Next Generation Event Scheduling System

Honors Thesis

Feburary 2014

Jeremy Cantor

Dr. Saeed Rajput, Advisor

Nova Southeastern University

Farquhar College of Arts and Sciences

Undergraduate Honors Program

Division of Math Science and Technology

Undergraduate Honors Thesis

of

Jeremy Austin Cantor

Next Generation Event Scheduling System

Approved:

Honors Thesis Committee

Dr. Don Rosenblum, Dean, Farquhar College of Arts and Sciences

---

Dr. Matthew He, Director, Division of Math, Science, and Technology

---

Dr. Saeed Rajput, Thesis Advisor, Assistant Director, Division of Math, Science, and Technology

---

Jeremy Austin Cantor, Undergraduate Honors Research Student

Nova Southeastern University

March 2014

Preface

Prior to enrolling at Nova Southeastern University and completing this thesis, I developed a strong interest in software engineering and other various aspects of computer science. I was informed at a Nova Southeastern Honors Program seminar that research and extending beyond what is required in the classroom can enrich your learning experience.

Initially, I did not know what my research would actually involve, but after completing an extensive software project in the summer semester of my freshman year, I decided to elaborate upon it. Dr. Saeed Rajput, who would later become my adviser for this project, offered significant assistance toward improving upon this project from his prior experience in software engineering. Through constant improvement upon this software and thesis, I have developed a sincere appreciation of the amount effort devoted to designing a software application. Furthermore, I genuinely believe that the completion of this project will provide me with an advantage once at the conclusion of my academic career and when I seek employment. This project is a significant addition to my expanding portfolio and has taught me the thought process involved in developing a large-scale enterprise application.

Summary

Current business practices involved in event scheduling systems are inefficient. Traditional business practices, such as employing an office assistance to manually record event times often requires the customer to languish needlessly on the telephone while waiting to receive assistance and to repeat that process several times to establish just one appointment or a meeting. This manual process is subject to human errors that may introduce significant inefficiencies in typical business processes. Use of recent software scheduling tools reduce errors, yet they are still modeled after traditional processes. Their use is often restricted to systems that support a single platform and often only support collaboration between individuals from a single organization. These single-user systems lack adequate features to address issues introduced during the event scheduling to be coordinated by multiple parties. The main source of inefficiency is caused by fact that the single party running the software has complete control over how the event will be scheduled and its duration. All other participants can only acknowledge or deny if they will be attending the scheduled event. If one party is not able to agree with the proposed time, the process has to be reinitiated from scratch. Therefore, several iterations are sometimes needed to schedule a single meeting that has several participants. An ideal scheduling system would provide control into the hands of all participating parties and allow them to use their own favorite personal calendar apps while making the task of finding a common ground easy for all participants.

Using the concepts already familiar to diplomats and used in computer networks, in this research, we have defined a methodology of specifying scheduling as a “protocol”. We have proposed a new scheduling system that allows participants more flexibility in the scheduling process. This system uses a new scheduling protocol that is much easier to use with the aid of modern technologies. Accompanied with the proposed scheduled system is a prototype to demonstrate the viability of such

a protocol using modern software analysis, specifications and development methodologies. The proposed system enables participants to interact with each other more conveniently during the scheduling process and achieve an agreement which much higher likelihood in a single iteration.

Acknowledgments

This Divisional Honors Research Project was made possible through the mentoring Dr. Rajput and assistance of several of my colleagues that I have forged strong relationship with during my college career. My deepest gratitude goes to my adviser, Dr. Saeed Rajput, who provided me with great insight into different aspect of software engineering that I have applied into this project. I am also very grateful to of my closest friend and colleagues**,** Thomas Sylvester and Jonathan Reyna. Both Thomas and Jonathan assisted with software requirement for this project. Thomas, who I considered as a senior and mentor prior to college, also assisted with me with certain parts of the software implementation. Jonathan, who I had the pleasure of meeting in my freshmen year, provided me with insight on computer hardware from his prior experience in industry. Finally, I would like to thank my family and college for giving me opportunity to conducting such research and providing me with the resources needed to complete this project.

Table of Contents

List of Figures

Next Generation Event Scheduling System

I. Introduction

There is an assortment of scheduling tools and systems that are currently being utilized by a variety of different organizations. The majority of scheduling software systems typically cater to personal use with limited support for multiple-user interaction. Google Calendar and Microsoft Outlook are the most popular examples [1]. Google Calendar and Microsoft Outlook both support a simplistic scheduling process that is composed of two-transactions [2]. A user who initiates an appointment request must specify the time along with the relevant details of the appointment. First transaction is completed once this user submits their request to participating users which we refer to as the appointment receivers. In the next transaction, each of the appointment receivers is restricted to confirming his/her attendance at the appointment by choosing from options, “yes”, “no”, or “maybe”. If any of the receivers opts “no”, the initiator has to reinitiate the process.

On the other end of the spectrum, commercial-grade software products such as Time Trade, cater to large-scale organizations. In contrast to its small-scale competitors, Time Trade produces a slightly better interaction in their scheduling protocol between the appointment requester and the requested attendee(s). The requested attendee(s) have limited authority over appointments by having the ability to reject it or request alternative appointment times [3].

Observing this interaction between the appointment requester and the requested attendee(s) is a crucial component in defining a process that will promote fluid communication in the scheduling process. Both appointment requester and the requested attendee(s) should be able to express all their preferences during each transaction. Furthermore, the preferences of both parties should be displayed in a user-friendly manner in which the appointment requester and the requested attendee(s) can easily compromise on an appointment with minimal transactions.

The subsequent section II describes our proposed methodology of describing the scheduling process as a protocol by illustrating traditional scheduling processes as message sequence charts. Section III, introduces our proposed methods and elaborates on how our proposed method addresses the inefficiencies that can occur through the use of traditional scheduling processes. Section IV, independent from the previous sections presents the high level implementation of the software prototype through the use of UML (Unified Modeling Language) and database representation through Entity-Relationship model diagrams.

II. Scheduling Process As A Protocol

The goal of this project is to establish a scheduling procedure that would minimize the amount of communications or transactions between both parties. A reduced amount of transactions will translate into a more fluid and less time intensive scheduling process.

Prior to explaining our innovative scheduling process proposed by this paper, it is necessary to understand traditional schedule processes that occur over the telephone. Furthermore, to truly understand a scheduling process, it must be formalized as a protocol, or a set of procedures that define a communication between multiple *participants*. In the case of scheduling, two types of *participants* are involved. They are denoted as the *requester* and the *receiver*. The requester is the individual who initiates the appointment scheduling protocol by requesting an appointment with a single or multiple *receivers*. The request containing the suggested times and details concerning the appointment will be referred to as the *appointment proposal*. The appointment proposal is sent to one or more receivers through *request transactions*. Each receiver responds to the appointment proposal transaction by a *response transaction*. Both types of transactions can be sent as computer messages between systems, as telephone communication/messages or an e-mails. An *iteration* consists of one request transactions and a response transaction from possibly every receiver originally targeted in the appointment proposal. In traditional systems this compromise often includes dropping or ignoring one or more of the participants, or their voluntary withdrawal in the interest of carrying the business forward. Once a compromise has been achieved a confirmation transaction is sent out to all participants.

We will express various scheduling protocols using enhanced use case notation commonly used to express protocols in computer networking literature [4]. In a use case diagram the participants (known as actors) and systems are represented by vertical lines and the interactions between them are indicated by horizontal arrows. The direction of the arrows corresponds to the direction of the interaction. Any activity that is conducted by the actors or systems in-between interactions is indicated by a narrow rectangle drawn on the vertical lines. The time generally moves forward as we move down the vertical line that corresponds to the actor or a system. In our expression of the scheduling protocol, there are two types of actors; a requester and the receiver. Slanting dotted arrows are used to indicate transactions that require phone calls and voice messages, as they inherently introduce significant delays. The roles and components of the protocol are defined here are depicted in Figure 1. The two dotted lines indicate the phone based request transactions. The use of slanted solid line indicate an electronic message such as an e-mail which is more efficient than a phone message but still involves manual processing by the participants. In Figure 1, the first receiver is shown as responding to the phone request transaction by an e-mail. Finally, a fully automated computer-to-computer message is indicated by a horizontal solid line as there are no delays involved due to manual processing. Figure 1 shows that a confirmation was sent out automatically once all responses were collected.

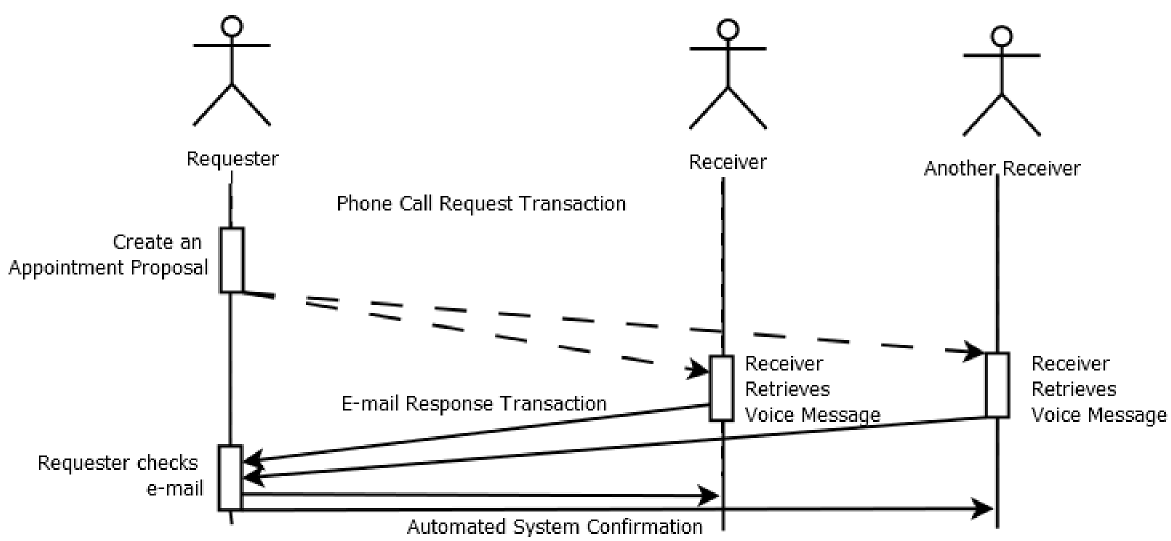

Figure 1 Depiction of key components of the Protocol Notation.

a. Redefining Traditional Scheduling Method as a Protocol

Arguably, traditional business scheduling practices may contain half a dozen phone or e-mail based transactions before an appointment is finalized. Traditional practice in many businesses is to hire a secretary to maintain the schedule and book appointments on the behalf of other employees.

To initiate an appointment protocol via telephone, the requester would either have to contact the secretary or receiver directly. It is assumed that there is a possibility that the requester may be unable to communicate directly to the person they desire to schedule the appointment with. Therefore, a secretary's responsibility is to communicate on behalf of the employees or communication is recorded by an answering service. In both of these scenarios, there can be significant delays. In case of the recording service, it is assumed that the receiver is unable to respond immediately. Furthermore, there can be other messages that are queued in the recording service. In the case of the secretary, there is a possibility that the secretary would have to consult the actual receiver to inquire if they can accommodate the appointment request. In both of these cases, another transaction has to occur so that receiver can call back the requester with an available opening. Similarly, this communication to suggest available openings is not assumed to be instantaneous. The requester may not be available at the moment that the receiver calls back to confirm or reject the openings.

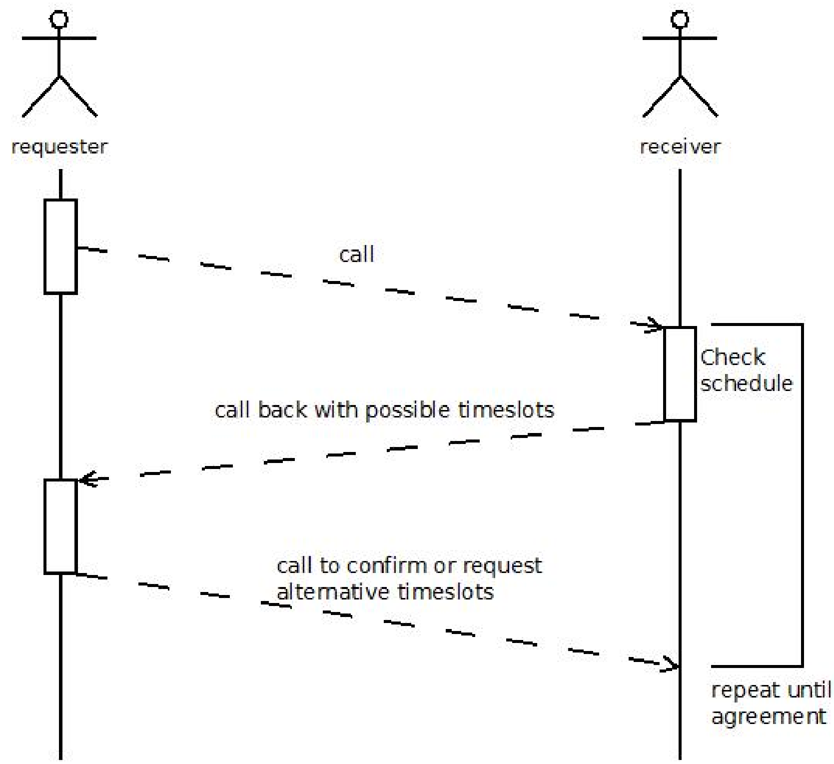

Figure 2 Message Sequence chart depiction of an appointment scheduled by phone.

Fig. 1. Message Sequence chart depection of an appointment scheduled by phone.

In the most complex scenario, once the receiver is in communication with the requester, the requester may not be satisfied with the available timeslots and may suggest alternative timeslots. If the secretary is communicating on the behalf of the receiver, he/she must communicate the alternatives with receiver. The receiver can either accept the alternative or can increase the complexity by suggesting another opening. In return, another communication has to be initiated by the receiver to convey this information for the possibility of a compromise. This protocol scenario is depicted in Figure 2.

It is important to note that there are software tools that supposedly assist with the scheduling process. After defining this protocol as shown in Figure 3, it doesn't deviate too much from the traditional schedule protocol that involves phone calls and messages. Similar to the phone protocol, a user request an appointment, and awaits a response from the receiver. The receiver then responds with one of the following options: “yes”, “no”, or “maybe”. In the case of a no or maybe, the requester may resort to calling or e-mailing the receiver to discuss a possible compromise.

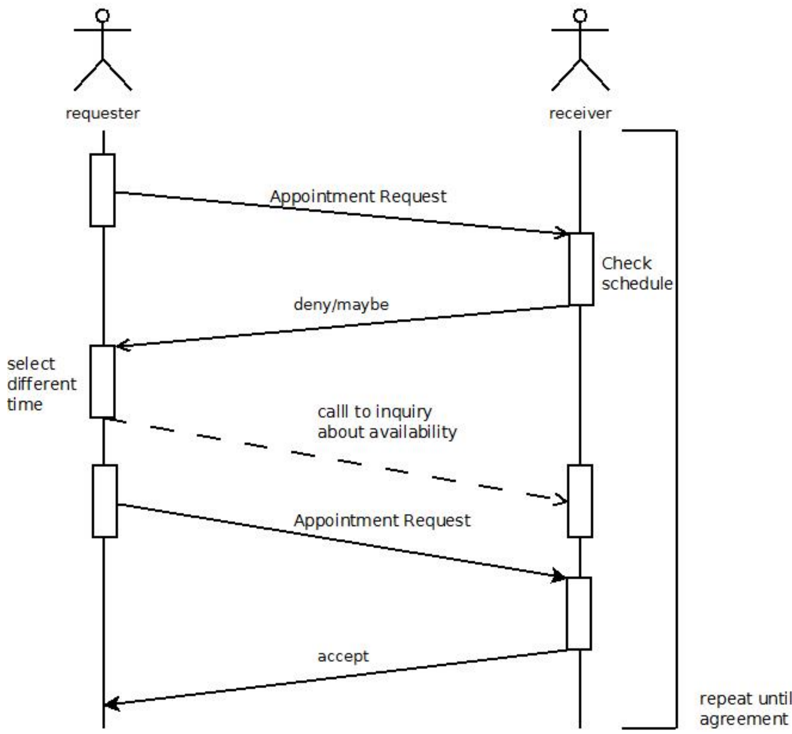

Figure 3 Message Sequence chart depiction of an appointment scheduled with assistance of software tools

III. Description of Our Scheduling Protocol

The scheduling protocol defined in this paper takes into consideration the possible redundancies and delays that can occur during traditional scheduling processes. Minimizing redundancy and delays can be achieved through reducing the amount of transactions involved in successfully scheduling an appointment. The initial step of our defined scheduling protocol is to have the requester specify the receivers. Once the appointment receivers are specified, the system displays the union of unavailable times for all of the participants whose calendars are accessible to the system. This feature addresses the issue of the requester lacking insight to the receivers' schedule. Notice in Figure 2, the requester has to contact the receiver to obtain their schedule. In the case of multiple receivers, this feature becomes increasing beneficial since the user can superimpose all of the receivers' schedules to find possible compromises. The requester can initiate an appointment proposal in a single transaction. In comparison, a conventional system requester would have to contact each receiver prior to suggesting possible appointment times.

Once the requester sees the union of the unavailable times in the same calendar interface, he/she can select or drag blocks that will represent multiple suggested appointment times. Along with the specifying of potential appointment times the requester can specify details about the appointment such as location, topics of discussion (description), and if the appointment will reoccur weekly or monthly. With that information completed, the suggested appointment proposal along with the information is sent to the receiver(s).

In the example shown in Figure 4, depicts the case where the appointment requester has generated suggested appointment times for a single appointment receiver Bob. Bob's unavailable/occupied times are represented by the purple shade. Whereas the appointment proposal is shown in light blue and shows three alternative time slots.

Appointment Requester Alice

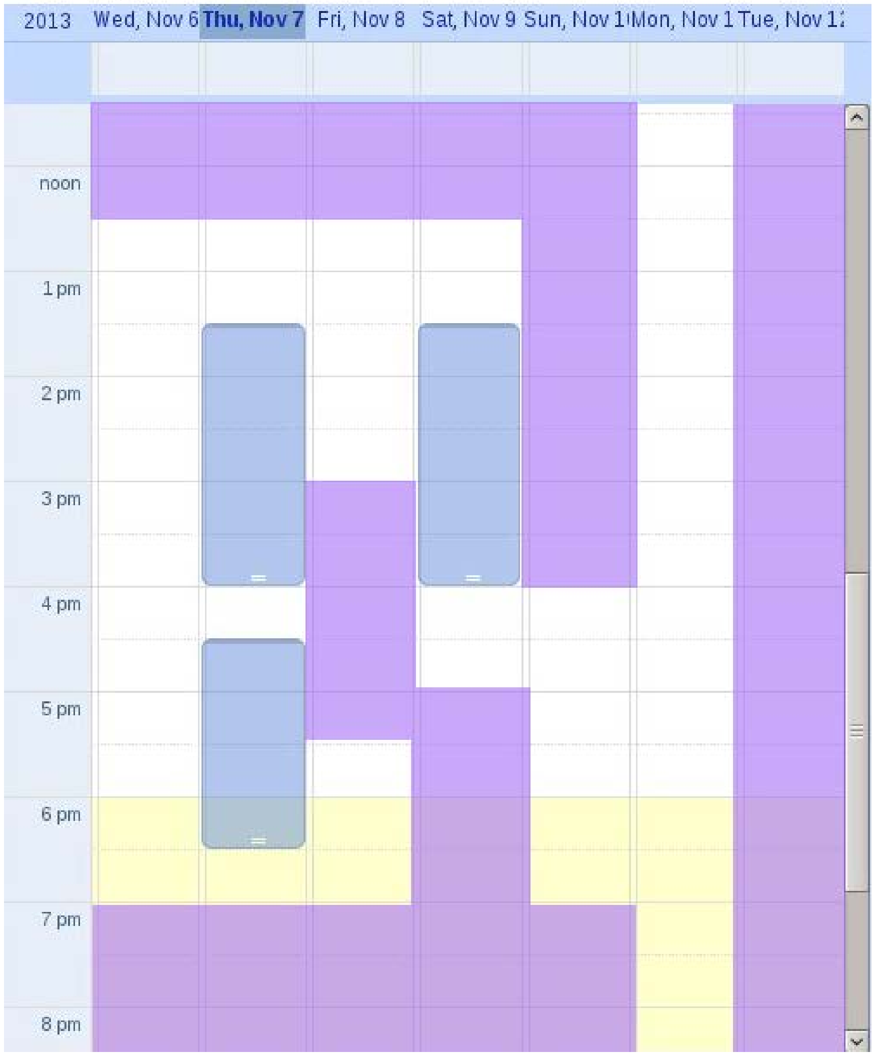

Figure 4 Depiction of unavailable times of a single receiver and the appointment proposal

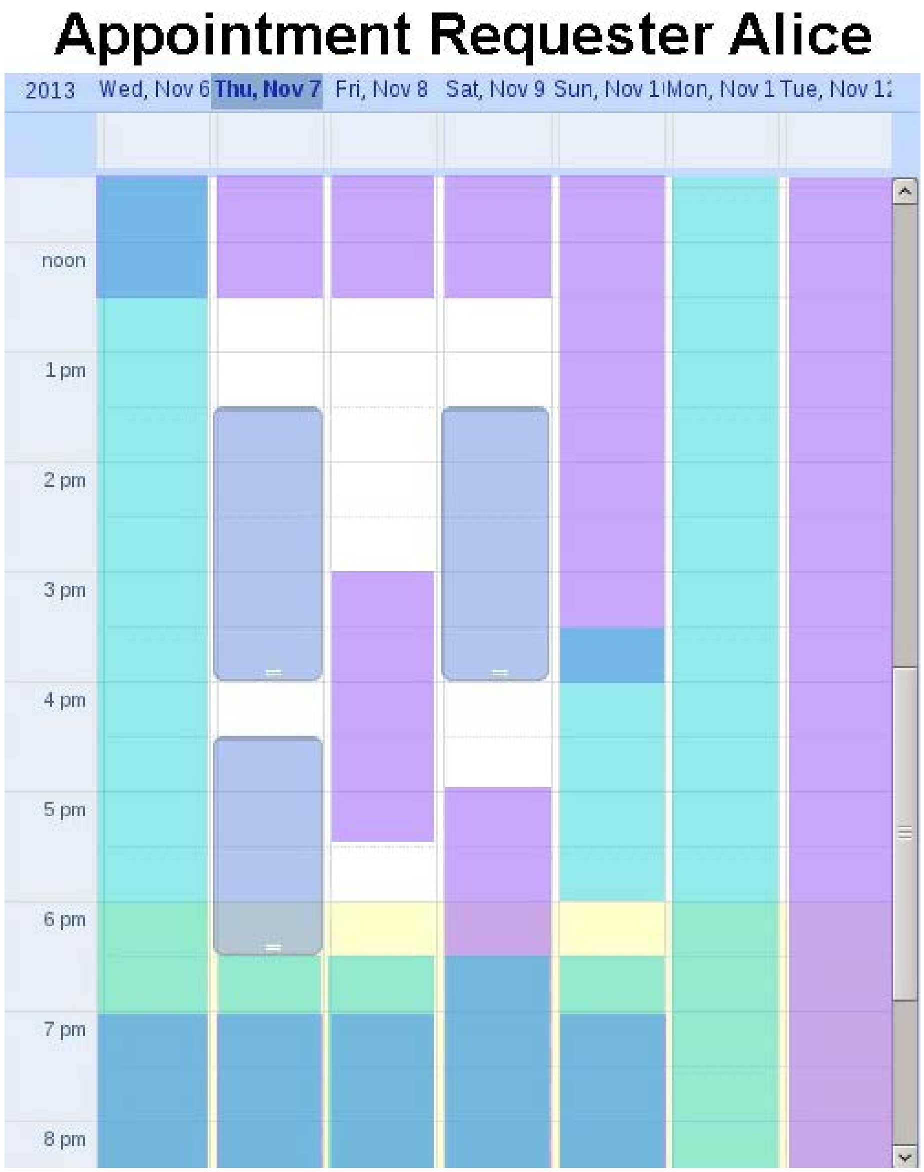

Figure 5 Appointment Proposal with two receivers.

The next step of the defined protocol can be described in either of two cases depending on the amount of receivers in the proposal. In the case of a single receiver, the receiver is only responsible for selecting a single appointment time, is she is satisfied with at least one of the suggested times. This protocol scenario is depicted in Figure 6.

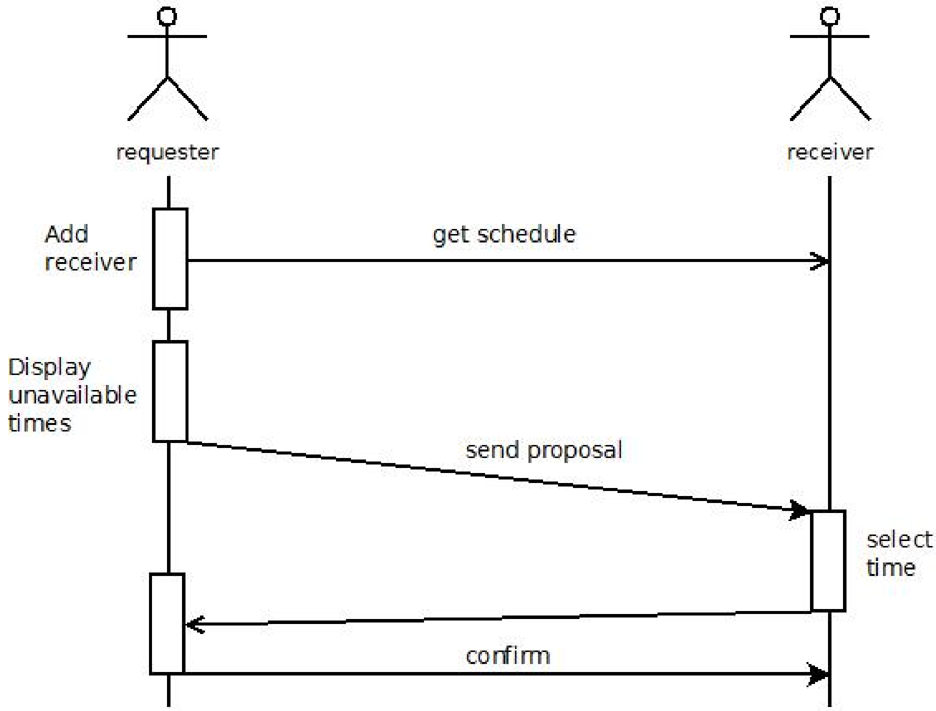

Figure 6 Message sequence chart for appointment proposal involving a single appointment receiver

Figure 7 Receiver only needs to select a single time slot to confirm in the case of single receiver appointment.

If the appointment receiver is not satisfied with any of the time slots in the appointment proposal, the receiver can initiate a counter-proposal, in which he or she can specify alternative appointment times. This scenario is similar to the initial scheduling process on the requester's end, the appointment receiver can also view the unavailable times of the requester. This will increase the likelihood of finding compromise with fewer transactions. It is important to contrast this with the traditional scheduling process where the receivers would have to contact the requester to inform him/her that the suggested appointment is not acceptable. This can present a possible delay if the requester is unavailable when the requester contacts them. Therefore, the requester is not able to negotiate a possible compromise on for an alternative appointment time, and requiring another transaction on the requester's part. The defined schedule protocol remedies this issue by condensing both the appointment rejection and alternative appointments times into a single transaction.

In the second, more complex case of multiple receivers, each receiver can, and is recommended to select multiple available appointment times that he/she considers to be convenient. The screen shot of our prototype for such a selection is shown in Figure 8. The other appointment receivers can then see the previously selected appointment times as depicted in Figure 9, making it possible for them to select only the time slots that are more likely to get an agreement on.

Figure 8 First receiver selects the suggested appointment time that he/she deems convenient

Figure 9 previously submitted preferences of receiver(s) can be viewed by other receivers.

Once all appointment receivers have submitted their respective preferred appointment times, the requestor is provided a view with all suggested appointment times. Each suggested appointment time will contain the name of the appointment receiver and a selected available time in a view similar to what is depicted in Figure 9. The appointment requester must evaluate to see what the best compromise appointment time is. If the appointment requester is not satisfied with the availability of suggested appointment times, he/she may initiate another appointment request process. The protocol scenarios specified above is depicted in Figure 10.

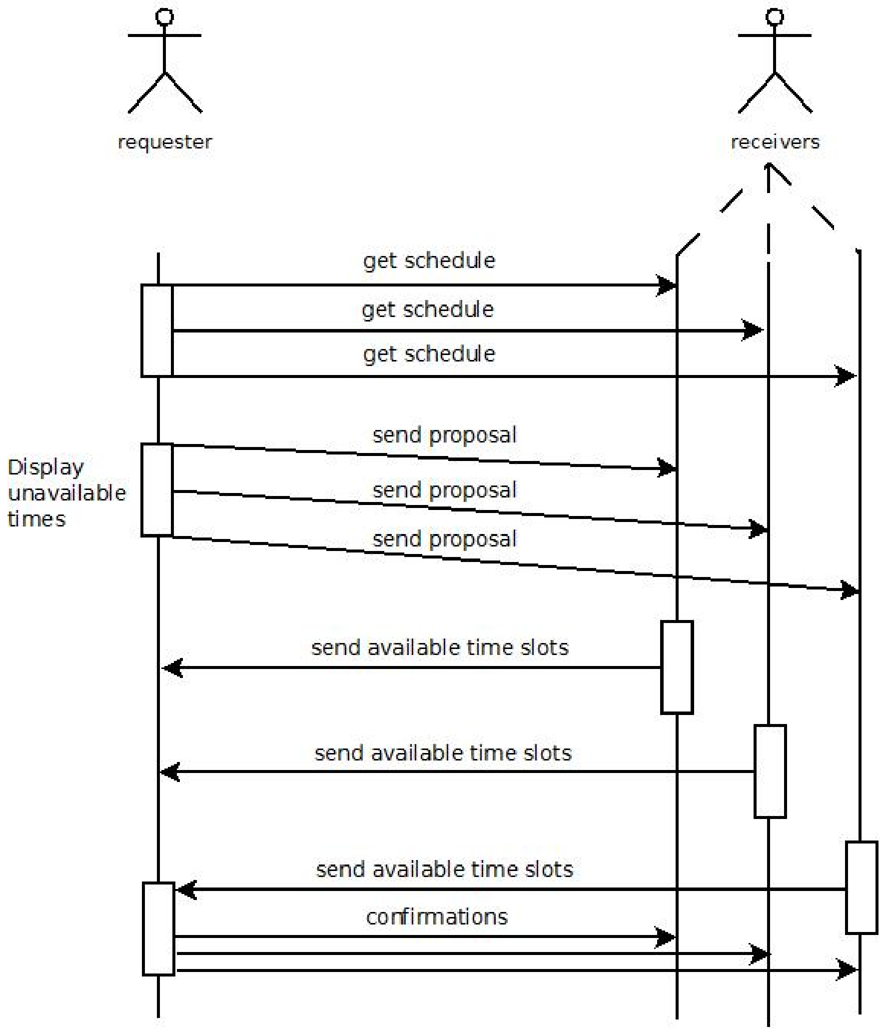

Figure 10 Message sequence chart for appointment proposal involving multiple appointment receivers

IV. Implementation

In this section, we provide details of how our prototype was implemented. Well-designed software consists of three major parts [5]. First is the software implementation which includes the user-interface and business logic. The second is the database where the information is retained. The third component, the servlet, links the three components together by transferring the database information to the actual software. We first describe the design pattern we used for implementation of the user interface.

a. Model-View-Presenter

For the implementation of our prototype, we have used the Model-View-Presenter (MVP) design pattern. We describe how this pattern was used specifically in our prototype.

1. View

View classes can be described as the part of the software that displays information to the user [6]. In this implementation, there are three view major view objects shown in Figure 10. There is a view for an appointment request that has several option fields along with a calendar view where the user can suggest appointment times. The user is also provided the list of pending appointment requests.

View object should contain minimal logic [6]. For instance, the user has the ability to drag appointment times across the screen in the calendar view. By itself, the calendar view is not fully capable of interpreting the user interaction. It simply reads numerical data concerning where the user dragged across the screen and sends it to a presenter. The interaction is analyzed so the view can display an appropriate response.

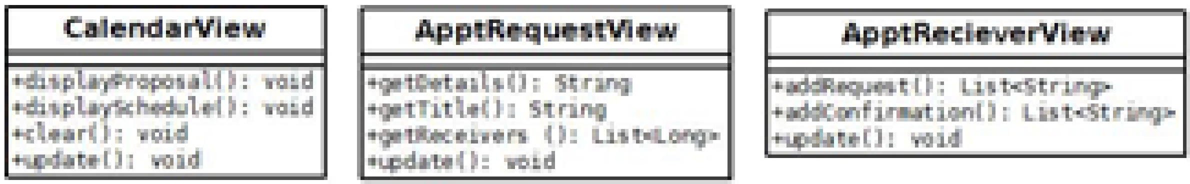

Figure 7 The three view classes that the user interacts with

2. Presenter

The project consists of three presenter classes associated with each view class that are depicted in Figure 11. Each presenter is responsible for populating the information in their respective views so it may be seen by the user [4]. The presenters must also perform logical operations when a user interaction occurs on of the views [7]. In the case of the calendar presenter it is associated with the calendar view. When the user drags a suggested appointment time, the presenter is responsible for translating coordinates to actual dates and times on the calendar. The start and end time are then encapsulated into a model object which represents the software information.

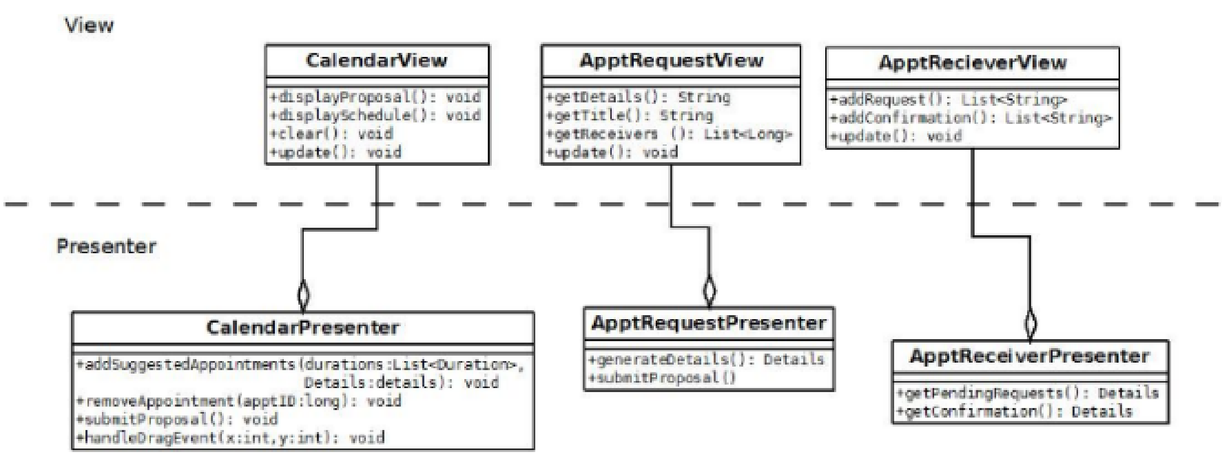

Figure 8 View classes with their respective presenter classes.

3. Model

The model classes represent the actual information that handled by the presenter which the presenter subsequently translates into information that can displayed by the view objects [7]. For this project, there are several major view classes which include appointment times, appointment details, user and counter proposal. The entire appointment process will be represented by a proposal. A proposal will have three major features or aggregated associations. The first feature includes the proposal, which will maintain details concerning the appointment such as subject, location, attached files, etc. Second feature of the proposal is that it also serves as a logging mechanism for the entire appointment process. The log maintains information about requested appointment times. The third feature which is optional, references another object called a counter-proposal. A counter-proposal is only used when the suggested appointment times are rejected by an appointment receiver. The alternative suggested appointment times are encapsulated in this counter proposal object.

Once the proposal process is successfully concluded, the chosen appointment time and the appointment details in the proposal will be encapsulated in a single appointment object. The appointment object also contains an alarm mechanism that can be set by the user to send reminders to him/her via e-mail. Depending on the options selected, the alarm can send reminders hourly, daily, or at a certain specified time prior to the actual appointment.

b. Database

In terms of the database, there are four major tables that are utilized to store information. Their fields and relationships are depicted in an E-R diagram in Figure 12. The first table represents a proposal. A proposal entry (table row) is represented by its identification or primary key. It also maintains the details about the appointment, such as the user who initiated it, the location, subject, etc.

The second table is established to maintain the list of appointment attendees. It contains a field that references an associated proposal and a particular user that is invited to the appointment. An additional field will specify if this user is an optional attendee or mandatory attendee, meaning he/she is an appointment receiver.

The third table maintains suggested appointment times. Appointment time maintains five fields. The first field will reference the proposal entry it is associated with by the proposal ID. The second is a simple integer value that represents the order in which this transaction occurred. A value of one means that this suggested appointment was originally proposed by the appointment requester at the beginning of the appointment proposal process. The next transaction, a possible counter proposal, has a value of two. The third field simply references the user who suggested this appointment time. The next two fields will represent the start and time respectively.

The last crucial table is used to store confirmed appointments. It will maintain the information of appointments that have been confirmed.

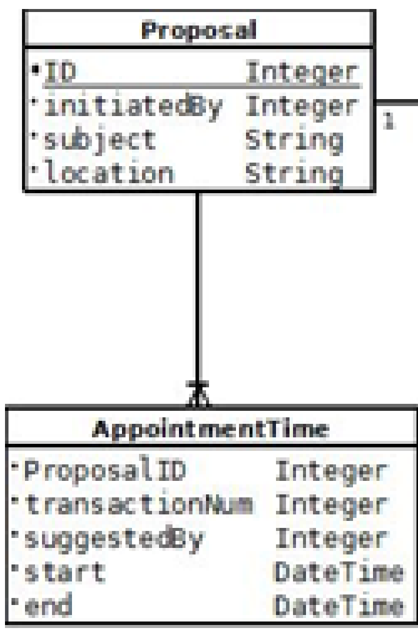

Figure 9 View Database tables representation of information with associated relationships

c. Servlet

A servlet can be simply described as the component that extends the functionality of the server [8]. The sole responsibility of the servlet is to translate the entries in the database table and create corresponding model objects that encapsulate the information of the table entries. The servlet then delivers the model objects to the presenter for computation. As mentioned before, the presenter will then translate the information so that it is displayed into information that is useful in the view. Model classes and Servlets used are depicted in Figure 13.

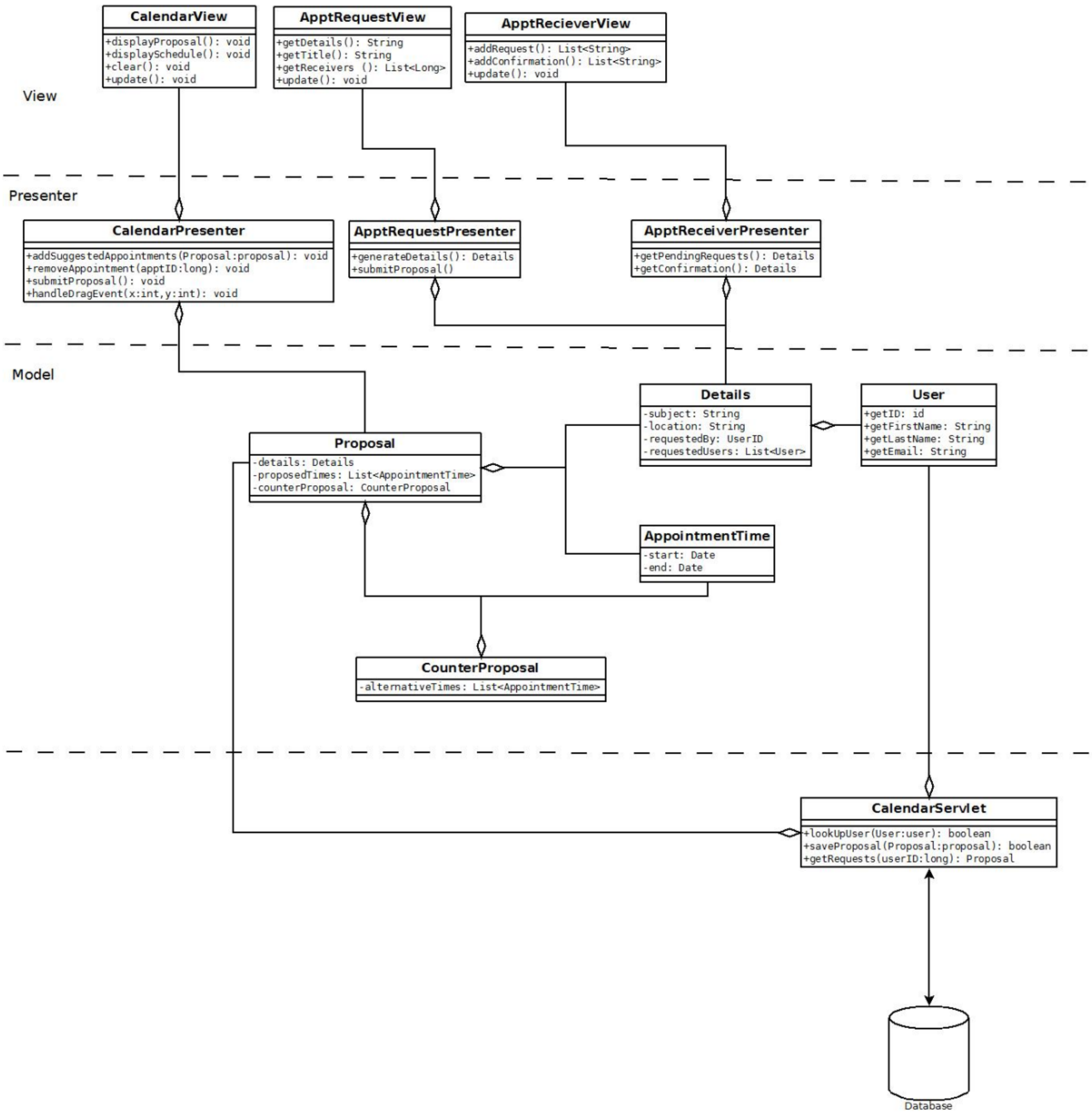

Figure 10 The entire software architecture with all major components

V. Conclusion

In this research, we have defined a methodology of specifying scheduling as a “protocol” and have proposed an enhanced notation for depicting scheduling protocols using a familiar notational paradigm. Whereas the conventional scheduling systems were constrained by lack of interactivity, we have used the opportunities provided by computer automation to redefine how appointments should be proposed and scheduled. This was achieved by proposing a flexible technology environment that allows participants more flexibility by offering them ways of proposing and accepting multiple time slots rather than old-fashioned rigid single-time hit or miss approach. The implemented prototype demonstrates the viability of such a protocol using modern software analysis, specifications and development methodologies. The proposed system enables participants to interact with each other more conveniently during the scheduling and achieve an agreement which much higher likelihood in a single iteration. In the event, where an agreement still cannot be achieved in a single pass, it significantly reduces the effort of the initiator during subsequent iterations.

VI. Future Work

The software product accompanied with this thesis was a prototype that includes implementation of the protocol that was proposed in this paper and was created to justify the viability of it. The goal of this project which will extend beyond the boundaries of the thesis is develop a fully function implementation that can integrate with other commonly used calendar software tools. Hopefully this project will have the potential of making the next leap in office automation by significantly reducing mundane tasks associated with scheduling meetings and events.

VII. References

[1] Google, "Google Calendar," [Online]. Available: www.google.com/calendar.

[2] Microsoft Corporation, "Microsoft Outlook," [Online]. Available: office.microsoft.com/outlook.

[3] TimeTrade Systems, "Solutions: TimeTrade Systems," [Online]. Available: http://www.timetrade.com/solutions.

[4] A. S. Tanenbaum and D. J. Wetherall , in *Computer Networks*, Boston, Prentice Hall, 2010.

[5] R. PRESSMAN, Software Engineering: A Practitioner's Approach, New York, NY: McGraw-Hill., 2010.

[6] M. Potel, "MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java," Wildcrest associates, 1996. [Online]. Available: http://www.wildcrest.com/Potel/Portfolio/mvp.pdf. [Accessed July 2012].

[7] C. Ramsdale, "Mvp-Architecture," Google, Marth 2010. [Online]. Available: http://www.gwtproject.org/articles/mvp-architecture.html.

[8] R. Johnson, E. Gamma, R. Helm and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Boston, Massachusetts: Addison-Wesley, 1995.

[9] M. Fowler, "GUI Architectures," 18 July 2006. [Online]. Available: http://www.martinfowler.com/eaaDev/uiArchs.html.

[10] Oracle Corporation, "Java Servlet Technology Overview," [Online]. Available: http://www.oracle.com/technetwork/java/javaee/servlet/index.html.