.2 Feedbacks
The feedback has been mainly positive even though there were some issues that need to be addressed in a possible future work. The first thing to clarify is that this ATGSD has been developed as a prototype; therefore, it is not suitable for a real world scenario. Taking into account this premise, all participants have found this kind of application useful, since it provides many advantages in comparison with the
traditional papers based templates: As explained by one of the participants: “It is a standalone tool, that can make distributed teams coordinate in an efficient manner without any overhead”.

Moreover, another participant also provided an explanation of the advantages of the ATGSD tool compared with the traditional papers based templates is that it provides a fast and easy approach for accessing team members’ information: “You can quickly get information from your partners on the ATGSD tool in comparison to the traditional paper based templates, which is normally required time to reach them.” and “… in many situations the tool may have the advantage of accessibility, maintainability, cost, peoples disabilities etc.”

As claimed by the developers of the ATGSD methodology, the team poster is one of the most important artifacts of the ATGSD methodology, which helps to enhance awareness among team members. Most of the participants agreed with how the ATGSD tool is presenting the team poster artifact. However, one of the participants also provided a suggestion for making the team poster more attractive to users. As suggested by this participant the presentation of the team poster should be: “… less IT and data centric - more people and team centric is required. Social networks provide good starting points for such tools. In general, information must be editable wherever it is displayed.”

Furthermore, one of the features of the ATGSD tool that attracted the participant interest is the team assignment feature. It helps to enhance the reusability of existing team information so that the administrative users (or the facilitators) were free from re-entering the team members’ information again and again. This feature somehow helps the facilitators to save time in managing the participant information. Additionally, the feature for displaying different time zones and presenting available team members in collaborative sites also helps to enhance awareness among team members. As stated by one of the participants who is a developer of the ATGSD methodology: “most of the developed features of the ATGSD tool are helpful”. However, since the ATGSD tool is at the initiation of the development process, there are still many features that need to be incorporated in the tool to make it more attractive to users for example one of the participants suggested potential features for the ATGSD tool: “…features like real time commenting, sharing and file sharing can be added to the software development” and another participant also suggested that “…in the Project Details page there shall be a field where users can attach a text document”. In the near future, we are going to analyze these requirements carefully to verify whether it is reasonable and feasible to incorporate these features in the ATGSD tool.

One of the issues of the ATGSD tool was the navigation mechanism. It could be explained that at the current stage the ATGSD tool did not implement a flow between phases of the ATGSD methodology (e.g., forming, norming, storming and performing). Also, there was a lack of a mechanism for preventing users navigating from one phase to another if the current phase was not completed. This may cause the ATGSD tool to present an empty page when there is no data available. For example one participant complained: “…nothing is displayed in the communication tab … the visualization of connectivity map of some projects lead to server error”. This mistake could be explained by that the participant was logged in to the ATGSD tool as an administrative user. He/she created a new project but he/she did not assign any team
for the project yet. He/she then clicked on the communication tab for viewing communication status between team members. As a result, there were no available team members to show their communication status. A future improvement of this can be to incorporate a mechanism for preventing users from using the functionalities that are not available at a certain phase e.g., forming, norming, storming and performing.

This evaluation study is lightweight by design. This means that the design of the tasks and questionnaire has been thought out according to the number of available participants. Moreover, the ATGSD tool was deployed on a dedicated server to serve the purpose of evaluating the features of the ATGSD tool. A possible future work will likely include an intensive evaluation involving more participants in a real day work as well as the ATGSD tool will be deployed on a real cloud environment in order to assess whether the capability of the ATGSD tool in compatible with the cloud infrastructure. To support this goal the ATGSD tool must implement the authentication service that is provided by the PTaaS platform.

Finally, an important issue associated with the team building spirit is to incorporate the information about handling of team building event in the project such as providing the team building model and introduction to the users.
8 Conclusion and future work

This paper has presented the design and implementation of the ATGSD tool, which aims to enhance awareness among team members in a globally dispersed environment as well as facilitating the management of distributed teams and project artifacts efficiently. In addition, it has also provided a consolidated access point to the ATGSD tool for software development organizations located around the world. Furthermore, the architecture of the ATGSD was designed for supporting the multi-tenancy characteristic of SaaS applications.

In order to support these goals, we have conducted four interviews with ATGSD methodology developers and two observations with the teams practicing ATGSD methodology. The findings from the study were used for eliciting architectural significant requirements for the ATGSD tool. For enhancing awareness among distributed team members, the ATGSD tool implemented the team poster artifact of the ATGSD methodology, which is one of the most important artifacts of the ATGSD methodology. The team poster presents all relevant information of team members involved in a particular project including information about group members, their status and their roles (group structural awareness[1]) and which project members are currently available and what they are doing, their working hours, expectation of the collaborative work (informal awareness[1]) etc. These pieces of information are important in a distributed team context in order to enhance awareness among team members and can be used to facilitate casual interactions and initiation of appropriate modes of communication [13]. In addition, the ATGSD tool also provides features for displaying project mission statements and best practices in a project. These pieces of information are also important in the globally dispersed environment for enhancing awareness among distributed team members.

The ATGSD tool also supports automatically computing and displaying different time zones as well as available members in the collaborative locations. Furthermore, it also implemented a communication matrix feature, which is used to measure how well the performance of a team is by keeping track of whom is talking to whom and how often. Last but not least the ADTGS assists the facilitators during a team forming session (or/and team changed events) to accomplish their job better in comparison with the traditional papers based templates, since in the traditional papers based templates, facilitators have to manually ask and fulfill the personal information of each participant. With the support of the ATGSD tool, participants are able to access the ATGSD tool and fill in their personal data themselves. The ATGSD methodology contains a second matrix named compliance matrix. This matrix is used to assess how compliant distributed team members are with best practices that are defined as the team practices. Due to the time constraint, the compliance matrix has not been developed as a feature yet. In future, we intend to incorporate it in the ATGSD tool for measuring whether distributed team members are in compliance with their practices as well as having disciplines in the team to carry out high performing jobs.

The ATGSD methodology defines four main phases including forming, norming, storming and performing. The main problem with this is that a team member has no chance to go back to see historical data of the previous phase. Since in a distributed team setting, team members might join and leave the team as frequently. Therefore,
when a new team member joins a particular team he/she has no chance to catch up on what has been decided or discussed in the previous phase e.g., project mission statement, and other important decisions. This can lead to lack of awareness of the project. At the current stage of project, the ATGSD tool only displayed the forming stage as a default stage of a newly created project. We did not implement a navigation mechanism for switching between stages as well as saving the decisions made in previous stage. A future improvement of this issue can be to incorporate a feature in the ATGSD tool for saving historical data e.g., decisions made, project mission statements and best practices etc. Moreover, we will also incorporate a navigation mechanism for switching between phases in the ATGSD methodology e.g., forming, norming, storming and performing. Also, the ATGSD tool shall display the current phase of the project as well as preventing users to go to a later phase if the current phase is not completed. For already completed phases of a project, the historical data shall only be browsed but not be edited.

For supporting multi-tenant characteristic we incorporated TenantID as a parameter of web service functions to uniquely identify tenants. We also implemented a shared database - shared schemas approach in which a relevant table of the database contained a TenantID column for uniquely storing data related to that particular tenant. Moreover, we have also implemented the tenant web service as described in section 5.4.4 for manipulating tenant user data. However the ATGSD tool did not integrate the authentication mechanism provided by the PTaaS platform yet, but uses its own user database. In the future, we plan to incorporate the authentication service of PTaaS platform for authenticating tenant users. Another potential improvement pertaining to multi-tenancy is that we intend to implement a database access module (or a mapping table), which dynamically processes a query statement and redirects tenant users to their private database in case some of the tenants have special requirements for hosting their data in a private database. Currently, the ATGSD users manually insert their profile information. A future improvement of this can be that the ATGSD tool shall support various approaches for inserting user profile data for example the user profile data is automatically retrieved and inserted to the ATGSD tool via open APIs e.g., LinkedIn APIs.

The ATGSD tool is designed mainly for supporting the team-forming session (e.g., create a new team) and team-changed event (e.g., team members join or leave a team). It means that the ATGSD tool are highly demanded at these phases of a project life cycle, and less demanded at the other phases. The future improvement of this is that we seek to answer the following question: How can we improve the sustainability of the ATGSD tool? One of the potential answers can be to incorporate features for supporting self-managing teams with the corresponding teamwork components identified in Dickinson and McIntyre teamwork model [36]. This teamwork model is comprised of a set of teamwork components including communication, team orientation, team leadership, monitoring, feedback, backup and coordination and relationship among them. We will carefully identify the artifacts of each component to seek for possibilities to incorporate these artifacts in the ATGSD tool so that the ATGSD tool will be helpful in supporting a self-managing team during a project life cycle. With these features integrated, the ATGSD potentially helps to improve team performance in a GSD context since much research has also found that team

http://developer.linkedin.com/apis
performance is linked with the effectiveness of teamwork coordination [37] [38] and a software development project depends significantly on team performance.

In the near future we intend to implement the requirements identified in this research project that we due to the time constraint did not implement in the initial version of the ATGSD tool. Finally, we plan to provide the solutions for the issues reported in the evaluation as well as systematically incorporating the implementation of the features suggested by the participants of the evaluation in order to make the ATGSD tool more attractive to the users and fulfill their needs.
9 References


