INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2012, VOL. 58, NO. 4, PP. 433–439
Manuscript received October 25, 2012; revised Decemer, 2012. DOI: 10.2478/v10177-012-0059-6

The Implementation of 3TZ Model of Software Development

Zenon Chaczko, Shahrzad Aslanzadeh, Frank Jiang, and Ryszard Klempous

***Abstract*—This paper presents the concepts and explores issues related to the 3 Time Zones (3TZ) model of software development in global workspace environment. The 3TZ model itself seeks to take advantages of differences in time zones between places around the world. By engaging software development teams in different regions separated by 8 hours each, it is possible for their combined working hours to cover the whole 24 hours period. Thus, while they each work their normal 8 hour days, together they are able to achieve in 1 day what a single team would achieve in 3 days. They are able to achieve this by passing on their work from one team to the next as one finishes their workday and the next team starts their workday. The 3TZ model of software development revolves around the employment of a software development team distributed in at least 3 different locations around the world in 3 different time zones. If work was passed on from one team to the next and adjacent teams were separated by 8 hours, then 24 hours continuous collaborative software development could be achieved. Though this poses many challenges, when dealt with there is great potential for software to be developed much faster than is possible for a single, collocated development team. In the global economy, we have seen a decrease in the barriers towards communication across the globe along with an increase of service availability to support this communication. Software development is one of the disciplines that is capable of effectively utilizing and benefiting from global collaboration prospect lent by ever increasing capability of information and communication technology. 24 hours continuous development is ideal for application towards tasks that have hard-deadlines or require work completed as soon as possible. This article will mainly focus on introducing 24/7 global models that can be applied in cloud environment used in three different time zones.**

***Keywords*—Software development, cloud computing, distributed software services, 3TZ environment.**

I. Background

THE SOFTWARE development methodology based on the 3 Time Zones (3TZ) model intends to explore and evaluate a new collaborative software development processes and tools situated in a global workspace environment.

The idea of the 3TZ model of software development overlaps with existing iterative, incremental, agile frameworks such as Scrum, which includes the concept of day sprints and daily meetings. Instead of teams just having their own location specific daily team meeting, a cross-team meeting

Z. Chaczko and S. Aslanzadeh are with Faculty of Engineering and IT University of Technology (UTS) Sydney, NSW, Australia (e-mails: zenon.chaczko@uts.edu.au; shahrzad.aslanzadeh@uts.edu.au).

F. Jiang is with School of Engineering and IT, University of New South Wales, Sydney, Australia (F.Jiang@adfa.edu.au).

R. Klempous is with Faculty of Electronics, Wrocaw University of Technology, Wroclaw, Poland (ryszard.klempous@pwr.wroc.pl).

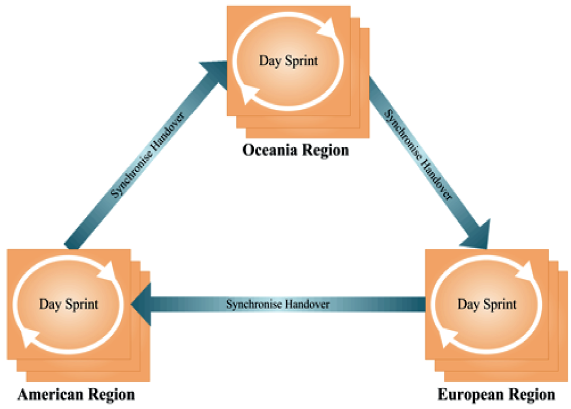

Fig. 1. The 3TZ model of software development.

during an overlap in work time would facilitate a handover-synchronization so that work can smoothly be transitioned from one team to another, with any questions, concerns and problems resolved. This is an example of knowledge transfer in what Gupta [1] describes as a 24 hour knowledge factory. An illustration of how this works is shown in Fig. 1.

This paper highlights software development management process that would facilitate the 3TZ model of software development. Also this paper sought to fill the gap in the available software for the specific purpose of facilitating the implementation of a three time zone (3TZ) model of software development. This gap exists in both the educational and commercial space, though there are many products that could potentially be utilized.

II. Advantages and Disadvantages of 3TZ Model

3TZ is just one mode of software development that has emerged as companies have grown their once single location development team into globally distributed software teams. Carmel [2] identified numerous reasons for the growth of global software teams.

A. Specialised Talent

Software developers are not all equal, all having their own levels of training, experience and expertise. Although it is possible to relocate the needed developers, often it is necessary (or simply more practical and cost effective) to employ them in their home country. Also, for many countries the demand for software developers is simply greater than the labor supply, and companies are forced to look overseas for the developers they require. Herbsleb and Moitra [3] describe this as the need to effectively utilize scarce global resources (software developer labor supply) in a cost-competitive manner.

B. Reduction in Time-to-Market

Although having teams spread across different time zones presents its challenges, it presents an opportunity for development to occur in a 3TZ model as described above and thus for work to be completed quicker than it could be otherwise. This can provide companies with a competitive advantage or reduce development times.

C. Location Transparency

Through the development of information and communication technologies, it is becoming easier for teams to communicate and collaborate. Cloud computing is now dominating this growth, providing global software teams with integrated and uniform software and databases, helping teams to act as one cohesive team. Although this does not completely compensate for the lack of rich face to face communication between teams, it certainly makes global software teams that much more viable [4].

D. Challenges

Despite these reasons and advantages, moving from a single collocated software development team to globally distribute teams comes with many challenges, many of which are exacerbations of those of single collocated teams. The three main differences between local and global teams that give rise to these challenges are distance, time zone and national culture, as identified by [2].

E. Distance

The main impact of distance between development sites is the reduction and constraining of communication. Project managers must rely on information communicated by team managers and less on face to face meetings or informal face to face or telephone conversations. Team members in different site teams find it harder to bond and become cohesive without physical interaction and spontaneous informal communication.

Communication instead becomes dominated by electronic methods which are not only less rich (by lacking non-verbal cues) and so less able to convey information accurately (which is especially important in cultures that rely a lot on context for understanding), but they are also predominantly asynchronous, which in non-western cultures is less acceptable unless solid personal relationships have already been formed.

However, these issues are not entirely absent from relatively closely located teams. The study on communication between colleagues showed that informal communications dropped significantly once offices were separated by 25 meters and to the same extent as if they were separate by several hundred kilometers. Various studies have shown that software departments event placed on different floors in the same building had very little interaction [1], [5]. This is not to say that informal communication does not occur between cross-city offices, since meetings and phone calls are still possible. But these studies do show that offices do not have to be separated by hundreds of kilometers before informal communication becomes far less common.

F. Time Zone

Time zone differences exacerbate these communication issues by preventing synchronous communication during normal working hours. If non-asynchronous communication is necessary, one member or team has to compromise by conducting a meeting outside of their working hours, or if possible, only in the short period when working hours overlap.

G. National Culture

Despite English being the predominant common language of computing, developers from different cultures display different behaviors, norms, and assumptions. These differences can often lead to mistrust, miscommunication and lack of cohesion. This can complicate understanding the root cause of disagreements between teams, which could stem from differences between national culture, organizational culture or simply professional opinion. Also, the increased diversity can make consensus and decision making take longer. However, as mentioned above, differences in national culture also present an opportunity for greater productivity through different ways of thinking about challenges including ways to solve a problem, how to design a product or how the development process should work. Gupta et al. describes this as having a "global mind-set, unconstrained by single country concerns or cultural factors" [1]. In addition, some surveys have found that the computer subculture is so dominant that there are little differences between developers from different cultures in regards to work being a common personal motivation, placing the most importance on technical values (over social or political values), and the use of similar tools [2].

III. Enabling and Inhibiting Factors of Cloud Computing for Business

A. Availability and SLA

Cichon et al. in their paper 'Worldwide Teams in Software Development' [4] propose an "adaptive development methodology" to address the issues faced by 3TZ development and the fast changing business requirements. The authors [4] suggest the use of 'adaptive contracts' between the development company and the client that focus on the non-functional requirements of scope, cost and time, while functional requirements are developed during the implementation phase. This is to ensure the company can keep up with the rapid changes in the client's needs but also addresses the limited time available in projects that 3TZ development usually involves. This process is well illustrated in Fig. 2. Cichon et al [4] note that this model has been successful in projects undertaken by the software company SAP as well as different commercial projects that the authors have been involved in. Several features of the proposed model are described in the following paragraphs. It is essential for managers and leaders to keep team members up to date with the project vision and their roles in the project

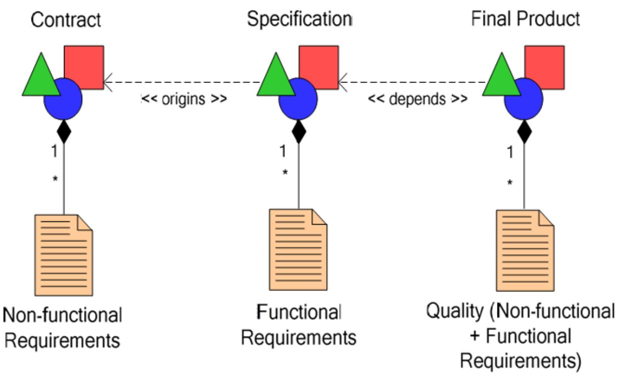

Fig. 2. Requirements in the context of an adaptive project life cycle (SDLC).

such that they do not lose sight of the bigger picture of what might seem like a massive project. It also helps to make their own decisions when they may not always have access to team leaders or consultants. This includes being regularly informed of the architectural system overview, project schedule, critical dates, task priorities and the reasons behind them.

IV. Case Study: Development of Software Engineering Tools Supporting 3TZ Model

The model of 3TZ adopted for this case study was based upon experiences of team members who were originally involved in defining the 3TZ model and who then made tactical adjustments to overcome the challenges related to the new methodology [5], [6]. The model generally involves a process, which starts with a client approaching the 3TZ company with a project that requires the expertise, resources, speed, or scale that the 3TZ company can offer. The company then assigns a project manager, consultants, site team leaders and developers for each team, if not simply using existing teams. The project manager (PM) and consultants work with the client to develop an adaptive contract that covers all non-functional requirements such as scope, cost, time and resources. It is adaptive because the functional requirements are not 100% set, and allows for changes where necessary. They also develop a functional breakdown structure (FBS) instead of a work breakdown structure (WBS) since that they are easier to break down and manage for large projects. Next, all site teams gather for an all-in worldwide team meeting, where they get to know each other as well as become familiar with the project. The PM, consultants and client identify the core functionality they want to be developed first, and assign these to the first 2 iterations. These first 2 iterations are completed by the site team leaders and most experienced developers who stay at the client site. Once the client has evaluated this prototype, they provide feedbacks and work with the PM and consultants to choose the functionalities to be developed in the next iterations. The site team leaders and experienced developers return to their teams, and share with them the experiences and knowledge of the project that they have. The site teams then start the cycle of working on functionalities allocated by the site team manager, writing daily closing reports of the work they have done, holding handover meetings with the next team via video conference during the overlapping period in working hours and answering any questions, before the next team carries on the work. This cycle continues until iteration is getting completed (typically a week), when the client reviews the work and again helps to decide the functionalities to be completed for the next iteration. This continues until the iterations are completed and the FBS is completed or the adaptive contract is fulfilled [5].

A. Project Implementation

The actual implementation involved the addition of new modules to or modification of existing ones in the existing open source Endeavour program, with modules being individual sections of functionality. There were 2 main new modules added: Daily Closing Reports (DCR) and Sites.

The DCR module was added as a high priority due to its significant role in the 3TZ model by facilitating formal knowledge transfer in the handover of development information from one team to the next. Any additional information can be provided in the comments tab, which can also be added to by other team members. The Sites module is an example of an introduced requirement, as it was not originally part of the requirements list. The need for this module only became apparent during a feedback session with the client. It essentially allows the different sites of the component worldwide teams to be included in the database. This means that as individual team members are added or moved, their site information, such as country, location and time zone would automatically be updated. There were several existing modules that were modified to help the program conform to the 3TZ model of development. These included Users, Documents, the Main Page, Security Groups and some cosmetic changes. The Users module was changed to include information from the Site module, depending on which existing site was selected for each user as a drop down box. Fields such as country, location and time zone were shown in the Users list page and showed as uneditable fields in the user details edit form. This form also had the contact number field added to facilitate communication between team members of the same site team or those of different site teams during the overlap in working hour's period. A category field was added to the Documents module, to aid in the searching and sorting of documents. This was an alternative to the original idea of having different document sections or folders for different items such as client information, high level design and project information. The Main Page was slightly modified to show the current site of the logged in user, and a slight rearrangement of module sections to accommodate the added modules. The Security Groups module was simply changed to add controls for the new DCR and Sites modules. The other minor changes involved removing the Endeavour logos and titles and replacing them with the CCD: Continuous Collaborative Development title in order to distinguish it from the original Endeavour program.

B. Design

Since this was a project involving the modification of an existing open source program rather than developing a whole

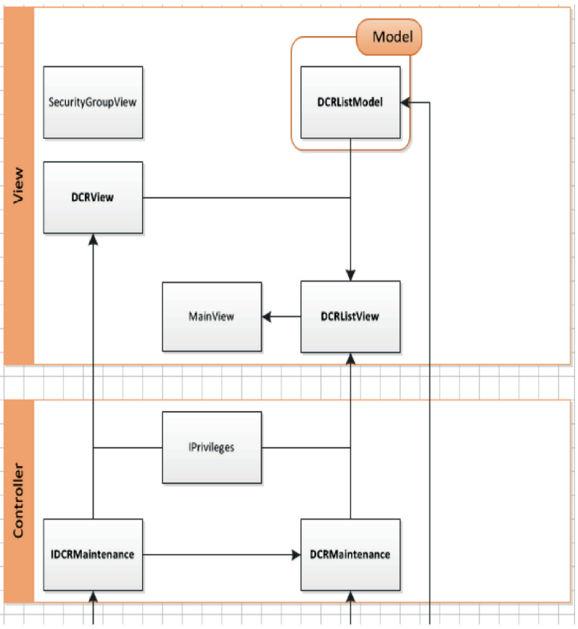

Fig. 3. Figure 3-3TZ designing concepts. Source: [6].

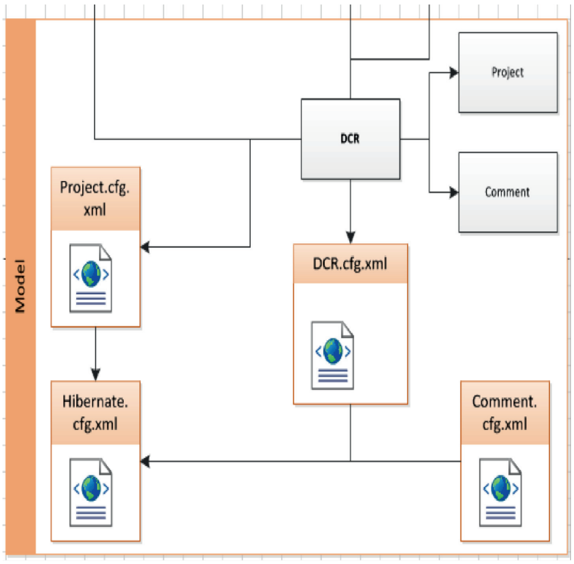

Fig. 4. DCR in endeavour with changed files. Source: [6].

new one, the design basically involved understanding that of the Endeavour program and understanding where to implement the modules that were to be added to satisfy the requirements.

The diagrams depicted in Fig. 3 and Fig. 4 illustrate how this new module fit in with the other classes of the Endeavour program. The first diagram (Fig. 3) shows, which classes had to be edited in order to implement the new module, and the second diagram shows all the classes that the new module interacted with, or used. Currently, the adaptation and software system development related to 3TZ is still continued, and the open source software project management tool called Endeavour Agile ALM has been chosen for this work.

Endeavour is an Open Source solution, designed to manage the creation of large-scale enterprise systems in an iterative

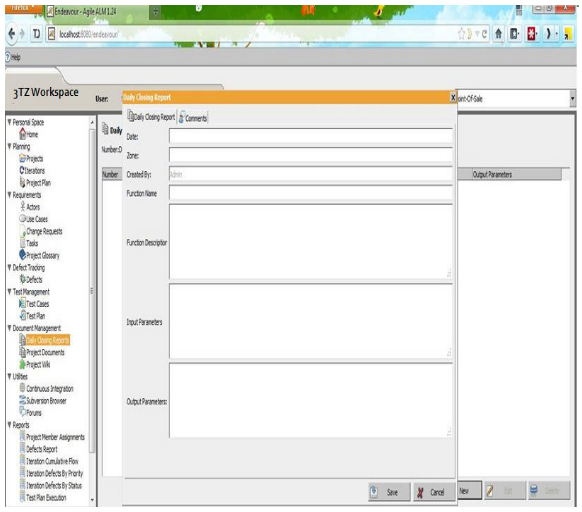

Fig. 5. Daily closing report function of the 3TZ workspace project management tool based on OSS endeavour framework. Source: [6].

and incremental agile software development process. This original OSS framework encapsulates many features required to support various tasks in software analysis and design. It is written in Java, and thanks to its open Model-View-Controller architecture it can be easily ported, adapted and extended to include advanced features of the 3TZ model (See Fig. 5). The extended version of Endeavour can also be easily deployed in Cloud such as ODVL at UTS or any other open source, community or commercially oriented environment.

C. *Application to the Model*

The above implementations, although they do not provide a full-fledged 3TZ model program, they do make advances toward one and help to incorporate some important elements of the model into the program. As mentioned above, the DCR module enables the developers to pass on knowledge of their work to the next team whether that be in real time or not, facilitating continuous collaborative development, and provides formal documentation that can be easily referred to in the future by any team member. The Sites and modified Users modules and Main Page help to make it clear that all team members are members of site teams that fit into a larger global development team. The modified Documents section will aid in the searching and sorting of the great variety of documents that are involved in any large software development project, and the cosmetic changes simply help to reinforce that this is a program designed for following the 3TZ model.

V. Case Study: Development of Mobile Phone Software Applications Using 3TZ Model

Since mobile phone inception in 1973, development of software applications for their effective use has grown rapidly. Initially, first generation mobile phones were only used for telephone calls; however, the second generation mobile phones introduced in the 1990s allowed users to send text messages (otherwise known as SMSs) to each other. At present, the third generation mobile phones have evolved to a point in which they can be called smart phones and can be compared to having a small PC in our pockets. The modern version of smart phones allows for browsing the web, reading emails, running applications and even creating our own software applications. For a long period, there was little application development on smart home infrastructure front; however, in recent times it has started to change. Home devices leading the way are smart televisions able to stream media from the internet, read emails and surf the web. These television sets are also now capable of being controlled by a smart phone.

In the past, there were several application areas and projects for which the 3TZ collaborative model was tested [4], [5], [7]. However, it was found, that the development of various mobile phones applications seems to be particularly well suited for validating the effectiveness of the methodology when working on various student projects. This case study demonstrates how students, over a course of a single semester, could develop a proof of concept (POC) application [8] while working in 3TZ mode. The POC under our study aims to address the gap between the continued ongoing developments of mobile phones vs. the minimal advance of centralised control of smart home infrastructure. Controlling an air conditioner via SMS however, requires the air conditioner to have a receiving mobile phone, which can be seen as an ongoing expense. Turning on and off AC devices in the home via an iPhone however no feedback is available to determine if the device is actually on or not. The two main smart phone technologies currently available based on market share are: the Apple iPhone and Android phones. Other smart phones include Windows Mobile and Blackberry. We needed to determine which type of phone to use to develop the POC. Apple's iPhone development environment is licensed costing $99 AUD per annum and is required to be run on an Apple Mac. The Android development environment is free and can run on any PC (Windows or Mac based). Applications developed on the iPhone require a payment to make them commercially available on iTunes, whereas for Android Applications there is no cost. Based on these points it was decided an Android phone would be used. Android is a base operating system for mobile phones founded in 2003. Google acquired Android in 2005 making it a wholly owned subsidiary. Android is powered by a Linux kernel and available under a practically free software license; 3.1 Subject to the terms of this License Agreement, Google grants you a limited, worldwide, royalty- free, non- assignable and non-exclusive license to use the SDK solely to develop applications to run on the Android platform. Handsets are developed by the Open Handset Alliance which includes companies such as Intel, LG, Motorola and Samsung. The goal of this alliance is to develop open standards for mobile devices as per the Android operating system, which they run. Android applications are available on the Android Market web site. As of Dec 2010 there were 200,000 applications, games and widgets available to be downloaded and in April 2011 Google Inc. stated that from Android Market three billion apps had been downloaded and installed. Users can also develop their own applications and install them on their own Android phone. There are also no restrictions on developers setting up their own online stores to sell their developed Android applications. The physical Android phone chosen for this project was a Motorola Droid due to its high performance and it also comes pack-aged with the development environment needed.

Fig. 6. WSN control menu options. Source: [8].

A. Development Environment

Various environments are available to develop Android applications. Eclipse is a popular development environment for Java, which Android applications are written in. The Eclipse environment can be used by included the required Android SDK. The Motorola development environment MotoDev comes packaged with the Motorola Droid phone and includes the specific SDKs to set up an Android Virtual Device to mimic the Droid phone for debugging and testing. The MotoDev environment was used for the required Android development. Arduino microcontroller development is completed using the Arduino Alpha development environment. A simple and user friendly environment allows code to be written, saved, compiled and uploaded to the different types of Arduino boards. The development environment additionally includes a Serial Monitor which is useful in communicating with the Arduino BT Board to ensure the program was working as expected prior to testing it with the Android phone.

B. WSN Control Application

The application created was WSN Control (Fig. 6). This application provided the GUI and simulated control of home infrastructure. Initial development was the control of an air conditioner. The application contained three basic views/controls:

- On/Off radio buttons to turn the air conditioner on/off.
- A slide bar to set the desired temperature between 18 to 30 degrees.
- A text view showing the current temperature in the home.

The current temperature was set to a default of 23 degrees. The slide bar could be moved to select the desired temperature however the temperature would not change until the air conditioner was turned on. Once the air conditioner was turned on the program simulated the air conditioner by

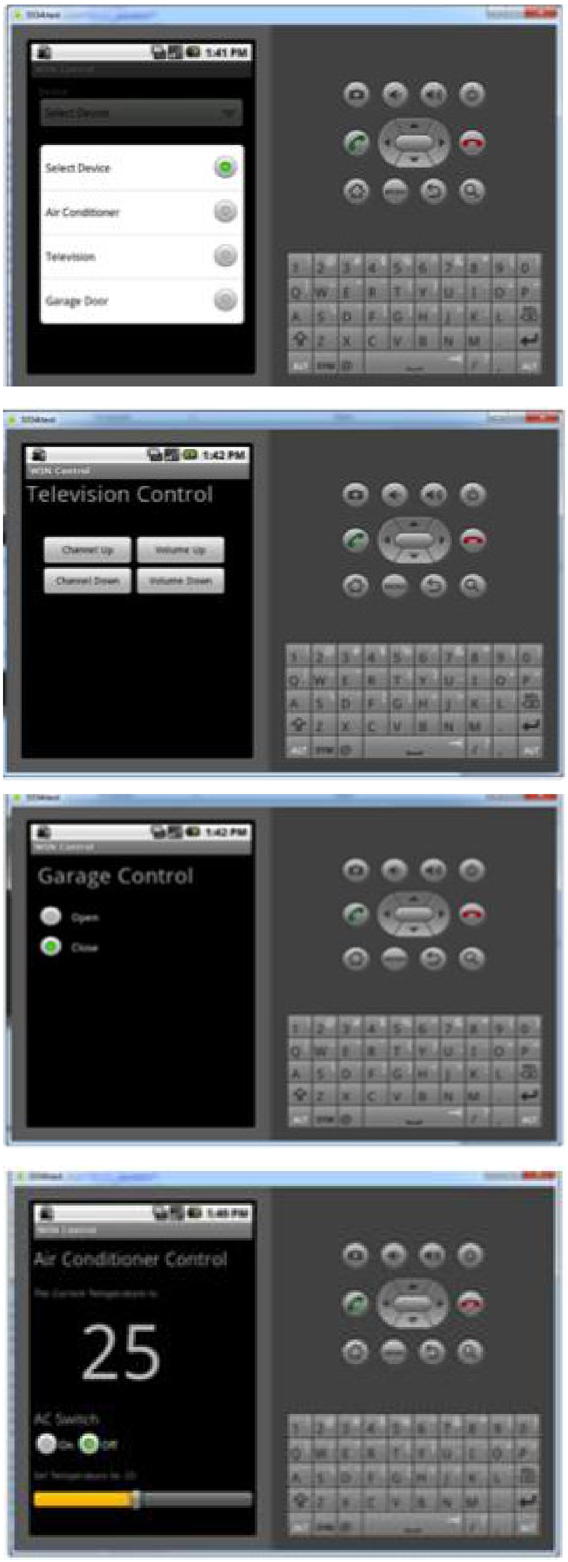

Fig. 7. Devices selection menu options. Source: [8].

adjusting the current temperature by 1 degree every minute until it reached the desired selected temperature. For example, if the temperature was 23 degrees and the air conditioner on the application was set to 20 degrees, it would simulate the temperature going down by 1 degree per minute and adjust the current temperature display. After 3 minutes, the current temperature and air conditioner set temperature would match.

For the current temperature there would be a temperature sensor on the RCK sending data back to the Android mobile when re-quested i.e. once per minute while the air conditioner was on. In order to control other devices in the home more screen real estate on the Android phone was needed, or a separate t screen could be used for each device.

A different screen for each device was designed, although an application menu was required to allow users to select which device they wanted to control needed to be set up. The application menu contained three options. Each of the listed below options had a unique icon to make it easily distinguishable and user friendly:

- Devices: Presented the user a spinner to select which device they wanted to control. Once selected it would move to the relevant screen to enable control of the device.
- Settings: This menu item was not developed but intention was to enable various options such as adding new devices to be controlled, modifying or removing existing devices; and scheduling control of devices at certain times.
- Quit: This option enables the user to close the application.

A title screen was added so it would not start up by default controlling the air conditioner. For the Quit menu button to ensure the application did not close if the user accidentally selected it, a popup dialog was added to confirm the user wanted to exit the application. For the devices menu button a spinner was used, which only allows one item from the menu to be selected. Once selected the screen would change to the appropriate device screen. A mock up screen was created for the television and garage along with the existing air conditioner screen. These screens were created as they enabled simulation of both transmission methods IR for the air conditioner and television, and RF for the garage. For the television screen two text views were initially added and subsequently removed, showing the current channel and volume.

The issue was there was no way to get data from the television. The alternative would be to set the volume and channel to a default value each time the television was turned on, however it would not be a very good user experience. An- other option would be to include a keypad with digits to allow the user to select the channel they want instead of having to scroll through channels to their desired channel. To determine the state of the television being on or off, a suitably placed light sensor could be used, or alternatively a heat sensor placed behind the television, as most of television sets produce a significant amount of heat when on. For the Garage Door it was important for it to have a feedback loop to determine its state. The reason for this was that if the user was away from their home and wanted to make sure the garage was closed, they could easily check this via their Android phone. A light sensor could have been used; however, it is likely it would not work at night. A pressure pad resistor changes its resistance once pressure is applied. This could be placed under the garage door to determine if it was closed, hence applying pressure to the resistor. An alternative could be to use a micro switch or reed switch to determine if the garage door is opened, these are commonly used in alarms. Once the application was developed and tested on the PC using a Virtual Android Environment, it was packaged up and tested on the physical Android phone.

VI. Conclusion

It is apparent that there are different views of global software development and 3TZ method in particular. It was generally noted, that global software development poses greater challenges and more opportunities than locally based development teams. Even more so is this the case with 3TZ. While it is possible to accomplish more in the same amount of time with globally distributed teams that together work 24 hours a day, the complexity of large team sizes, the lack of informal and nonverbal communication, and cultural differences all add up to prevent all the potential that 3TZ has to offer. However, strategies and tactics have been developed to address these issues. Although Endeavour does not include all of the functions that some propriety software does that could facilitate the 3TZ model, it certainly has the potential for doing so. This comes both in the form of third party applications that can be included as open source code or services that are incorporated through application programming interfaces (APIs), as well as more experienced and knowledgeable software developers. Although it may not meet the standards required of larger corporate organizations, it could certainly provide potential usage to the academic community and smaller businesses.

Acknowledgments

I would like to express my gratitude to Ali Rey and Kevin Lim, whose hard work on their projects, expertise, and understanding, added considerably to this work.

References

[1] A. Gupta, S. Sesbasai, I. Crk, and D. B. Smith, "Toward the 24-hour knowledge factory in software development," in *Stealing Time: Explorations in 24/7 Software Engineering Development*, Z. Chaczko, R. Klempous, and J. Nikodem, Eds. Denmark: River Publishers, 2010, pp. 31–62.

[2] E. Carmel, *Global Software Teams: Collaborating Across Borders and Time Zones*. Upper Saddle River, New Jersey: Prentice Hall, 1999.

[3] J. D. Herbsleb and D. Moitra, "Global software development," *IEEE Software*, vol. 18, pp. 16–20, 2001.

[4] P. Cichon, Z. Huzar, Z. Mazur, and A. Mroz, "Worldwide teams in software development," in *Stealing Time: Explorations in 24/7 Software Engineering Development*, Z. Chaczko, R. Klempous, and J. Nikodem, Eds. Denmark: River Publishers, 2010, pp. 85–109.

[5] Z. Chaczko, R. Klempous, and J. Nikodem, *Stealing Time: Explorations in 24/7 Software Engineering Development*. River Publishers, 2010.

[6] S. A. Lim and Z. Chaczko, *Cloud Computing and Its Enablement of the # Time Zone Workflow Model in an Open Source Application Lifecycle Management Program*. UTS, 2012, Supervison by Z. Chaczko.

[7] R. Klempous, J. Nikodem, and A. Wytyczak-Partyka, "Application of simulation techniques in a virtual laparoscopic laboratory," in *Computer Aided Systems Theory – EUROCAST 2011 – 13th International Conference*, Las Palmas de Gran Canaria, Spain, February 6-11 2011, Revised Selected Papers, Part II; 01/2011.

[8] A. Rey, *Android Phone Control of Smart Home Infrastructure, Capstone Project*. UTS, 2012, Supervison by Z. Chaczko.

[9] R. D. Battin, R. Crocker, J. Kreidler, and K. Subramanian, "Leveraging resources in global software development," *IEEE Software*, vol. 18, pp. 70–77, 2001.